

---

# **PyTan Documentation**

***Release 1.0.4***

**Jim Olsen**

March 26, 2015



## CONTENTS

<b>1</b>	<b>Table of Contents</b>	<b>1</b>
1.1	PyTan Introduction . . . . .	1
1.2	pytan package . . . . .	3
1.3	taniumpy package . . . . .	306
1.4	xmltodict module . . . . .	320
1.5	ddt module . . . . .	322
1.6	threaded_http module . . . . .	323
<b>2</b>	<b>Indices and tables</b>	<b>325</b>
	<b>Python Module Index</b>	<b>327</b>
	<b>Index</b>	<b>329</b>



## TABLE OF CONTENTS

### 1.1 PyTan Introduction

#### 1.1.1 Description

This is a set of packages and scripts that provides a simple way for programmatically interfacing with Tanium's SOAP API. It is comprised of four parts:

- Tanium Server SOAP API: The SOAP server embedded into the Tanium server itself, listens on port 444 but is also available via port 443.
- TaniumPy Python Package (`taniumpy`): A python package comprised of a set of python objects automatically generated from the WSDL file that describes the Tanium SOAP API. These python objects handle the serialization and deserialization of XML to and from the Tanium Server SOAP API. Located in `lib/taniumpy`
- PyTan Python Package: (`pytan`): A python package that provides a set of methods to make interfacing with TaniumPy more human friendly. Located in `lib/pytan`
- PyTan Command Line Scripts: A set of command line scripts that utilize the PyTan Package (`pytan`) to make it easy for non-programmers to create/get/delete/ask/deploy objects via the Tanium Server SOAP API.

#### 1.1.2 Why it was created

This was created to solve for the following needs:

- Create a python package (`pytan`) to provide a set of methods for making it easier to programmatically interface with Tanium via the SOAP API.
- Create a set of command line scripts utilizing the `pytan` package that handle the argument parsing, thereby providing non-programmers with command line access to the functionality therein.
- Provide a way to ask questions and get results via Python and/or the command line.
- Provide a way to deploy actions and get results via Python and/or the command line.
- Provide a way to export/import objects in JSON via Python and/or the command line.

#### 1.1.3 Requirements

- Python 2.7: To date PyTan has only been qualified against 2.7.6 and 2.7.9 on Mac/Linux/Windows.
- A working install of Tanium Server 6.2: To date PyTan has only been qualified against 6.2.X versions of Tanium. It does not yet run against 6.5.X versions.

### 1.1.4 Installation

#### Windows Installation

- Download Python 2.7 from <https://www.python.org/downloads/windows/>
- Install Python 2.7 – if you accept the default paths it will install to `C:\Python27`
- Copy PyTan from github to your local machine somewhere
- If you did not accept the default install path for Python 2.7, edit `pytan\winbin\CONFIG.bat` to change the `PYTHON` variable to point to the full path of `python.exe`

#### OS X Installation

- OS X 10.8 and higher come with Python 2.7 out of the box
- Copy PyTan from github to your local machine somewhere

#### Linux Installation

- Ensure Python 2.7 is installed
- Ensure the first `python` binary in your path points to your Python 2.7 installation
- Copy PyTan from github to your local machine somewhere

### 1.1.5 Usage

- For command line usage, refer to Command Line Help Index
- For API Examples, refer to the *pytan API examples*
- For in depth API Documentation, refer to the *pytan package*, especially the *pytan.handler module*

### 1.1.6 Directory Layout

- **EXAMPLES/ directory:** contains a set of example python files that show how to use the various methods exposed by (`pytan`)
- **BUILD/ directory:** contains the scripts that build the HTML and PDF documentation in `doc/`, generate the (`taniumpy`), generate the python examples in `EXAMPLES/`, generate some of the command line scripts in `bin/`, and generate all of the documentation for the command line scripts in `doc/_static/bin_doc`
- **bin/ directory:** contains all of the command line scripts that utilize the (`pytan`)
- **doc/ directory:** contains the HTML and PDF documentation
- **lib/ directory:** contains the python libraries (`pytan`) and (`taniumpy`), as well as other python libraries
- **test/ directory:** contains the unit and functional tests for (`pytan`)
- **winbin/ directory:** contains the Windows batch scripts which wrap around the python command line scripts in `bin/`
- **ZIP\_DIST/ directory:** contains standalone windows executables for certain tools, created by batch files in `BUILD/STATICWINBUILD/`

## 1.2 pytan package

A python package that makes using (`taniumpy`) more human friendly.

```
pytan.__version__ = '1.0.4'
    Version of PyTan

pytan.__copyright__ = 'Copyright 2014 Tanium'
    Copyright for PyTan

pytan.__license__ = 'MIT'
    License for PyTan

pytan.__author__ = 'Jim Olsen <jim.olsen@tanium.com>'
    Author of Pytan
```

### 1.2.1 pytan API examples

#### Pytan api basic handler example

Here is an example for how to instantiate a `pytan.Handler` object.

The username, password, host, and maybe port as well need to be provided on a per Tanium server basis.

#### Example Python Code

```
1  import os
2  import sys
3  sys.dont_write_bytecode = True
4
5  # Determine our script name, script dir
6  my_file = os.path.abspath(sys.argv[0])
7  my_dir = os.path.dirname(my_file)
8
9  # determine the pytan lib dir and add it to the path
10 parent_dir = os.path.dirname(my_dir)
11 pytan_root_dir = os.path.dirname(parent_dir)
12 lib_dir = os.path.join(pytan_root_dir, 'lib')
13 path_adds = [lib_dir]
14
15 for aa in path_adds:
16     if aa not in sys.path:
17         sys.path.append(aa)
18
19
20 # connection info for Tanium Server
21 USERNAME = "Tanium User"
22 PASSWORD = "T@n!um"
23 HOST = "172.16.31.128"
24 PORT = "444"
25
26 # Logging conrols
27 LOGLEVEL = 2
28 DEBUGFORMAT = False
29
30 import tempfile
```

```
31
32 import pytan
33 handler = pytan.Handler(
34     username=USERNAME,
35     password=PASSWORD,
36     host=HOST,
37     port=PORT,
38     loglevel=LOGLEVEL,
39     debugformat=DEBUGFORMAT,
40 )
41
42 print handler
```

### Output from Python Code

```
1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3279
```

### pytan API Valid Question Examples

#### Ask saved question by name in list

Ask a saved question by referencing the name of a saved question in a list of strings.

#### Example Python Code

```
1 import os
2 import sys
3 sys.dont_write_bytecode = True
4
5 # Determine our script name, script dir
6 my_file = os.path.abspath(sys.argv[0])
7 my_dir = os.path.dirname(my_file)
8
9 # determine the pytan lib dir and add it to the path
10 parent_dir = os.path.dirname(my_dir)
11 pytan_root_dir = os.path.dirname(parent_dir)
12 lib_dir = os.path.join(pytan_root_dir, 'lib')
13 path_adds = [lib_dir]
14
15 for aa in path_adds:
16     if aa not in sys.path:
17         sys.path.append(aa)
18
19
20 # connection info for Tanium Server
21 USERNAME = "Tanium User"
22 PASSWORD = "T@n!um"
23 HOST = "172.16.31.128"
24 PORT = "444"
25
26 # Logging conrols
27 LOGLEVEL = 2
```



```

28 DEBUGFORMAT = False
29
30 import tempfile
31
32 import pytan
33 handler = pytan.Handler(
34     username=USERNAME,
35     password=PASSWORD,
36     host=HOST,
37     port=PORT,
38     loglevel=LOGLEVEL,
39     debugformat=DEBUGFORMAT,
40 )
41
42 print handler
43
44 # setup the arguments for the handler method
45 kwargs = {}
46 kwargs["qtype"] = u'saved'
47 kwargs["name"] = [u'Installed Applications']
48
49 # call the handler with the ask method, passing in kwargs for arguments
50 response = handler.ask(**kwargs)
51 import pprint, io
52
53 print ""
54 print "Type of response: ", type(response)
55
56 print ""
57 print "Pretty print of response:"
58 print pprint.pformat(response)
59
60 print ""
61 print "Equivalent Question if it were to be asked in the Tanium Console: "
62 print response['question_object'].query_text
63
64 # create an IO stream to store CSV results to
65 out = io.BytesIO()
66
67 # call the write_csv() method to convert response to CSV and store it in out
68 response['question_results'].write_csv(out, response['question_results'])
69
70 print ""
71 print "CSV Results of response: "
72 out = out.getvalue()
73 if len(out.splitlines()) > 15:
74     out = out.splitlines()[0:15]
75     out.append('..trimmed for brevity..')
76     out = '\n'.join(out)
77 print out

```

### Output from Python Code

```

1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3279
2 2015-03-26 11:37:59,316 INFO      question_progress: Results 100% (Get Installed Applications from al
3
4 Type of response: <type 'dict'>

```

```
5
6 Pretty print of response:
7 {'question_object': <taniumpy.object_types.saved_question.SavedQuestion object at 0x107608b90>,
8  'question_results': <taniumpy.object_types.result_set.ResultSet object at 0x107646690>}}
9
10 Equivalent Question if it were to be asked in the Tanium Console:
11 Get Installed Applications from all machines
12
13 CSV Results of response:
14 Name,Silent Uninstall String,Uninstallable,Version
15 Google Search,nothing,Not Uninstallable,37.0.2062.120
16 Microsoft Chart Converter,nothing,Not Uninstallable,14.4.7
17 Wish,nothing,Not Uninstallable,8.5.9
18 BluetoothUIServer,nothing,Not Uninstallable,4.3.2
19 Time Machine,nothing,Not Uninstallable,1.3
20 AppleGraphicsWarning,nothing,Not Uninstallable,2.3.0
21 Python 2.7 py2exe-0.6.9, ""C:\Python27\Removepy2exe.exe" -u ""C:\Python27\py2exe-wininst.log"",Not
22 soagent,nothing,Not Uninstallable,7.0
23 AinuIM,nothing,Not Uninstallable,1.0
24 ARDAgent,nothing,Not Uninstallable,3.8.2
25 Microsoft Clip Gallery,nothing,Not Uninstallable,14.4.7
26 Pass Viewer,nothing,Not Uninstallable,1.0
27 PressAndHold,nothing,Not Uninstallable,1.2
28 PluginIM,nothing,Not Uninstallable,15
29 ..trimmed for brevity..
```

### Ask saved question by name

Ask a saved question by referencing the name of a saved question in a string.

### Example Python Code

```
1 import os
2 import sys
3 sys.dont_write_bytecode = True
4
5 # Determine our script name, script dir
6 my_file = os.path.abspath(sys.argv[0])
7 my_dir = os.path.dirname(my_file)
8
9 # determine the pytan lib dir and add it to the path
10 parent_dir = os.path.dirname(my_dir)
11 pytan_root_dir = os.path.dirname(parent_dir)
12 lib_dir = os.path.join(pytan_root_dir, 'lib')
13 path_adds = [lib_dir]
14
15 for aa in path_adds:
16     if aa not in sys.path:
17         sys.path.append(aa)
18
19
20 # connection info for Tanium Server
21 USERNAME = "Tanium User"
22 PASSWORD = "T@n!um"
23 HOST = "172.16.31.128"
```

```

24 PORT = "444"
25
26 # Logging conrols
27 LOGLEVEL = 2
28 DEBUGFORMAT = False
29
30 import tempfile
31
32 import pytan
33 handler = pytan.Handler(
34     username=USERNAME,
35     password=PASSWORD,
36     host=HOST,
37     port=PORT,
38     loglevel=LOGLEVEL,
39     debugformat=DEBUGFORMAT,
40 )
41
42 print handler
43
44 # setup the arguments for the handler method
45 kwargs = {}
46 kwargs["qtype"] = u'saved'
47 kwargs["name"] = u'Installed Applications'
48
49 # call the handler with the ask method, passing in kwargs for arguments
50 response = handler.ask(**kwargs)
51 import pprint, io
52
53 print ""
54 print "Type of response: ", type(response)
55
56 print ""
57 print "Pretty print of response:"
58 print pprint.pformat(response)
59
60 print ""
61 print "Equivalent Question if it were to be asked in the Tanium Console: "
62 print response['question_object'].query_text
63
64 # create an IO stream to store CSV results to
65 out = io.BytesIO()
66
67 # call the write_csv() method to convert response to CSV and store it in out
68 response['question_results'].write_csv(out, response['question_results'])
69
70 print ""
71 print "CSV Results of response: "
72 out = out.getvalue()
73 if len(out.splitlines()) > 15:
74     out = out.splitlines()[0:15]
75     out.append('..trimmed for brevity..')
76     out = '\n'.join(out)
77 print out

```

## Output from Python Code

```
1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3279
2 2015-03-26 11:37:59,464 INFO      question_progress: Results 100% (Get Installed Applications from al
3
4 Type of response:  <type 'dict'>
5
6 Pretty print of response:
7 {'question_object': <taniumpy.object_types.saved_question.SavedQuestion object at 0x107593750>,
8  'question_results': <taniumpy.object_types.result_set.ResultSet object at 0x1077e2890>}
9
10 Equivalent Question if it were to be asked in the Tanium Console:
11 Get Installed Applications from all machines
12
13 CSV Results of response:
14 Name,Silent Uninstall String,Uninstallable,Version
15 Google Search,nothing,Not Uninstallable,37.0.2062.120
16 Microsoft Chart Converter,nothing,Not Uninstallable,14.4.7
17 Wish,nothing,Not Uninstallable,8.5.9
18 BluetoothUIServer,nothing,Not Uninstallable,4.3.2
19 Time Machine,nothing,Not Uninstallable,1.3
20 AppleGraphicsWarning,nothing,Not Uninstallable,2.3.0
21 Python 2.7 py2exe-0.6.9,"""C:\Python27\Removepy2exe.exe"" -u ""C:\Python27\py2exe-wininst.log""",Not
22 soagent,nothing,Not Uninstallable,7.0
23 AinuIM,nothing,Not Uninstallable,1.0
24 ARDAgent,nothing,Not Uninstallable,3.8.2
25 Microsoft Clip Gallery,nothing,Not Uninstallable,14.4.7
26 Pass Viewer,nothing,Not Uninstallable,1.0
27 PressAndHold,nothing,Not Uninstallable,1.2
28 PluginIM,nothing,Not Uninstallable,15
29 ..trimmed for brevity..
```

### Ask manual human question simple single sensor

Ask a manual question using human strings by referencing the name of a single sensor in a string.

No sensor filters, sensor parameters, sensor filter options, question filters, or question options supplied.

### Example Python Code

```
1 import os
2 import sys
3 sys.dont_write_bytecode = True
4
5 # Determine our script name, script dir
6 my_file = os.path.abspath(sys.argv[0])
7 my_dir = os.path.dirname(my_file)
8
9 # determine the pytan lib dir and add it to the path
10 parent_dir = os.path.dirname(my_dir)
11 pytan_root_dir = os.path.dirname(parent_dir)
12 lib_dir = os.path.join(pytan_root_dir, 'lib')
13 path_adds = [lib_dir]
14
15 for aa in path_adds:
16     if aa not in sys.path:
17         sys.path.append(aa)
```

```

18
19
20 # connection info for Tanium Server
21 USERNAME = "Tanium User"
22 PASSWORD = "T@n!um"
23 HOST = "172.16.31.128"
24 PORT = "444"
25
26 # Logging conrols
27 LOGLEVEL = 2
28 DEBUGFORMAT = False
29
30 import tempfile
31
32 import pytan
33 handler = pytan.Handler(
34     username=USERNAME,
35     password=PASSWORD,
36     host=HOST,
37     port=PORT,
38     loglevel=LOGLEVEL,
39     debugformat=DEBUGFORMAT,
40 )
41
42 print handler
43
44 # setup the arguments for the handler method
45 kwargs = {}
46 kwargs["sensors"] = u'Computer Name'
47 kwargs["qtype"] = u'manual_human'
48
49 # call the handler with the ask method, passing in kwargs for arguments
50 response = handler.ask(**kwargs)
51 import pprint, io
52
53 print ""
54 print "Type of response: ", type(response)
55
56 print ""
57 print "Pretty print of response:"
58 print pprint.pformat(response)
59
60 print ""
61 print "Equivalent Question if it were to be asked in the Tanium Console: "
62 print response['question_object'].query_text
63
64 # create an IO stream to store CSV results to
65 out = io.BytesIO()
66
67 # call the write_csv() method to convert response to CSV and store it in out
68 response['question_results'].write_csv(out, response['question_results'])
69
70 print ""
71 print "CSV Results of response: "
72 out = out.getvalue()
73 if len(out.splitlines()) > 15:
74     out = out.splitlines()[0:15]
75     out.append('..trimmed for brevity..')

```

```
76     out = '\n'.join(out)
77     print out
```

### Output from Python Code

```
1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3279
2 2015-03-26 11:37:59,628 INFO      question_progress: Results 0% (Get Computer Name from all machines)
3 2015-03-26 11:38:04,641 INFO      question_progress: Results 100% (Get Computer Name from all machines)
4
5 Type of response: <type 'dict'>
6
7 Pretty print of response:
8 {'question_object': <taniumpy.object_types.question.Question object at 0x107662990>,
9  'question_results': <taniumpy.object_types.result_set.ResultSet object at 0x107612e50>}
10
11 Equivalent Question if it were to be asked in the Tanium Console:
12 Get Computer Name from all machines
13
14 CSV Results of response:
15 Computer Name
16 Casus-Belli.local
17 jttanium1.localdomain
```

### Ask manual human question simple multiple sensors

Ask a manual question using human strings by referencing the name of multiple sensors in a list.

No sensor filters, sensor parameters, sensor filter options, question filters, or question options supplied.

### Example Python Code

```
1 import os
2 import sys
3 sys.dont_write_bytecode = True
4
5 # Determine our script name, script dir
6 my_file = os.path.abspath(sys.argv[0])
7 my_dir = os.path.dirname(my_file)
8
9 # determine the pytan lib dir and add it to the path
10 parent_dir = os.path.dirname(my_dir)
11 pytan_root_dir = os.path.dirname(parent_dir)
12 lib_dir = os.path.join(pytan_root_dir, 'lib')
13 path_adds = [lib_dir]
14
15 for aa in path_adds:
16     if aa not in sys.path:
17         sys.path.append(aa)
18
19
20 # connection info for Tanium Server
21 USERNAME = "Tanium User"
22 PASSWORD = "T@n!um"
23 HOST = "172.16.31.128"
```

```

24 PORT = "444"
25
26 # Logging conrols
27 LOGLEVEL = 2
28 DEBUGFORMAT = False
29
30 import tempfile
31
32 import pytan
33 handler = pytan.Handler(
34     username=USERNAME,
35     password=PASSWORD,
36     host=HOST,
37     port=PORT,
38     loglevel=LOGLEVEL,
39     debugformat=DEBUGFORMAT,
40 )
41
42 print handler
43
44 # setup the arguments for the handler method
45 kwargs = {}
46 kwargs["sensors"] = [u'Computer Name', u'Installed Applications']
47 kwargs["qtype"] = u'manual_human'
48
49 # call the handler with the ask method, passing in kwargs for arguments
50 response = handler.ask(**kwargs)
51 import pprint, io
52
53 print ""
54 print "Type of response: ", type(response)
55
56 print ""
57 print "Pretty print of response:"
58 print pprint.pformat(response)
59
60 print ""
61 print "Equivalent Question if it were to be asked in the Tanium Console: "
62 print response['question_object'].query_text
63
64 # create an IO stream to store CSV results to
65 out = io.BytesIO()
66
67 # call the write_csv() method to convert response to CSV and store it in out
68 response['question_results'].write_csv(out, response['question_results'])
69
70 print ""
71 print "CSV Results of response: "
72 out = out.getvalue()
73 if len(out.splitlines()) > 15:
74     out = out.splitlines()[0:15]
75     out.append('..trimmed for brevity..')
76     out = '\n'.join(out)
77 print out

```

## Output from Python Code

```
1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3279
2 2015-03-26 11:38:04,754 INFO      question_progress: Results 0% (Get Computer Name and Installed Appl
3 2015-03-26 11:38:09,776 INFO      question_progress: Results 0% (Get Computer Name and Installed Appl
4 2015-03-26 11:38:14,794 INFO      question_progress: Results 0% (Get Computer Name and Installed Appl
5 2015-03-26 11:38:19,816 INFO      question_progress: Results 100% (Get Computer Name and Installed Ap
6
7 Type of response:  <type 'dict'>
8
9 Pretty print of response:
10 {'question_object': <taniumpy.object_types.question.Question object at 0x107618e10>,
11  'question_results': <taniumpy.object_types.result_set.ResultSet object at 0x1077e7310>}
12
13 Equivalent Question if it were to be asked in the Tanium Console:
14 Get Computer Name and Installed Applications from all machines
15
16 CSV Results of response:
17 Computer Name,Name,Silent Uninstall String,Uninstallable,Version
18 Casus-Belli.local,"Google Search
19 Microsoft Chart Converter
20 Wish
21 BluetoothUIServer
22 Time Machine
23 AppleGraphicsWarning
24 soagent
25 AinuIM
26 ARDAgent
27 Microsoft Clip Gallery
28 Pass Viewer
29 PressAndHold
30 PluginIM
31 UserNotificationCenter
32 ..trimmed for brevity..
```

### Ask manual human question multiple sensors identified by name

Ask a manual question using human strings by referencing the name of multiple sensors and providing a selector that tells pytan explicitly that we are providing a name of a sensor.

No sensor filters, sensor parameters, sensor filter options, question filters, or question options supplied.

### Example Python Code

```
1 import os
2 import sys
3 sys.dont_write_bytecode = True
4
5 # Determine our script name, script dir
6 my_file = os.path.abspath(sys.argv[0])
7 my_dir = os.path.dirname(my_file)
8
9 # determine the pytan lib dir and add it to the path
10 parent_dir = os.path.dirname(my_dir)
11 pytan_root_dir = os.path.dirname(parent_dir)
12 lib_dir = os.path.join(pytan_root_dir, 'lib')
13 path_adds = [lib_dir]
```



```

14
15 for aa in path_adds:
16     if aa not in sys.path:
17         sys.path.append(aa)
18
19
20 # connection info for Tanium Server
21 USERNAME = "Tanium User"
22 PASSWORD = "T@n!um"
23 HOST = "172.16.31.128"
24 PORT = "444"
25
26 # Logging conrols
27 LOGLEVEL = 2
28 DEBUGFORMAT = False
29
30 import tempfile
31
32 import pytan
33 handler = pytan.Handler(
34     username=USERNAME,
35     password=PASSWORD,
36     host=HOST,
37     port=PORT,
38     loglevel=LOGLEVEL,
39     debugformat=DEBUGFORMAT,
40 )
41
42 print handler
43
44 # setup the arguments for the handler method
45 kwargs = {}
46 kwargs["sensors"] = [u'name:Computer Name', u'name:Installed Applications']
47 kwargs["qtype"] = u'manual_human'
48
49 # call the handler with the ask method, passing in kwargs for arguments
50 response = handler.ask(**kwargs)
51 import pprint, io
52
53 print ""
54 print "Type of response: ", type(response)
55
56 print ""
57 print "Pretty print of response:"
58 print pprint.pformat(response)
59
60 print ""
61 print "Equivalent Question if it were to be asked in the Tanium Console: "
62 print response['question_object'].query_text
63
64 # create an IO stream to store CSV results to
65 out = io.BytesIO()
66
67 # call the write_csv() method to convert response to CSV and store it in out
68 response['question_results'].write_csv(out, response['question_results'])
69
70 print ""
71 print "CSV Results of response: "

```

```
72 out = out.getvalue()
73 if len(out.splitlines()) > 15:
74     out = out.splitlines()[0:15]
75     out.append('..trimmed for brevity..')
76     out = '\n'.join(out)
77 print out
```

### Output from Python Code

```
1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3279
2 2015-03-26 11:38:19,994 INFO      question_progress: Results 0% (Get Computer Name and Installed Appl
3 2015-03-26 11:38:25,016 INFO      question_progress: Results 0% (Get Computer Name and Installed Appl
4 2015-03-26 11:38:30,037 INFO      question_progress: Results 100% (Get Computer Name and Installed Ap
5
6 Type of response: <type 'dict'>
7
8 Pretty print of response:
9 {'question_object': <taniumpy.object_types.question.Question object at 0x10760d7d0>,
10  'question_results': <taniumpy.object_types.result_set.ResultSet object at 0x10780ea10>}
11
12 Equivalent Question if it were to be asked in the Tanium Console:
13 Get Computer Name and Installed Applications from all machines
14
15 CSV Results of response:
16 Computer Name,Name,Silent Uninstall String,Uninstallable,Version
17 Casus-Belli.local,"Google Search
18 Microsoft Chart Converter
19 Wish
20 BluetoothUIServer
21 Time Machine
22 AppleGraphicsWarning
23 soagent
24 AinuIM
25 ARDAgent
26 Microsoft Clip Gallery
27 Pass Viewer
28 PressAndHold
29 PluginIM
30 UserNotificationCenter
31 ..trimmed for brevity..
```

### Ask manual human question sensor with parameters and some supplied parameters

Ask a manual question using human strings by referencing the name of a single sensor that takes parameters, but supplying only two of the four parameters that are used by the sensor (and letting pytan automatically determine the appropriate default value for those parameters which require a value and none was supplied).

No sensor filters, sensor parameters, sensor filter options, question filters, or question options supplied.

### Example Python Code

```
1 import os
2 import sys
3 sys.dont_write_bytecode = True
```

```

4
5 # Determine our script name, script dir
6 my_file = os.path.abspath(sys.argv[0])
7 my_dir = os.path.dirname(my_file)
8
9 # determine the pytan lib dir and add it to the path
10 parent_dir = os.path.dirname(my_dir)
11 pytan_root_dir = os.path.dirname(parent_dir)
12 lib_dir = os.path.join(pytan_root_dir, 'lib')
13 path_adds = [lib_dir]
14
15 for aa in path_adds:
16     if aa not in sys.path:
17         sys.path.append(aa)
18
19
20 # connection info for Tanium Server
21 USERNAME = "Tanium User"
22 PASSWORD = "T@n!um"
23 HOST = "172.16.31.128"
24 PORT = "444"
25
26 # Logging conrols
27 LOGLEVEL = 2
28 DEBUGFORMAT = False
29
30 import tempfile
31
32 import pytan
33 handler = pytan.Handler(
34     username=USERNAME,
35     password=PASSWORD,
36     host=HOST,
37     port=PORT,
38     loglevel=LOGLEVEL,
39     debugformat=DEBUGFORMAT,
40 )
41
42 print handler
43
44 # setup the arguments for the handler method
45 kwargs = {}
46 kwargs["sensors"] = u'Folder Name Search with RegEx Match(dirname=Program Files,regex=Microsoft.*)'
47 kwargs["qtype"] = u'manual_human'
48
49 # call the handler with the ask method, passing in kwargs for arguments
50 response = handler.ask(**kwargs)
51 import pprint, io
52
53 print ""
54 print "Type of response: ", type(response)
55
56 print ""
57 print "Pretty print of response:"
58 print pprint.pformat(response)
59
60 print ""
61 print "Equivalent Question if it were to be asked in the Tanium Console: "

```

```

62 print response['question_object'].query_text
63
64 # create an IO stream to store CSV results to
65 out = io.BytesIO()
66
67 # call the write_csv() method to convert response to CSV and store it in out
68 response['question_results'].write_csv(out, response['question_results'])
69
70 print ""
71 print "CSV Results of response: "
72 out = out.getvalue()
73 if len(out.splitlines()) > 15:
74     out = out.splitlines()[0:15]
75     out.append('..trimmed for brevity..')
76     out = '\n'.join(out)
77 print out

```

### Output from Python Code

```

1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3279
2 2015-03-26 11:38:30,171 INFO question_progress: Results 0% (Get Folder Name Search with RegEx Ma
3 2015-03-26 11:38:35,186 INFO question_progress: Results 0% (Get Folder Name Search with RegEx Ma
4 2015-03-26 11:38:40,203 INFO question_progress: Results 0% (Get Folder Name Search with RegEx Ma
5 2015-03-26 11:38:45,220 INFO question_progress: Results 0% (Get Folder Name Search with RegEx Ma
6 2015-03-26 11:38:50,236 INFO question_progress: Results 0% (Get Folder Name Search with RegEx Ma
7 2015-03-26 11:38:55,250 INFO question_progress: Results 0% (Get Folder Name Search with RegEx Ma
8 2015-03-26 11:39:00,268 INFO question_progress: Results 0% (Get Folder Name Search with RegEx Ma
9 2015-03-26 11:39:05,287 INFO question_progress: Results 0% (Get Folder Name Search with RegEx Ma
10 2015-03-26 11:39:10,307 INFO question_progress: Results 0% (Get Folder Name Search with RegEx Ma
11 2015-03-26 11:39:15,323 INFO question_progress: Results 0% (Get Folder Name Search with RegEx Ma
12 2015-03-26 11:39:20,339 INFO question_progress: Results 0% (Get Folder Name Search with RegEx Ma
13 2015-03-26 11:39:25,357 INFO question_progress: Results 0% (Get Folder Name Search with RegEx Ma
14 2015-03-26 11:39:30,378 INFO question_progress: Results 0% (Get Folder Name Search with RegEx Ma
15 2015-03-26 11:39:35,396 INFO question_progress: Results 0% (Get Folder Name Search with RegEx Ma
16 2015-03-26 11:39:40,413 INFO question_progress: Results 100% (Get Folder Name Search with RegEx
17
18 Type of response: <type 'dict'>
19
20 Pretty print of response:
21 {'question_object': <taniumpy.object_types.question.Question object at 0x10760fb50>,
22  'question_results': <taniumpy.object_types.result_set.ResultSet object at 0x10763a9d0>}
23
24 Equivalent Question if it were to be asked in the Tanium Console:
25 Get Folder Name Search with RegEx Match[No, Program Files, No, , Microsoft.*] from all machines
26
27 CSV Results of response:
28 "Folder Name Search with RegEx Match[No, Program Files, No, , Microsoft.*]"
29 C:\Program Files\Microsoft SQL Server\110\Setup Bootstrap\Update Cache\KB2674319\ServicePack\1033_en
30 C:\Program Files\Microsoft SQL Server\110\Setup Bootstrap\Update Cache\KB2958429\ServicePack\1033_en
31 C:\Program Files\Microsoft SQL Server\110\Setup Bootstrap\Update Cache\KB2958429\ServicePack\1033_en
32 C:\Program Files\VMware\VMware Tools\plugins\vmtoolsd
33 C:\Program Files\Common Files\Microsoft Shared\VS7Debug
34 C:\Program Files\Tanium\Tanium Server\Apache24>manual\style
35 C:\Program Files\Microsoft SQL Server\110\Setup Bootstrap\Update Cache\KB2674319\ServicePack\1033_en
36 C:\Program Files\Microsoft SQL Server\110\Setup Bootstrap\Log\20150306_224415\resources
37 C:\Program Files\Tanium\Tanium Server\Apache24\htdocs\console\history
38 C:\Program Files\Windows Portable Devices

```

```

39 C:\Program Files\Microsoft SQL Server\110\Setup Bootstrap\Update Cache\KB2977326\GDR\1033_enu_lp\x64
40 C:\Program Files\Microsoft SQL Server\110\Setup Bootstrap\Update Cache\KB2674319\ServicePack\1033_en
41 C:\Program Files\Common Files\VMware\Drivers\vmci\sockets\include
42 C:\Program Files\Common Files\Microsoft Shared\ink\ar-SA
43 ..trimmed for brevity..

```

### Ask manual human question sensor without parameters and supplied parameters

Ask a manual question using human strings by referencing the name of a single sensor that does NOT take parameters, but supplying parameters anyways (which will be ignored since the sensor does not take parameters).

No sensor filters, sensor filter options, question filters, or question options supplied.

### Example Python Code

```

1  import os
2  import sys
3  sys.dont_write_bytecode = True
4
5  # Determine our script name, script dir
6  my_file = os.path.abspath(sys.argv[0])
7  my_dir = os.path.dirname(my_file)
8
9  # determine the pytan lib dir and add it to the path
10 parent_dir = os.path.dirname(my_dir)
11 pytan_root_dir = os.path.dirname(parent_dir)
12 lib_dir = os.path.join(pytan_root_dir, 'lib')
13 path_adds = [lib_dir]
14
15 for aa in path_adds:
16     if aa not in sys.path:
17         sys.path.append(aa)
18
19
20 # connection info for Tanium Server
21 USERNAME = "Tanium User"
22 PASSWORD = "T@n!um"
23 HOST = "172.16.31.128"
24 PORT = "444"
25
26 # Logging conrols
27 LOGLEVEL = 2
28 DEBUGFORMAT = False
29
30 import tempfile
31
32 import pytan
33 handler = pytan.Handler(
34     username=USERNAME,
35     password=PASSWORD,
36     host=HOST,
37     port=PORT,
38     loglevel=LOGLEVEL,
39     debugformat=DEBUGFORMAT,
40 )
41

```

```
42 print handler
43
44 # setup the arguments for the handler method
45 kwargs = {}
46 kwargs["sensors"] = u'Computer Name{fake=Dweedle}'
47 kwargs["qtype"] = u'manual_human'
48
49 # call the handler with the ask method, passing in kwargs for arguments
50 response = handler.ask(**kwargs)
51 import pprint, io
52
53 print ""
54 print "Type of response: ", type(response)
55
56 print ""
57 print "Pretty print of response:"
58 print pprint.pformat(response)
59
60 print ""
61 print "Equivalent Question if it were to be asked in the Tanium Console: "
62 print response['question_object'].query_text
63
64 # create an IO stream to store CSV results to
65 out = io.BytesIO()
66
67 # call the write_csv() method to convert response to CSV and store it in out
68 response['question_results'].write_csv(out, response['question_results'])
69
70 print ""
71 print "CSV Results of response: "
72 out = out.getvalue()
73 if len(out.splitlines()) > 15:
74     out = out.splitlines()[0:15]
75     out.append('..trimmed for brevity..')
76     out = '\n'.join(out)
77 print out
```

## Output from Python Code

```
1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3279
2 2015-03-26 11:39:40,593 INFO question_progress: Results 0% (Get Computer Name[Dweedle] from all
3 2015-03-26 11:39:45,610 INFO question_progress: Results 0% (Get Computer Name[Dweedle] from all
4 2015-03-26 11:39:50,628 INFO question_progress: Results 0% (Get Computer Name[Dweedle] from all
5 2015-03-26 11:39:55,642 INFO question_progress: Results 0% (Get Computer Name[Dweedle] from all
6 2015-03-26 11:40:00,657 INFO question_progress: Results 0% (Get Computer Name[Dweedle] from all
7 2015-03-26 11:40:05,675 INFO question_progress: Results 0% (Get Computer Name[Dweedle] from all
8 2015-03-26 11:40:10,689 INFO question_progress: Results 0% (Get Computer Name[Dweedle] from all
9 2015-03-26 11:40:15,705 INFO question_progress: Results 0% (Get Computer Name[Dweedle] from all
10 2015-03-26 11:40:20,718 INFO question_progress: Results 0% (Get Computer Name[Dweedle] from all
11 2015-03-26 11:40:25,732 INFO question_progress: Results 0% (Get Computer Name[Dweedle] from all
12 2015-03-26 11:40:30,747 INFO question_progress: Results 0% (Get Computer Name[Dweedle] from all
13 2015-03-26 11:40:35,763 INFO question_progress: Results 0% (Get Computer Name[Dweedle] from all
14 2015-03-26 11:40:40,778 INFO question_progress: Results 0% (Get Computer Name[Dweedle] from all
15 2015-03-26 11:40:45,791 INFO question_progress: Results 0% (Get Computer Name[Dweedle] from all
16 2015-03-26 11:40:50,806 INFO question_progress: Results 0% (Get Computer Name[Dweedle] from all
17 2015-03-26 11:40:55,823 INFO question_progress: Results 50% (Get Computer Name[Dweedle] from all
18 2015-03-26 11:41:00,839 INFO question_progress: Results 100% (Get Computer Name[Dweedle] from all
```

```

19
20 Type of response: <type 'dict'>
21
22 Pretty print of response:
23 {'question_object': <taniumpy.object_types.question.Question object at 0x10762aa10>,
24  'question_results': <taniumpy.object_types.result_set.ResultSet object at 0x10760f290>}
25
26 Equivalent Question if it were to be asked in the Tanium Console:
27 Get Computer Name[Dweedle] from all machines
28
29 CSV Results of response:
30 Computer Name[Dweedle]
31 [no results]
32 JTANIUM1

```

### Ask manual human question multiple sensors with parameters and some supplied parameters

Ask a manual question using human strings by referencing the name of multiple sensors, one that takes parameters, but supplying only two of the four parameters that are used by the sensor (and letting pytan automatically determine the appropriate default value for those parameters which require a value and none was supplied), and one that does not take parameters.

No sensor filters, question filters, or question options supplied.

### Example Python Code

```

1  import os
2  import sys
3  sys.dont_write_bytecode = True
4
5  # Determine our script name, script dir
6  my_file = os.path.abspath(sys.argv[0])
7  my_dir = os.path.dirname(my_file)
8
9  # determine the pytan lib dir and add it to the path
10 parent_dir = os.path.dirname(my_dir)
11 pytan_root_dir = os.path.dirname(parent_dir)
12 lib_dir = os.path.join(pytan_root_dir, 'lib')
13 path_adds = [lib_dir]
14
15 for aa in path_adds:
16     if aa not in sys.path:
17         sys.path.append(aa)
18
19
20 # connection info for Tanium Server
21 USERNAME = "Tanium User"
22 PASSWORD = "T@n!um"
23 HOST = "172.16.31.128"
24 PORT = "444"
25
26 # Logging conrols
27 LOGLEVEL = 2
28 DEBUGFORMAT = False
29

```

```
30 import tempfile
31
32 import pytan
33 handler = pytan.Handler(
34     username=USERNAME,
35     password=PASSWORD,
36     host=HOST,
37     port=PORT,
38     loglevel=LOGLEVEL,
39     debugformat=DEBUGFORMAT,
40 )
41
42 print handler
43
44 # setup the arguments for the handler method
45 kwargs = {}
46 kwargs["sensors"] = [u'Folder Name Search with RegEx Match{dirname=Program Files,regex=Microsoft.*}'
47     u'Computer Name']
48 kwargs["qtype"] = u'manual_human'
49
50 # call the handler with the ask method, passing in kwargs for arguments
51 response = handler.ask(**kwargs)
52 import pprint, io
53
54 print ""
55 print "Type of response: ", type(response)
56
57 print ""
58 print "Pretty print of response:"
59 print pprint.pformat(response)
60
61 print ""
62 print "Equivalent Question if it were to be asked in the Tanium Console: "
63 print response['question_object'].query_text
64
65 # create an IO stream to store CSV results to
66 out = io.BytesIO()
67
68 # call the write_csv() method to convert response to CSV and store it in out
69 response['question_results'].write_csv(out, response['question_results'])
70
71 print ""
72 print "CSV Results of response: "
73 out = out.getvalue()
74 if len(out.splitlines()) > 15:
75     out = out.splitlines()[0:15]
76     out.append('..trimmed for brevity..')
77     out = '\n'.join(out)
78 print out
```

### Output from Python Code

```
1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3279
2 2015-03-26 11:41:00,959 INFO      question_progress: Results 0% (Get Computer Name from all machines)
3 2015-03-26 11:41:05,976 INFO      question_progress: Results 0% (Get Computer Name from all machines)
4 2015-03-26 11:41:10,992 INFO      question_progress: Results 0% (Get Computer Name from all machines)
5 2015-03-26 11:41:16,155 INFO      question_progress: Results 50% (Get Computer Name from all machines)
```



```

6 2015-03-26 11:41:21,169 INFO      question_progress: Results 100% (Get Computer Name from all machine
7
8 Type of response: <type 'dict'>
9
10 Pretty print of response:
11 {'question_object': <taniumpy.object_types.question.Question object at 0x1075b8c10>,
12  'question_results': <taniumpy.object_types.result_set.ResultSet object at 0x107623550>}
13
14 Equivalent Question if it were to be asked in the Tanium Console:
15 Get Computer Name from all machines
16
17 CSV Results of response:
18 Computer Name,"Folder Name Search with RegEx Match[No, Program Files, No, , Microsoft.*]"
19 Casus-Belli.local,Windows Only
20 jtanium1.localdomain,"C:\Program Files\Microsoft SQL Server\110\Setup Bootstrap\Update Cache\KB26743
21 C:\Program Files\Microsoft SQL Server\110\Setup Bootstrap\Update Cache\KB2958429\ServicePack\1033_en
22 C:\Program Files\Microsoft SQL Server\110\Setup Bootstrap\Update Cache\KB2958429\ServicePack\1033_en
23 C:\Program Files\VMware\VMware Tools\plugins\vmtoolsd
24 C:\Program Files\Common Files\Microsoft Shared\VS7Debug
25 C:\Program Files\Tanium\Tanium Server\Apache24>manual\style
26 C:\Program Files\Microsoft SQL Server\110\Setup Bootstrap\Update Cache\KB2674319\ServicePack\1033_en
27 C:\Program Files\Microsoft SQL Server\110\Setup Bootstrap\Log\20150306_224415\resources
28 C:\Program Files\Tanium\Tanium Server\Apache24\htdocs\console\history
29 C:\Program Files\Windows Portable Devices
30 C:\Program Files\Microsoft SQL Server\110\Setup Bootstrap\Update Cache\KB2977326\GDR\1033_enu_lp\x64
31 C:\Program Files\Microsoft SQL Server\110\Setup Bootstrap\Update Cache\KB2674319\ServicePack\1033_en
32 C:\Program Files\Common Files\VMware\Drivers\vmtoolsd\sockets\include
33 ..trimmed for brevity..

```

### Ask manual human question sensor with parameters and no supplied parameters

Ask a manual question using human strings by referencing the name of a single sensor that takes parameters, but not supplying any parameters (and letting pytan automatically determine the appropriate default value for those parameters which require a value).

No sensor filters, sensor parameters, sensor filter options, question filters, or question options supplied.

### Example Python Code

```

1 import os
2 import sys
3 sys.dont_write_bytecode = True
4
5 # Determine our script name, script dir
6 my_file = os.path.abspath(sys.argv[0])
7 my_dir = os.path.dirname(my_file)
8
9 # determine the pytan lib dir and add it to the path
10 parent_dir = os.path.dirname(my_dir)
11 pytan_root_dir = os.path.dirname(parent_dir)
12 lib_dir = os.path.join(pytan_root_dir, 'lib')
13 path_adds = [lib_dir]
14
15 for aa in path_adds:
16     if aa not in sys.path:

```

```
17         sys.path.append(aa)
18
19
20     # connection info for Tanium Server
21     USERNAME = "Tanium User"
22     PASSWORD = "T@n!um"
23     HOST = "172.16.31.128"
24     PORT = "444"
25
26     # Logging conrols
27     LOGLEVEL = 2
28     DEBUGFORMAT = False
29
30     import tempfile
31
32     import pytan
33     handler = pytan.Handler(
34         username=USERNAME,
35         password=PASSWORD,
36         host=HOST,
37         port=PORT,
38         loglevel=LOGLEVEL,
39         debugformat=DEBUGFORMAT,
40     )
41
42     print handler
43
44     # setup the arguments for the handler method
45     kwargs = {}
46     kwargs["sensors"] = u'Folder Name Search with RegEx Match'
47     kwargs["qtype"] = u'manual_human'
48
49     # call the handler with the ask method, passing in kwargs for arguments
50     response = handler.ask(**kwargs)
51     import pprint, io
52
53     print ""
54     print "Type of response: ", type(response)
55
56     print ""
57     print "Pretty print of response:"
58     print pprint.pformat(response)
59
60     print ""
61     print "Equivalent Question if it were to be asked in the Tanium Console: "
62     print response['question_object'].query_text
63
64     # create an IO stream to store CSV results to
65     out = io.BytesIO()
66
67     # call the write_csv() method to convert response to CSV and store it in out
68     response['question_results'].write_csv(out, response['question_results'])
69
70     print ""
71     print "CSV Results of response: "
72     out = out.getvalue()
73     if len(out.splitlines()) > 15:
74         out = out.splitlines()[0:15]
```

```

75     out.append('..trimmed for brevity..')
76     out = '\n'.join(out)
77 print out

```

### Output from Python Code

```

1  Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3279
2  2015-03-26 11:41:21,293 INFO      question_progress: Results 0% (Get Folder Name Search with RegEx Ma
3  2015-03-26 11:41:26,311 INFO      question_progress: Results 0% (Get Folder Name Search with RegEx Ma
4  2015-03-26 11:41:31,325 INFO      question_progress: Results 0% (Get Folder Name Search with RegEx Ma
5  2015-03-26 11:41:36,344 INFO      question_progress: Results 0% (Get Folder Name Search with RegEx Ma
6  2015-03-26 11:41:41,362 INFO      question_progress: Results 0% (Get Folder Name Search with RegEx Ma
7  2015-03-26 11:41:46,376 INFO      question_progress: Results 0% (Get Folder Name Search with RegEx Ma
8  2015-03-26 11:41:51,393 INFO      question_progress: Results 0% (Get Folder Name Search with RegEx Ma
9  2015-03-26 11:41:56,410 INFO      question_progress: Results 0% (Get Folder Name Search with RegEx Ma
10 2015-03-26 11:42:01,427 INFO      question_progress: Results 0% (Get Folder Name Search with RegEx Ma
11 2015-03-26 11:42:06,442 INFO      question_progress: Results 0% (Get Folder Name Search with RegEx Ma
12 2015-03-26 11:42:11,461 INFO      question_progress: Results 0% (Get Folder Name Search with RegEx Ma
13 2015-03-26 11:42:16,479 INFO      question_progress: Results 50% (Get Folder Name Search with RegEx M
14 2015-03-26 11:42:21,503 INFO      question_progress: Results 50% (Get Folder Name Search with RegEx M
15 2015-03-26 11:42:26,523 INFO      question_progress: Results 50% (Get Folder Name Search with RegEx M
16 2015-03-26 11:42:31,545 INFO      question_progress: Results 50% (Get Folder Name Search with RegEx M
17 2015-03-26 11:42:36,564 INFO      question_progress: Results 100% (Get Folder Name Search with RegEx
18
19 Type of response: <type 'dict'>
20
21 Pretty print of response:
22 {'question_object': <taniumpy.object_types.question.Question object at 0x1075b8210>,
23  'question_results': <taniumpy.object_types.result_set.ResultSet object at 0x107817210>}
24
25 Equivalent Question if it were to be asked in the Tanium Console:
26 Get Folder Name Search with RegEx Match[No, , No, ] from all machines
27
28 CSV Results of response:
29 Count,"Folder Name Search with RegEx Match[No, , No, ]"
30 24707,[too many results]
31 1,C:\Windows\winsxs\amd64_microsoft-windows-s..structure.resources_31bf3856ad364e35_6.1.7600.16385_e
32 1,C:\Windows\winsxs\x86_microsoft-windows-e..host-authenticator_31bf3856ad364e35_6.1.7601.17514_non
33 1,C:\Windows\winsxs\amd64_microsoft-windows-ocspsvc_31bf3856ad364e35_6.1.7601.22807_none_3bfeae72930
34 1,C:\Windows\winsxs\amd64_microsoft-windows-c..ityclient.resources_31bf3856ad364e35_6.1.7601.22865_e
35 1,C:\Program Files\Microsoft SQL Server\110\Setup Bootstrap\Update Cache\KB2674319\ServicePack\1033_
36 1,C:\Windows\assembly\NativeImages_v2.0.50727_64\System.Xml
37 1,C:\Windows\winsxs\amd64_microsoft-windows-scripting.resources_31bf3856ad364e35_6.1.7600.16385_en-u
38 1,C:\Windows\winsxs\x86_microsoft-windows-mlang.resources_31bf3856ad364e35_6.1.7600.16385_ru-ru_cf3a
39 1,C:\Windows\winsxs\amd64_microsoft-windows-ie-internetexplorer_31bf3856ad364e35_11.2.9600.17041_non
40 1,C:\Windows\Installer\$\PatchCache$\Managed\1F1FFB6230C555C4C9C7DF5688A9AF07
41 1,C:\Program Files (x86)\Windows Defender
42 1,C:\Users\Jim Olsen\AppData\Local\Google
43 1,C:\Windows\winsxs\x86_microsoft-windows-e..nt-client.resources_31bf3856ad364e35_6.1.7600.16385_en-
44 ..trimmed for brevity..

```

### Ask manual human question sensor with parameters and filter

Ask a manual question using human strings by referencing the name of a single sensor that takes parameters, but supplying only two of the four parameters that are used by the sensor.

Also supply a sensor filter that limits the column data that is shown to values that match the regex `'.*Shared.*'`.

No sensor filter options, question filters, or question options supplied.

### Example Python Code

```
1  import os
2  import sys
3  sys.dont_write_bytecode = True
4
5  # Determine our script name, script dir
6  my_file = os.path.abspath(sys.argv[0])
7  my_dir = os.path.dirname(my_file)
8
9  # determine the pytan lib dir and add it to the path
10 parent_dir = os.path.dirname(my_dir)
11 pytan_root_dir = os.path.dirname(parent_dir)
12 lib_dir = os.path.join(pytan_root_dir, 'lib')
13 path_adds = [lib_dir]
14
15 for aa in path_adds:
16     if aa not in sys.path:
17         sys.path.append(aa)
18
19
20 # connection info for Tanium Server
21 USERNAME = "Tanium User"
22 PASSWORD = "T@n!um"
23 HOST = "172.16.31.128"
24 PORT = "444"
25
26 # Logging conrols
27 LOGLEVEL = 2
28 DEBUGFORMAT = False
29
30 import tempfile
31
32 import pytan
33 handler = pytan.Handler(
34     username=USERNAME,
35     password=PASSWORD,
36     host=HOST,
37     port=PORT,
38     loglevel=LOGLEVEL,
39     debugformat=DEBUGFORMAT,
40 )
41
42 print handler
43
44 # setup the arguments for the handler method
45 kwargs = {}
46 kwargs["sensors"] = u'Folder Name Search with RegEx Match(dirname=Program Files,regex=Microsoft.*),
47 kwargs["qtype"] = u'manual_human'
48
49 # call the handler with the ask method, passing in kwargs for arguments
50 response = handler.ask(**kwargs)
51 import pprint, io
52
```

```

53 print ""
54 print "Type of response: ", type(response)
55
56 print ""
57 print "Pretty print of response:"
58 pprint.pprint(response)
59
60 print ""
61 print "Equivalent Question if it were to be asked in the Tanium Console: "
62 print response['question_object'].query_text
63
64 # create an IO stream to store CSV results to
65 out = io.BytesIO()
66
67 # call the write_csv() method to convert response to CSV and store it in out
68 response['question_results'].write_csv(out, response['question_results'])
69
70 print ""
71 print "CSV Results of response: "
72 out = out.getvalue()
73 if len(out.splitlines()) > 15:
74     out = out.splitlines()[0:15]
75     out.append('..trimmed for brevity..')
76     out = '\n'.join(out)
77 print out

```

### Output from Python Code

```

1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3279
2 2015-03-26 11:42:36,899 INFO question_progress: Results 0% (Get Folder Name Search with RegEx Ma
3 2015-03-26 11:42:41,918 INFO question_progress: Results 0% (Get Folder Name Search with RegEx Ma
4 2015-03-26 11:42:46,933 INFO question_progress: Results 0% (Get Folder Name Search with RegEx Ma
5 2015-03-26 11:42:51,950 INFO question_progress: Results 0% (Get Folder Name Search with RegEx Ma
6 2015-03-26 11:42:56,970 INFO question_progress: Results 0% (Get Folder Name Search with RegEx Ma
7 2015-03-26 11:43:01,988 INFO question_progress: Results 0% (Get Folder Name Search with RegEx Ma
8 2015-03-26 11:43:07,011 INFO question_progress: Results 0% (Get Folder Name Search with RegEx Ma
9 2015-03-26 11:43:12,029 INFO question_progress: Results 0% (Get Folder Name Search with RegEx Ma
10 2015-03-26 11:43:17,053 INFO question_progress: Results 0% (Get Folder Name Search with RegEx Ma
11 2015-03-26 11:43:22,076 INFO question_progress: Results 0% (Get Folder Name Search with RegEx Ma
12 2015-03-26 11:43:27,096 INFO question_progress: Results 0% (Get Folder Name Search with RegEx Ma
13 2015-03-26 11:43:32,118 INFO question_progress: Results 100% (Get Folder Name Search with RegEx
14
15 Type of response: <type 'dict'>
16
17 Pretty print of response:
18 {'question_object': <taniumpy.object_types.question.Question object at 0x1075b84d0>,
19  'question_results': <taniumpy.object_types.result_set.ResultSet object at 0x10761bc10>}
20
21 Equivalent Question if it were to be asked in the Tanium Console:
22 Get Folder Name Search with RegEx Match[No, Program Files, No, , Microsoft.*] contains "Shared" from
23
24 CSV Results of response:
25 "Folder Name Search with RegEx Match[No, Program Files, No, , Microsoft.*]"
26 [no results]
27 C:\Program Files\Common Files\Microsoft Shared\VS7Debug
28 C:\Program Files\Common Files\Microsoft Shared\ink\ar-SA
29 C:\Program Files\Common Files\Microsoft Shared\ink\ru-RU

```

```
30 C:\Program Files\Common Files\Microsoft Shared\ink\fsdefinitions\keypad
31 C:\Program Files\Common Files\Microsoft Shared\ink
32 C:\Program Files\Common Files\Microsoft Shared\ink\sv-SE
33 C:\Program Files\Microsoft SQL Server\110\Setup Bootstrap\Update Cache\KB2977326\GDR\1033_enu_lp\x64
34 C:\Program Files\Common Files\Microsoft Shared\ink\uk-UA
35 C:\Program Files\Common Files\Microsoft Shared\ink\sl-SI
36 C:\Program Files\Common Files\Microsoft Shared\ink\hu-HU
37 C:\Program Files\Common Files\Microsoft Shared\ink\zh-TW
38 C:\Program Files\Common Files\Microsoft Shared\ink\zh-CN
39 C:\Program Files\Common Files\Microsoft Shared\ink\fi-FI
40 ..trimmed for brevity..
```

### Ask manual human question sensor with filter and 3 options

Ask a manual question using human strings by referencing the name of a single sensor.

Also supply a sensor filter that limits the column data that is shown to values that contain Windows (which is short hand for regex match against `.*Windows.*`).

Also supply filter options that re-fetches any cached data that is older than 3600 seconds, matches all values supplied in the filter, and ignores case for any value match of the filter.

No sensor paramaters, question filters, or question options supplied.

### Example Python Code

```
1  import os
2  import sys
3  sys.dont_write_bytecode = True
4
5  # Determine our script name, script dir
6  my_file = os.path.abspath(sys.argv[0])
7  my_dir = os.path.dirname(my_file)
8
9  # determine the pytan lib dir and add it to the path
10 parent_dir = os.path.dirname(my_dir)
11 pytan_root_dir = os.path.dirname(parent_dir)
12 lib_dir = os.path.join(pytan_root_dir, 'lib')
13 path_adds = [lib_dir]
14
15 for aa in path_adds:
16     if aa not in sys.path:
17         sys.path.append(aa)
18
19
20 # connection info for Tanium Server
21 USERNAME = "Tanium User"
22 PASSWORD = "T@n!um"
23 HOST = "172.16.31.128"
24 PORT = "444"
25
26 # Logging conrols
27 LOGLEVEL = 2
28 DEBUGFORMAT = False
29
30 import tempfile
```

```

31
32 import pytan
33 handler = pytan.Handler(
34     username=USERNAME,
35     password=PASSWORD,
36     host=HOST,
37     port=PORT,
38     loglevel=LOGLEVEL,
39     debugformat=DEBUGFORMAT,
40 )
41
42 print handler
43
44 # setup the arguments for the handler method
45 kwargs = {}
46 kwargs["sensors"] = u'Operating System, that contains:Windows, opt:match_all_values, opt:ignore_case
47 kwargs["qtype"] = u'manual_human'
48
49 # call the handler with the ask method, passing in kwargs for arguments
50 response = handler.ask(**kwargs)
51 import pprint, io
52
53 print ""
54 print "Type of response: ", type(response)
55
56 print ""
57 print "Pretty print of response:"
58 print pprint.pformat(response)
59
60 print ""
61 print "Equivalent Question if it were to be asked in the Tanium Console: "
62 print response['question_object'].query_text
63
64 # create an IO stream to store CSV results to
65 out = io.BytesIO()
66
67 # call the write_csv() method to convert response to CSV and store it in out
68 response['question_results'].write_csv(out, response['question_results'])
69
70 print ""
71 print "CSV Results of response: "
72 out = out.getvalue()
73 if len(out.splitlines()) > 15:
74     out = out.splitlines()[0:15]
75     out.append('..trimmed for brevity..')
76     out = '\n'.join(out)
77 print out

```

### Output from Python Code

```

1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3279
2 2015-03-26 11:43:32,326 INFO      question_progress: Results 0% (Get Operating System contains "Windo
3 2015-03-26 11:43:37,344 INFO      question_progress: Results 0% (Get Operating System contains "Windo
4 2015-03-26 11:43:42,359 INFO      question_progress: Results 50% (Get Operating System contains "Windo
5 2015-03-26 11:43:47,379 INFO      question_progress: Results 100% (Get Operating System contains "Win
6
7 Type of response:  <type 'dict'>

```

```
8
9 Pretty print of response:
10 {'question_object': <taniumpy.object_types.question.Question object at 0x107595990>,
11  'question_results': <taniumpy.object_types.result_set.ResultSet object at 0x107623190>}
12
13 Equivalent Question if it were to be asked in the Tanium Console:
14 Get Operating System contains "Windows" from all machines
15
16 CSV Results of response:
17 Operating System
18 [no results]
19 Windows Server 2008 R2 Standard
```

### Ask manual human question sensor with parameters and filter and options

Ask a manual question using human strings by referencing the name of a single sensor that takes parameters, but supplying only two of the four parameters that are used by the sensor.

Also supply a sensor filter that limits the column data that is shown to values that match the regex `‘.*Shared.*’`, and a sensor filter option that re-fetches any cached data that is older than 3600 seconds.

No question filters or question options supplied.

### Example Python Code

```
1 import os
2 import sys
3 sys.dont_write_bytecode = True
4
5 # Determine our script name, script dir
6 my_file = os.path.abspath(sys.argv[0])
7 my_dir = os.path.dirname(my_file)
8
9 # determine the pytan lib dir and add it to the path
10 parent_dir = os.path.dirname(my_dir)
11 pytan_root_dir = os.path.dirname(parent_dir)
12 lib_dir = os.path.join(pytan_root_dir, 'lib')
13 path_adds = [lib_dir]
14
15 for aa in path_adds:
16     if aa not in sys.path:
17         sys.path.append(aa)
18
19
20 # connection info for Tanium Server
21 USERNAME = "Tanium User"
22 PASSWORD = "T@n!um"
23 HOST = "172.16.31.128"
24 PORT = "444"
25
26 # Logging conrols
27 LOGLEVEL = 2
28 DEBUGFORMAT = False
29
30 import tempfile
31
```



```

32 import pytan
33 handler = pytan.Handler(
34     username=USERNAME,
35     password=PASSWORD,
36     host=HOST,
37     port=PORT,
38     loglevel=LOGLEVEL,
39     debugformat=DEBUGFORMAT,
40 )
41
42 print handler
43
44 # setup the arguments for the handler method
45 kwargs = {}
46 kwargs["sensors"] = u'Folder Name Search with RegEx Match(dirname=Program Files,regex=Microsoft.*),
47 kwargs["qtype"] = u'manual_human'
48
49 # call the handler with the ask method, passing in kwargs for arguments
50 response = handler.ask(**kwargs)
51 import pprint, io
52
53 print ""
54 print "Type of response: ", type(response)
55
56 print ""
57 print "Pretty print of response:"
58 print pprint.pformat(response)
59
60 print ""
61 print "Equivalent Question if it were to be asked in the Tanium Console: "
62 print response['question_object'].query_text
63
64 # create an IO stream to store CSV results to
65 out = io.BytesIO()
66
67 # call the write_csv() method to convert response to CSV and store it in out
68 response['question_results'].write_csv(out, response['question_results'])
69
70 print ""
71 print "CSV Results of response: "
72 out = out.getvalue()
73 if len(out.splitlines()) > 15:
74     out = out.splitlines()[0:15]
75     out.append('..trimmed for brevity..')
76     out = '\n'.join(out)
77 print out

```

### Output from Python Code

```

1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3279
2 2015-03-26 11:43:47,532 INFO question_progress: Results 0% (Get Folder Name Search with RegEx Ma
3 2015-03-26 11:43:52,555 INFO question_progress: Results 0% (Get Folder Name Search with RegEx Ma
4 2015-03-26 11:43:57,573 INFO question_progress: Results 0% (Get Folder Name Search with RegEx Ma
5 2015-03-26 11:44:02,596 INFO question_progress: Results 0% (Get Folder Name Search with RegEx Ma
6 2015-03-26 11:44:07,620 INFO question_progress: Results 50% (Get Folder Name Search with RegEx M
7 2015-03-26 11:44:12,644 INFO question_progress: Results 50% (Get Folder Name Search with RegEx M
8 2015-03-26 11:44:17,662 INFO question_progress: Results 100% (Get Folder Name Search with RegEx

```

```

9
10 Type of response: <type 'dict'>
11
12 Pretty print of response:
13 {'question_object': <taniumpy.object_types.question.Question object at 0x1076129d0>,
14  'question_results': <taniumpy.object_types.result_set.ResultSet object at 0x107856950>}
15
16 Equivalent Question if it were to be asked in the Tanium Console:
17 Get Folder Name Search with RegEx Match[No, Program Files, No, , Microsoft.*] contains "Shared" from
18
19 CSV Results of response:
20 "Folder Name Search with RegEx Match[No, Program Files, No, , Microsoft.*]"
21 [no results]
22 C:\Program Files\Common Files\Microsoft Shared\VS7Debug
23 C:\Program Files\Common Files\Microsoft Shared\ink\ar-SA
24 C:\Program Files\Common Files\Microsoft Shared\ink\ru-RU
25 C:\Program Files\Common Files\Microsoft Shared\ink\fsdefinitions\keypad
26 C:\Program Files\Common Files\Microsoft Shared\ink
27 C:\Program Files\Common Files\Microsoft Shared\ink\sv-SE
28 C:\Program Files\Microsoft SQL Server\110\Setup Bootstrap\Update Cache\KB2977326\GDR\1033_enu_lp\x64
29 C:\Program Files\Common Files\Microsoft Shared\ink\uk-UA
30 C:\Program Files\Common Files\Microsoft Shared\ink\sl-SI
31 C:\Program Files\Common Files\Microsoft Shared\ink\hu-HU
32 C:\Program Files\Common Files\Microsoft Shared\ink\zh-TW
33 C:\Program Files\Common Files\Microsoft Shared\ink\zh-CN
34 C:\Program Files\Common Files\Microsoft Shared\ink\fi-FI
35 ..trimmed for brevity..

```

### Ask manual human question sensor with filter and 2 options

Ask a manual question using human strings by referencing the name of a single sensor.

Also supply a sensor filter that limits the column data that is shown to values that contain Windows (which is short hand for regex match against .\*Windows.\*).

Also supply filter options that re-fetches any cached data that is older than 3600 seconds and treats the values as type string.

No question filters or question options supplied.

### Example Python Code

```

1 import os
2 import sys
3 sys.dont_write_bytecode = True
4
5 # Determine our script name, script dir
6 my_file = os.path.abspath(sys.argv[0])
7 my_dir = os.path.dirname(my_file)
8
9 # determine the pytan lib dir and add it to the path
10 parent_dir = os.path.dirname(my_dir)
11 pytan_root_dir = os.path.dirname(parent_dir)
12 lib_dir = os.path.join(pytan_root_dir, 'lib')
13 path_adds = [lib_dir]
14

```

```

15 for aa in path_adds:
16     if aa not in sys.path:
17         sys.path.append(aa)
18
19
20 # connection info for Tanium Server
21 USERNAME = "Tanium User"
22 PASSWORD = "T@n!um"
23 HOST = "172.16.31.128"
24 PORT = "444"
25
26 # Logging conrols
27 LOGLEVEL = 2
28 DEBUGFORMAT = False
29
30 import tempfile
31
32 import pytan
33 handler = pytan.Handler(
34     username=USERNAME,
35     password=PASSWORD,
36     host=HOST,
37     port=PORT,
38     loglevel=LOGLEVEL,
39     debugformat=DEBUGFORMAT,
40 )
41
42 print handler
43
44 # setup the arguments for the handler method
45 kwargs = {}
46 kwargs["sensors"] = u'Operating System, that contains:Windows, opt:max_data_age:3600, opt:value_type
47 kwargs["qtype"] = u'manual_human'
48
49 # call the handler with the ask method, passing in kwargs for arguments
50 response = handler.ask(**kwargs)
51 import pprint, io
52
53 print ""
54 print "Type of response: ", type(response)
55
56 print ""
57 print "Pretty print of response:"
58 print pprint.pformat(response)
59
60 print ""
61 print "Equivalent Question if it were to be asked in the Tanium Console: "
62 print response['question_object'].query_text
63
64 # create an IO stream to store CSV results to
65 out = io.BytesIO()
66
67 # call the write_csv() method to convert response to CSV and store it in out
68 response['question_results'].write_csv(out, response['question_results'])
69
70 print ""
71 print "CSV Results of response: "
72 out = out.getvalue()

```

```
73 if len(out.splitlines()) > 15:
74     out = out.splitlines()[0:15]
75     out.append('..trimmed for brevity..')
76     out = '\n'.join(out)
77 print out
```

### Output from Python Code

```
1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3279
2 2015-03-26 11:44:17,799 INFO      question_progress: Results 0% (Get Operating System contains "Windows")
3 2015-03-26 11:44:22,824 INFO      question_progress: Results 0% (Get Operating System contains "Windows")
4 2015-03-26 11:44:27,844 INFO      question_progress: Results 100% (Get Operating System contains "Windows")
5
6 Type of response: <type 'dict'>
7
8 Pretty print of response:
9 {'question_object': <taniumpy.object_types.question.Question object at 0x10760f9d0>,
10  'question_results': <taniumpy.object_types.result_set.ResultSet object at 0x107856050>}
11
12 Equivalent Question if it were to be asked in the Tanium Console:
13 Get Operating System contains "Windows" from all machines
14
15 CSV Results of response:
16 Operating System
17 [no results]
18 Windows Server 2008 R2 Standard
```

### Ask manual human question sensor with filter

Ask a manual question using human strings by referencing the name of a single sensor.

Also supply a sensor filter that limits the column data that is shown to values that contain Windows (which is short hand for regex match against `.*Windows.*`).

No sensor parameters, sensor filter options, question filters or question options supplied.

### Example Python Code

```
1 import os
2 import sys
3 sys.dont_write_bytecode = True
4
5 # Determine our script name, script dir
6 my_file = os.path.abspath(sys.argv[0])
7 my_dir = os.path.dirname(my_file)
8
9 # determine the pytan lib dir and add it to the path
10 parent_dir = os.path.dirname(my_dir)
11 pytan_root_dir = os.path.dirname(parent_dir)
12 lib_dir = os.path.join(pytan_root_dir, 'lib')
13 path_adds = [lib_dir]
14
15 for aa in path_adds:
16     if aa not in sys.path:
17         sys.path.append(aa)
```

```

18
19
20 # connection info for Tanium Server
21 USERNAME = "Tanium User"
22 PASSWORD = "T@n!um"
23 HOST = "172.16.31.128"
24 PORT = "444"
25
26 # Logging conrols
27 LOGLEVEL = 2
28 DEBUGFORMAT = False
29
30 import tempfile
31
32 import pytan
33 handler = pytan.Handler(
34     username=USERNAME,
35     password=PASSWORD,
36     host=HOST,
37     port=PORT,
38     loglevel=LOGLEVEL,
39     debugformat=DEBUGFORMAT,
40 )
41
42 print handler
43
44 # setup the arguments for the handler method
45 kwargs = {}
46 kwargs["sensors"] = u'Operating System, that contains:Windows'
47 kwargs["qtype"] = u'manual_human'
48
49 # call the handler with the ask method, passing in kwargs for arguments
50 response = handler.ask(**kwargs)
51 import pprint, io
52
53 print ""
54 print "Type of response: ", type(response)
55
56 print ""
57 print "Pretty print of response:"
58 print pprint.pformat(response)
59
60 print ""
61 print "Equivalent Question if it were to be asked in the Tanium Console: "
62 print response['question_object'].query_text
63
64 # create an IO stream to store CSV results to
65 out = io.BytesIO()
66
67 # call the write_csv() method to convert response to CSV and store it in out
68 response['question_results'].write_csv(out, response['question_results'])
69
70 print ""
71 print "CSV Results of response: "
72 out = out.getvalue()
73 if len(out.splitlines()) > 15:
74     out = out.splitlines()[0:15]
75     out.append('..trimmed for brevity..')

```

```
76     out = '\n'.join(out)
77     print out
```

### Output from Python Code

```
1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3279
2 2015-03-26 11:44:27,978 INFO      question_progress: Results 0% (Get Operating System contains "Windows")
3 2015-03-26 11:44:32,998 INFO      question_progress: Results 0% (Get Operating System contains "Windows")
4 2015-03-26 11:44:38,016 INFO      question_progress: Results 100% (Get Operating System contains "Windows")
5
6 Type of response: <type 'dict'>
7
8 Pretty print of response:
9 {'question_object': <taniumpy.object_types.question.Question object at 0x10761df10>,
10  'question_results': <taniumpy.object_types.result_set.ResultSet object at 0x10759ac90>}
11
12 Equivalent Question if it were to be asked in the Tanium Console:
13 Get Operating System contains "Windows" from all machines
14
15 CSV Results of response:
16 Operating System
17 [no results]
18 Windows Server 2008 R2 Standard
```

### Ask manual human question complex query1

Ask a manual question using human strings by referencing the name of a two sensors sensor.

Supply 3 parameters for the second sensor, one of which is not a valid parameter (and will be ignored).

Supply one option to the second sensor.

Supply two question filters that limit the rows returned in the result to computers that match the sensor Operating System that contains Windows and does not contain Windows.

Supply two question options that 'or' the two question filters and ignore the case of any values while matching the question filters.

### Example Python Code

```
1 import os
2 import sys
3 sys.dont_write_bytecode = True
4
5 # Determine our script name, script dir
6 my_file = os.path.abspath(sys.argv[0])
7 my_dir = os.path.dirname(my_file)
8
9 # determine the pytan lib dir and add it to the path
10 parent_dir = os.path.dirname(my_dir)
11 pytan_root_dir = os.path.dirname(parent_dir)
12 lib_dir = os.path.join(pytan_root_dir, 'lib')
13 path_adds = [lib_dir]
14
15 for aa in path_adds:
```

```

16     if aa not in sys.path:
17         sys.path.append(aa)
18
19
20     # connection info for Tanium Server
21     USERNAME = "Tanium User"
22     PASSWORD = "T@n!um"
23     HOST = "172.16.31.128"
24     PORT = "444"
25
26     # Logging conrols
27     LOGLEVEL = 2
28     DEBUGFORMAT = False
29
30     import tempfile
31
32     import pytan
33     handler = pytan.Handler(
34         username=USERNAME,
35         password=PASSWORD,
36         host=HOST,
37         port=PORT,
38         loglevel=LOGLEVEL,
39         debugformat=DEBUGFORMAT,
40     )
41
42     print handler
43
44     # setup the arguments for the handler method
45     kwargs = {}
46     kwargs["question_filters"] = [u'Operating System, that contains:Windows',
47         u'Operating System, that does not contain:Windows']
48     kwargs["sensors"] = [u'Computer Name',
49         u'Folder Name Search with RegEx Match{dirname=Program Files,regex=Microsoft.*, invalidparam=test}',
50     kwargs["question_options"] = [u'ignore_case', u'or']
51     kwargs["qtype"] = u'manual_human'
52
53     # call the handler with the ask method, passing in kwargs for arguments
54     response = handler.ask(**kwargs)
55     import pprint, io
56
57     print ""
58     print "Type of response: ", type(response)
59
60     print ""
61     print "Pretty print of response:"
62     print pprint.pformat(response)
63
64     print ""
65     print "Equivalent Question if it were to be asked in the Tanium Console: "
66     print response['question_object'].query_text
67
68     # create an IO stream to store CSV results to
69     out = io.BytesIO()
70
71     # call the write_csv() method to convert response to CSV and store it in out
72     response['question_results'].write_csv(out, response['question_results'])
73

```

```

74 print ""
75 print "CSV Results of response: "
76 out = out.getvalue()
77 if len(out.splitlines()) > 15:
78     out = out.splitlines()[0:15]
79     out.append('..trimmed for brevity..')
80     out = '\n'.join(out)
81 print out

```

## Output from Python Code

```

1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3279
2 2015-03-26 11:44:38,245 INFO question_progress: Results 0% (Get Computer Name and Folder Name Se
3 2015-03-26 11:44:43,272 INFO question_progress: Results 0% (Get Computer Name and Folder Name Se
4 2015-03-26 11:44:48,308 INFO question_progress: Results 0% (Get Computer Name and Folder Name Se
5 2015-03-26 11:44:53,341 INFO question_progress: Results 0% (Get Computer Name and Folder Name Se
6 2015-03-26 11:44:58,368 INFO question_progress: Results 0% (Get Computer Name and Folder Name Se
7 2015-03-26 11:45:03,396 INFO question_progress: Results 0% (Get Computer Name and Folder Name Se
8 2015-03-26 11:45:08,419 INFO question_progress: Results 100% (Get Computer Name and Folder Name
9
10 Type of response: <type 'dict'>
11
12 Pretty print of response:
13 {'question_object': <taniumpy.object_types.question.Question object at 0x107619110>,
14  'question_results': <taniumpy.object_types.result_set.ResultSet object at 0x1078087d0>}
15
16 Equivalent Question if it were to be asked in the Tanium Console:
17 Get Computer Name and Folder Name Search with RegEx Match[test, No, Program Files, No, , Microsoft.*]
18
19 CSV Results of response:
20 Computer Name,"Folder Name Search with RegEx Match[test, No, Program Files, No, , Microsoft.*]"
21 Casus-Belli.local,[no results]
22 jtanium1.localdomain,"C:\Program Files\Common Files\Microsoft Shared\VS7Debug
23 C:\Program Files\Common Files\Microsoft Shared\ink\ar-SA
24 C:\Program Files\Common Files\Microsoft Shared\ink\ru-RU
25 C:\Program Files\Common Files\Microsoft Shared\ink\fsdefinitions\keypad
26 C:\Program Files\Common Files\Microsoft Shared\ink
27 C:\Program Files\Common Files\Microsoft Shared\ink\sv-SE
28 C:\Program Files\Microsoft SQL Server\110\Setup Bootstrap\Update Cache\KB2977326\GDR\1033_enu_lp\x64
29 C:\Program Files\Common Files\Microsoft Shared\ink\uk-UA
30 C:\Program Files\Common Files\Microsoft Shared\ink\sl-SI
31 C:\Program Files\Common Files\Microsoft Shared\ink\hu-HU
32 C:\Program Files\Common Files\Microsoft Shared\ink\zh-TW
33 C:\Program Files\Common Files\Microsoft Shared\ink\zh-CN
34 C:\Program Files\Common Files\Microsoft Shared\ink\fi-FI
35 ..trimmed for brevity..

```

## Ask manual human question complex query2

This is another complex query that gets the Computer Name and Last Logged in User and Installed Applications that contains Google Search or Google Chrome and limits the rows that are displayed to computers that contain the Installed Applications of Google Search AND Google Chrome

## Example Python Code



```

1  import os
2  import sys
3  sys.dont_write_bytecode = True
4
5  # Determine our script name, script dir
6  my_file = os.path.abspath(sys.argv[0])
7  my_dir = os.path.dirname(my_file)
8
9  # determine the pytan lib dir and add it to the path
10 parent_dir = os.path.dirname(my_dir)
11 pytan_root_dir = os.path.dirname(parent_dir)
12 lib_dir = os.path.join(pytan_root_dir, 'lib')
13 path_adds = [lib_dir]
14
15 for aa in path_adds:
16     if aa not in sys.path:
17         sys.path.append(aa)
18
19
20 # connection info for Tanium Server
21 USERNAME = "Tanium User"
22 PASSWORD = "T@n!um"
23 HOST = "172.16.31.128"
24 PORT = "444"
25
26 # Logging conrols
27 LOGLEVEL = 2
28 DEBUGFORMAT = False
29
30 import tempfile
31
32 import pytan
33 handler = pytan.Handler(
34     username=USERNAME,
35     password=PASSWORD,
36     host=HOST,
37     port=PORT,
38     loglevel=LOGLEVEL,
39     debugformat=DEBUGFORMAT,
40 )
41
42 print handler
43
44 # setup the arguments for the handler method
45 kwargs = {}
46 kwargs["question_filters"] = [u'Installed Applications, that contains:Google Search',
47     u'Installed Applications, that contains:Google Chrome']
48 kwargs["sensors"] = [u'Computer Name',
49     u'Last Logged In User',
50     u'Installed Applications, that contains:Google Search',
51     u'Installed Applications, that contains:Google Chrome']
52 kwargs["question_options"] = [u'ignore_case', u'and']
53 kwargs["qtype"] = u'manual_human'
54
55 # call the handler with the ask method, passing in kwargs for arguments
56 response = handler.ask(**kwargs)
57 import pprint, io
58

```

```
59 print ""
60 print "Type of response: ", type(response)
61
62 print ""
63 print "Pretty print of response:"
64 print pprint.pformat(response)
65
66 print ""
67 print "Equivalent Question if it were to be asked in the Tanium Console: "
68 print response['question_object'].query_text
69
70 # create an IO stream to store CSV results to
71 out = io.BytesIO()
72
73 # call the write_csv() method to convert response to CSV and store it in out
74 response['question_results'].write_csv(out, response['question_results'])
75
76 print ""
77 print "CSV Results of response: "
78 out = out.getvalue()
79 if len(out.splitlines()) > 15:
80     out = out.splitlines()[0:15]
81     out.append('..trimmed for brevity..')
82     out = '\n'.join(out)
83 print out
```

### Output from Python Code

```
1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3279
2 2015-03-26 11:45:08,635 INFO      question_progress: Results 0% (Get Computer Name and Last Logged In
3 2015-03-26 11:45:13,672 INFO      question_progress: Results 0% (Get Computer Name and Last Logged In
4 2015-03-26 11:45:18,723 INFO      question_progress: Results 100% (Get Computer Name and Last Logged
5
6 Type of response:  <type 'dict'>
7
8 Pretty print of response:
9 {'question_object': <taniumpy.object_types.question.Question object at 0x10767b610>,
10  'question_results': <taniumpy.object_types.result_set.ResultSet object at 0x1076184d0>}
11
12 Equivalent Question if it were to be asked in the Tanium Console:
13 Get Computer Name and Last Logged In User and Installed Applications contains "Google Search" and In
14
15 CSV Results of response:
16 Computer Name,Last Logged In User,Name,Name,Silent Uninstall String,Silent Uninstall String,Uninstal
17 Casus-Belli.local,N/A on Mac,Google Search,Google Search,nothing,nothing,Not Uninstallable,Not Unins
```

### Ask manual question sensor complex

This provides an example for asking a manual question without using human strings.

It uses the Computer Name and Folder Name Search with RegEx Match sensors.

The second sensor has a single parameter, `dirname`, with a value of 'Program Files'.

The second sensor also has 3 sensor filter options that set the max data age to 3600 seconds, does NOT ignore case, and treats all values as string.

There is also a question filter supplied that limits the rows that are displayed to computers that match an Operating System that contains Windows, and has 3 question filter options supplied that set the max data age to 3600 seconds, does NOT ignore case, and uses 'and' to join all question filters.

### Example Python Code

```

1  import os
2  import sys
3  sys.dont_write_bytecode = True
4
5  # Determine our script name, script dir
6  my_file = os.path.abspath(sys.argv[0])
7  my_dir = os.path.dirname(my_file)
8
9  # determine the pytan lib dir and add it to the path
10 parent_dir = os.path.dirname(my_dir)
11 pytan_root_dir = os.path.dirname(parent_dir)
12 lib_dir = os.path.join(pytan_root_dir, 'lib')
13 path_adds = [lib_dir]
14
15 for aa in path_adds:
16     if aa not in sys.path:
17         sys.path.append(aa)
18
19
20 # connection info for Tanium Server
21 USERNAME = "Tanium User"
22 PASSWORD = "T@n!um"
23 HOST = "172.16.31.128"
24 PORT = "444"
25
26 # Logging conrols
27 LOGLEVEL = 2
28 DEBUGFORMAT = False
29
30 import tempfile
31
32 import pytan
33 handler = pytan.Handler(
34     username=USERNAME,
35     password=PASSWORD,
36     host=HOST,
37     port=PORT,
38     loglevel=LOGLEVEL,
39     debugformat=DEBUGFORMAT,
40 )
41
42 print handler
43
44 # setup the arguments for the handler method
45 kwargs = {}
46 kwargs["question_filter_defs"] = [{u'filter': {u'not_flag': 0,
47     u'operator': u'RegexMatch',
48     u'value': u'.*Windows.*'},
49     u'name': u'Operating System'}]
50 kwargs["sensor_defs"] = [u'Computer Name',
51     {u'filter': {u'not_flag': 0,
52     u'operator': u'RegexMatch',

```

```

53         u'value': u'.*Shared.*'},
54     u'name': u'Folder Name Search with RegEx Match',
55     u'options': {u'ignore_case_flag': 0,
56                 u'max_age_seconds': 3600,
57                 u'value_type': u'string'},
58     u'params': {u'dirname': u'Program Files'}}]
59 kwargs["question_option_defs"] = {u'and_flag': 0, u'ignore_case_flag': 0, u'max_age_seconds': 3600}
60 kwargs["qtype"] = u'manual'
61
62 # call the handler with the ask method, passing in kwargs for arguments
63 response = handler.ask(**kwargs)
64 import pprint, io
65
66 print ""
67 print "Type of response: ", type(response)
68
69 print ""
70 print "Pretty print of response:"
71 print pprint.pformat(response)
72
73 print ""
74 print "Equivalent Question if it were to be asked in the Tanium Console: "
75 print response['question_object'].query_text
76
77 # create an IO stream to store CSV results to
78 out = io.BytesIO()
79
80 # call the write_csv() method to convert response to CSV and store it in out
81 response['question_results'].write_csv(out, response['question_results'])
82
83 print ""
84 print "CSV Results of response: "
85 out = out.getvalue()
86 if len(out.splitlines()) > 15:
87     out = out.splitlines()[0:15]
88     out.append('..trimmed for brevity..')
89     out = '\n'.join(out)
90 print out

```

### Output from Python Code

```

1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3279
2 2015-03-26 11:45:18,920 INFO      question_progress: Results 0% (Get Computer Name and Folder Name Se
3 2015-03-26 11:45:23,943 INFO      question_progress: Results 0% (Get Computer Name and Folder Name Se
4 2015-03-26 11:45:28,969 INFO      question_progress: Results 50% (Get Computer Name and Folder Name S
5 2015-03-26 11:45:33,994 INFO      question_progress: Results 50% (Get Computer Name and Folder Name S
6 2015-03-26 11:45:39,017 INFO      question_progress: Results 100% (Get Computer Name and Folder Name
7
8 Type of response:  <type 'dict'>
9
10 Pretty print of response:
11 {'question_object': <taniumpy.object_types.question.Question object at 0x10757d750>,
12  'question_results': <taniumpy.object_types.result_set.ResultSet object at 0x107805e50>}
13
14 Equivalent Question if it were to be asked in the Tanium Console:
15 Get Computer Name and Folder Name Search with RegEx Match[No, Program Files, No, ] contains "Shared"
16

```

```

17 CSV Results of response:
18 Computer Name,"Folder Name Search with RegEx Match[No, Program Files, No, ]"
19 jtanium1.localdomain,"C:\Program Files\Common Files\Microsoft Shared\VS7Debug
20 C:\Program Files\Common Files\Microsoft Shared\ink\ar-SA
21 C:\Program Files\Common Files\Microsoft Shared\ink\ru-RU
22 C:\Program Files\Common Files\Microsoft Shared\ink\fsdefinitions\keypad
23 C:\Program Files\Common Files\Microsoft Shared\ink
24 C:\Program Files\Common Files\Microsoft Shared\ink\sv-SE
25 C:\Program Files\Common Files\Microsoft Shared\ink\uk-UA
26 C:\Program Files\Common Files\Microsoft Shared\ink\sl-SI
27 C:\Program Files\Common Files\Microsoft Shared\ink\hu-HU
28 C:\Program Files\Common Files\Microsoft Shared\ink\zh-TW
29 C:\Program Files\Common Files\Microsoft Shared\ink\zh-CN
30 C:\Program Files\Common Files\Microsoft Shared\ink\fi-FI
31 C:\Program Files\Common Files\Microsoft Shared
32 C:\Program Files\Common Files\Microsoft Shared\ink\da-DK
33 ..trimmed for brevity..

```

## pytan API Invalid Question Examples

### Invalid ask\_manual\_human question filter help

Have ask\_manual\_human() return the help for filters

### Example Python Code

```

1  import os
2  import sys
3  sys.dont_write_bytecode = True
4
5  # Determine our script name, script dir
6  my_file = os.path.abspath(sys.argv[0])
7  my_dir = os.path.dirname(my_file)
8
9  # determine the pytan lib dir and add it to the path
10 parent_dir = os.path.dirname(my_dir)
11 pytan_root_dir = os.path.dirname(parent_dir)
12 lib_dir = os.path.join(pytan_root_dir, 'lib')
13 path_adds = [lib_dir]
14
15 for aa in path_adds:
16     if aa not in sys.path:
17         sys.path.append(aa)
18
19
20 # connection info for Tanium Server
21 USERNAME = "Tanium User"
22 PASSWORD = "T@n!um"
23 HOST = "172.16.31.128"
24 PORT = "444"
25
26 # Logging conrols
27 LOGLEVEL = 2
28 DEBUGFORMAT = False
29

```

```
30 import tempfile
31
32 import pytan
33 handler = pytan.Handler(
34     username=USERNAME,
35     password=PASSWORD,
36     host=HOST,
37     port=PORT,
38     loglevel=LOGLEVEL,
39     debugformat=DEBUGFORMAT,
40 )
41
42 print handler
43
44 # setup the arguments for the handler method
45 kwargs = {}
46 kwargs["filters_help"] = True
47 kwargs["qtype"] = u'manual_human'
48
49
50 # call the handler with the ask method, passing in kwargs for arguments
51 # this should throw an exception: pytan.utils.PytanHelp
52 import traceback
53 try:
54     handler.ask(**kwargs)
55 except Exception as e:
56     traceback.print_exc(file=sys.stdout)
```

### Output from Python Code

```
1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3279
2 Traceback (most recent call last):
3   File "<string>", line 55, in <module>
4   File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 128, in ask
5     result = getattr(self, q_obj_map['handler'])(**kwargs)
6   File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 379, in ask_manual_human
7     raise PytanHelp(utils.help_filters())
8 PytanHelp:
9 Filters Help
10 =====
11
12 Filters are used generously throughout pytan. When used as part of a
13 sensor string, they control what data is shown for the columns that
14 the sensor returns. When filters are used for whole question filters,
15 they control what rows will be returned. They are used by Groups to
16 define group membership, deploy actions to determine which machines
17 should have the action deployed to it, and more.
18
19 A filter string is a human string that describes, a sensor followed
20 by ', that FILTER:VALUE', where FILTER is a valid filter string,
21 and VALUE is the string that you want FILTER to match on.
22
23 Valid Filters
24 -----
25
26 '<'
27     Help: Filter for less than VALUE
```

```

28         Example: "Sensor1, that <:VALUE"
29
30     'less'
31         Help: Filter for less than VALUE
32         Example: "Sensor1, that less:VALUE"
33
34     'lt'
35         Help: Filter for less than VALUE
36         Example: "Sensor1, that lt:VALUE"
37
38     'less than'
39         Help: Filter for less than VALUE
40         Example: "Sensor1, that less than:VALUE"
41
42     '!'<'
43         Help: Filter for not less than VALUE
44         Example: "Sensor1, that !<:VALUE"
45
46     'notless'
47         Help: Filter for not less than VALUE
48         Example: "Sensor1, that notless:VALUE"
49
50     'not less'
51         Help: Filter for not less than VALUE
52         Example: "Sensor1, that not less:VALUE"
53
54     'not less than'
55         Help: Filter for not less than VALUE
56         Example: "Sensor1, that not less than:VALUE"
57
58     '<='
59         Help: Filter for less than or equal to VALUE
60         Example: "Sensor1, that <=:VALUE"
61
62     'less equal'
63         Help: Filter for less than or equal to VALUE
64         Example: "Sensor1, that less equal:VALUE"
65
66     'lessequal'
67         Help: Filter for less than or equal to VALUE
68         Example: "Sensor1, that lessequal:VALUE"
69
70     'le'
71         Help: Filter for less than or equal to VALUE
72         Example: "Sensor1, that le:VALUE"
73
74     '!'<='
75         Help: Filter for not less than or equal to VALUE
76         Example: "Sensor1, that !<=:VALUE"
77
78     'not less equal'
79         Help: Filter for not less than or equal to VALUE
80         Example: "Sensor1, that not less equal:VALUE"
81
82     'not lessequal'
83         Help: Filter for not less than or equal to VALUE
84         Example: "Sensor1, that not lessequal:VALUE"
85

```

```
'>'
    Help: Filter for greater than VALUE
    Example: "Sensor1, that >:VALUE"

'greater'
    Help: Filter for greater than VALUE
    Example: "Sensor1, that greater:VALUE"

'gt'
    Help: Filter for greater than VALUE
    Example: "Sensor1, that gt:VALUE"

'greater than'
    Help: Filter for greater than VALUE
    Example: "Sensor1, that greater than:VALUE"

'!>'
    Help: Filter for not greater than VALUE
    Example: "Sensor1, that !>:VALUE"

'not greater'
    Help: Filter for not greater than VALUE
    Example: "Sensor1, that not greater:VALUE"

'notgreater'
    Help: Filter for not greater than VALUE
    Example: "Sensor1, that notgreater:VALUE"

'not greater than'
    Help: Filter for not greater than VALUE
    Example: "Sensor1, that not greater than:VALUE"

'>='
    Help: Filter for greater than or equal to VALUE
    Example: "Sensor1, that >=:VALUE"

'greater equal'
    Help: Filter for greater than or equal to VALUE
    Example: "Sensor1, that greater equal:VALUE"

'greaterequal'
    Help: Filter for greater than or equal to VALUE
    Example: "Sensor1, that greaterequal:VALUE"

'ge'
    Help: Filter for greater than or equal to VALUE
    Example: "Sensor1, that ge:VALUE"

'!>='
    Help: Filter for not greater than VALUE
    Example: "Sensor1, that !>=:VALUE"

'not greater equal'
    Help: Filter for not greater than VALUE
    Example: "Sensor1, that not greater equal:VALUE"

'notgreaterequal'
    Help: Filter for not greater than VALUE
```



```

144         Example: "Sensor1, that notgreaterequal:VALUE"
145
146     '='
147         Help: Filter for equals to VALUE
148         Example: "Sensor1, that =:VALUE"
149
150     'equal'
151         Help: Filter for equals to VALUE
152         Example: "Sensor1, that equal:VALUE"
153
154     'equals'
155         Help: Filter for equals to VALUE
156         Example: "Sensor1, that equals:VALUE"
157
158     'eq'
159         Help: Filter for equals to VALUE
160         Example: "Sensor1, that eq:VALUE"
161
162     '!='
163         Help: Filter for not equals to VALUE
164         Example: "Sensor1, that !=:VALUE"
165
166     'not equal'
167         Help: Filter for not equals to VALUE
168         Example: "Sensor1, that not equal:VALUE"
169
170     'notequal'
171         Help: Filter for not equals to VALUE
172         Example: "Sensor1, that notequal:VALUE"
173
174     'not equals'
175         Help: Filter for not equals to VALUE
176         Example: "Sensor1, that not equals:VALUE"
177
178     'notequals'
179         Help: Filter for not equals to VALUE
180         Example: "Sensor1, that notequals:VALUE"
181
182     'ne'
183         Help: Filter for not equals to VALUE
184         Example: "Sensor1, that ne:VALUE"
185
186     'contains'
187         Help: Filter for contains VALUE (adds .* before and after VALUE)
188         Example: "Sensor1, that contains:VALUE"
189
190     'does not contain'
191         Help: Filter for does not contain VALUE (adds .* before and after VALUE)
192         Example: "Sensor1, that does not contain:VALUE"
193
194     'doesnotcontain'
195         Help: Filter for does not contain VALUE (adds .* before and after VALUE)
196         Example: "Sensor1, that doesnotcontain:VALUE"
197
198     'not contains'
199         Help: Filter for does not contain VALUE (adds .* before and after VALUE)
200         Example: "Sensor1, that not contains:VALUE"
201

```

```
202 'notcontains'
203     Help: Filter for does not contain VALUE (adds .* before and after VALUE)
204     Example: "Sensor1, that notcontains:VALUE"
205
206 'starts with'
207     Help: Filter for starts with VALUE (adds .* after VALUE)
208     Example: "Sensor1, that starts with:VALUE"
209
210 'startswith'
211     Help: Filter for starts with VALUE (adds .* after VALUE)
212     Example: "Sensor1, that startswith:VALUE"
213
214 'does not start with'
215     Help: Filter for does not start with VALUE (adds .* after VALUE)
216     Example: "Sensor1, that does not start with:VALUE"
217
218 'doesnotstartswith'
219     Help: Filter for does not start with VALUE (adds .* after VALUE)
220     Example: "Sensor1, that doesnotstartswith:VALUE"
221
222 'not starts with'
223     Help: Filter for does not start with VALUE (adds .* after VALUE)
224     Example: "Sensor1, that not starts with:VALUE"
225
226 'notstartswith'
227     Help: Filter for does not start with VALUE (adds .* after VALUE)
228     Example: "Sensor1, that notstartswith:VALUE"
229
230 'ends with'
231     Help: Filter for ends with VALUE (adds .* before VALUE)
232     Example: "Sensor1, that ends with:VALUE"
233
234 'endswith'
235     Help: Filter for ends with VALUE (adds .* before VALUE)
236     Example: "Sensor1, that endswith:VALUE"
237
238 'does not end with'
239     Help: Filter for does bit end with VALUE (adds .* before VALUE)
240     Example: "Sensor1, that does not end with:VALUE"
241
242 'doesnotendwith'
243     Help: Filter for does bit end with VALUE (adds .* before VALUE)
244     Example: "Sensor1, that doesnotendwith:VALUE"
245
246 'not ends with'
247     Help: Filter for does bit end with VALUE (adds .* before VALUE)
248     Example: "Sensor1, that not ends with:VALUE"
249
250 'notstartswith'
251     Help: Filter for does bit end with VALUE (adds .* before VALUE)
252     Example: "Sensor1, that notstartswith:VALUE"
253
254 'is not'
255     Help: Filter for non regular expression match for VALUE
256     Example: "Sensor1, that is not:VALUE"
257
258 'not regex'
259     Help: Filter for non regular expression match for VALUE
```

```

260         Example: "Sensor1, that not regex:VALUE"
261
262     'notregex'
263         Help: Filter for non regular expression match for VALUE
264         Example: "Sensor1, that notregex:VALUE"
265
266     'not regex match'
267         Help: Filter for non regular expression match for VALUE
268         Example: "Sensor1, that not regex match:VALUE"
269
270     'notregexmatch'
271         Help: Filter for non regular expression match for VALUE
272         Example: "Sensor1, that notregexmatch:VALUE"
273
274     'nre'
275         Help: Filter for non regular expression match for VALUE
276         Example: "Sensor1, that nre:VALUE"
277
278     'is'
279         Help: Filter for regular expression match for VALUE
280         Example: "Sensor1, that is:VALUE"
281
282     'regex'
283         Help: Filter for regular expression match for VALUE
284         Example: "Sensor1, that regex:VALUE"
285
286     'regex match'
287         Help: Filter for regular expression match for VALUE
288         Example: "Sensor1, that regex match:VALUE"
289
290     'regexmatch'
291         Help: Filter for regular expression match for VALUE
292         Example: "Sensor1, that regexmatch:VALUE"
293
294     're'
295         Help: Filter for regular expression match for VALUE
296         Example: "Sensor1, that re:VALUE"

```

### Invalid ask manual human question option help

Have ask\_manual\_human() return the help for options

### Example Python Code

```

1  import os
2  import sys
3  sys.dont_write_bytecode = True
4
5  # Determine our script name, script dir
6  my_file = os.path.abspath(sys.argv[0])
7  my_dir = os.path.dirname(my_file)
8
9  # determine the pytan lib dir and add it to the path
10 parent_dir = os.path.dirname(my_dir)
11 pytan_root_dir = os.path.dirname(parent_dir)

```

```
12 lib_dir = os.path.join(pytan_root_dir, 'lib')
13 path_adds = [lib_dir]
14
15 for aa in path_adds:
16     if aa not in sys.path:
17         sys.path.append(aa)
18
19
20 # connection info for Tanium Server
21 USERNAME = "Tanium User"
22 PASSWORD = "T@n!um"
23 HOST = "172.16.31.128"
24 PORT = "444"
25
26 # Logging conrols
27 LOGLEVEL = 2
28 DEBUGFORMAT = False
29
30 import tempfile
31
32 import pytan
33 handler = pytan.Handler(
34     username=USERNAME,
35     password=PASSWORD,
36     host=HOST,
37     port=PORT,
38     loglevel=LOGLEVEL,
39     debugformat=DEBUGFORMAT,
40 )
41
42 print handler
43
44 # setup the arguments for the handler method
45 kwargs = {}
46 kwargs["options_help"] = True
47 kwargs["qtype"] = u'manual_human'
48
49
50 # call the handler with the ask method, passing in kwargs for arguments
51 # this should throw an exception: pytan.utils.PytanHelp
52 import traceback
53 try:
54     handler.ask(**kwargs)
55 except Exception as e:
56     traceback.print_exc(file=sys.stdout)
```

### Output from Python Code

```
1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3279
2 Traceback (most recent call last):
3   File "<string>", line 55, in <module>
4   File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 128, in ask
5     result = getattr(self, q_obj_map['handler'])(**kwargs)
6   File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 382, in ask_manual_human
7     raise PytanHelp(utils.help_options())
8 PytanHelp:
9 Options Help
```

```

10 =====
11
12 Options are used for controlling how filters act. When options are
13 used as part of a sensor string, they change how the filters
14 supplied as part of that sensor operate. When options are used for
15 whole question options, they change how all of the question filters
16 operate.
17
18 When options are supplied for a sensor string, they must be
19 supplied as ', opt:OPTION' or ', opt:OPTION:VALUE' for options
20 that require a value.
21
22 When options are supplied for question options, they must be
23 supplied as 'OPTION' or 'OPTION:VALUE' for options that require
24 a value.
25
26 Options can be used on 'filter' or 'group', where 'group' pertains
27 to group filters or question filters. All 'filter' options are also
28 applicable to 'group' for question options.
29
30 Valid Options
31 -----
32
33 'ignore_case'
34     Help: Make the filter do a case insensitive match
35     Usable on: filter
36     Example for sensor: "Sensor1, opt:ignore_case"
37     Example for question: "ignore_case"
38
39 'match_case'
40     Help: Make the filter do a case sensitive match
41     Usable on: filter
42     Example for sensor: "Sensor1, opt:match_case"
43     Example for question: "match_case"
44
45 'match_any_value'
46     Help: Make the filter match any value
47     Usable on: filter
48     Example for sensor: "Sensor1, opt:match_any_value"
49     Example for question: "match_any_value"
50
51 'match_all_values'
52     Help: Make the filter match all values
53     Usable on: filter
54     Example for sensor: "Sensor1, opt:match_all_values"
55     Example for question: "match_all_values"
56
57 'max_data_age'
58     Help: Re-fetch cached values older than N seconds
59     Usable on: filter
60     VALUE description and type: seconds, <type 'int'>
61     Example for sensor: "Sensor1, opt:max_data_age:seconds"
62     Example for question: "max_data_age:seconds"
63
64 'value_type'
65     Help: Make the filter consider the value type as VALUE_TYPE
66     Usable on: filter
67     VALUE description and type: value_type, <type 'str'>

```

```
68         Example for sensor: "Sensor1, opt:value_type:value_type"
69         Example for question: "value_type:value_type"
70
71     'and'
72     Help: Use 'and' for all of the filters supplied
73     Usable on: group
74     Example for sensor: "Sensor1, opt:and"
75     Example for question: "and"
76
77     'or'
78     Help: Use 'or' for all of the filters supplied
79     Usable on: group
80     Example for sensor: "Sensor1, opt:or"
81     Example for question: "or"
```

### Invalid ask manual human question sensor help

Have ask\_manual\_human() return the help for sensors

### Example Python Code

```
1  import os
2  import sys
3  sys.dont_write_bytecode = True
4
5  # Determine our script name, script dir
6  my_file = os.path.abspath(sys.argv[0])
7  my_dir = os.path.dirname(my_file)
8
9  # determine the pytan lib dir and add it to the path
10 parent_dir = os.path.dirname(my_dir)
11 pytan_root_dir = os.path.dirname(parent_dir)
12 lib_dir = os.path.join(pytan_root_dir, 'lib')
13 path_adds = [lib_dir]
14
15 for aa in path_adds:
16     if aa not in sys.path:
17         sys.path.append(aa)
18
19
20 # connection info for Tanium Server
21 USERNAME = "Tanium User"
22 PASSWORD = "T@n!um"
23 HOST = "172.16.31.128"
24 PORT = "444"
25
26 # Logging conrols
27 LOGLEVEL = 2
28 DEBUGFORMAT = False
29
30 import tempfile
31
32 import pytan
33 handler = pytan.Handler(
34     username=USERNAME,
```

```

35     password=PASSWORD,
36     host=HOST,
37     port=PORT,
38     loglevel=LOGLEVEL,
39     debugformat=DEBUGFORMAT,
40 )
41
42 print handler
43
44 # setup the arguments for the handler method
45 kwargs = {}
46 kwargs["qtype"] = u'manual_human'
47 kwargs["sensors_help"] = True
48
49
50 # call the handler with the ask method, passing in kwargs for arguments
51 # this should throw an exception: pytan.utils.PytanHelp
52 import traceback
53 try:
54     handler.ask(**kwargs)
55 except Exception as e:
56     traceback.print_exc(file=sys.stdout)

```

### Output from Python Code

```

1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3279
2 Traceback (most recent call last):
3   File "<string>", line 55, in <module>
4   File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 128, in ask
5     result = getattr(self, q_obj_map['handler'])(**kwargs)
6   File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 376, in ask_manual_human
7     raise PytanHelp(utils.help_sensors())
8 PytanHelp:
9 Sensors Help
10 =====
11
12 Supplying sensors controls what columns will be showed when you ask a
13 question.
14
15 A sensor string is a human string that describes, at a minimum, a sensor.
16 It can also optionally define a selector for the sensor, parameters for
17 the sensor, a filter for the sensor, and options for the filter for the
18 sensor. Sensors can be provided as a string or a list of strings.
19
20 Examples for basic sensors
21 -----
22
23 Supplying a single sensor:
24
25     'Computer Name'
26
27 Supplying two sensors in a list of strings:
28
29     ['Computer Name', 'IP Route Details']
30
31 Supplying multiple sensors with selectors (name is the default
32 selector if none is supplied):

```

```
33 [
34     'Computer Name',
35     'name:Computer Name',
36     'id:1',
37     'hash:123456789',
38 ]
39
```

#### 40 Sensor Parameters

41 -----

42  
43 Supplying parameters to a sensor can control the arguments that are  
44 supplied to a sensor, if that sensor takes any arguments.

45  
46 Sensor parameters must be surrounded with curly braces '{}',  
47 and must have a key and value specified that is separated by  
48 an equals '='. Multiple parameters must be separated by  
49 a comma ','. The key should match up to a valid parameter key  
50 for the sensor in question.  
51

52  
53 If a parameter is supplied and the sensor doesn't have a  
54 corresponding key name, it will be ignored. If the sensor has  
55 parameters and a parameter is NOT supplied then one of two  
56 paths will be taken:

- 57  
58 \* if the parameter does not require a default value, the  
59 parameter is left blank and not supplied.
- 60 \* if the parameter does require a value (pull downs, for  
61 example), a default value is derived (for pull downs,  
62 the first value available as a pull down entry is used).

#### 63 Examples for sensors with parameters

64 -----

65  
66 Supplying a single sensor with a single parameter 'dirname':

```
67  
68     'Sensor With Params{dirname=Program Files}'
69
```

70  
71 Supplying a single sensor with two parameters, 'param1' and  
72 'param2':

```
73  
74     'Sensor With Params{param1=value1,param2=value2}'
75
```

#### 76 Sensor Filters

77 -----

78  
79 Supplying a filter to a sensor controls what data will be shown in  
80 those columns (sensors) you've provided.

81  
82 Sensor filters can be supplied by adding ', that FILTER:VALUE',  
83 where FILTER is a valid filter string, and VALUE is the string  
84 that you want FILTER to match on.

85  
86 See filter help for a list of all possible FILTER strings.

87  
88 See options help for a list of options that can control how  
89 the filter works.  
90



## Examples for sensors with filters

-----

Supplying a sensor with a filter that limits the results to only show column data that matches the regular expression

'.\*Windows.\*' (Tanium does a case insensitive match by default):

```
'Computer Name, that contains:Windows'
```

Supplying a sensor with a filter that limits the results to only show column data that matches the regular expression

'Microsoft.\*':

```
'Computer Name, that starts with:Microsoft'
```

Supply a sensor with a filter that limits the results to only show column data that has a version greater or equal to '39.0.0.0'. Since this sensor uses Version as its default result type, there is no need to change the value type using filter options.

```
'Installed Application Version' \
'{Application Name=Google Chrome}, that =>:39.0.0.0'
```

## Sensor Options

-----

Supplying options to a sensor can change how the filter for that sensor works.

Sensor options can be supplied by adding ', opt:OPTION' or ', opt:OPTION:VALUE' for those options that require values, where OPTION is a valid option string, and VALUE is the appropriate value required by accordant OPTION.

See options help for a list of options that can control how the filter works.

## Examples for sensors with options

-----

Supplying a sensor with an option that forces tanium to re-fetch any cached column data that is older than 1 minute:

```
'Computer Name, opt:max_data_age:60'
```

Supplying a sensor with filter and an option that causes Tanium to match case for the filter value:

```
'Computer Name, that contains:Windows, opt:match_case'
```

Supplying a sensor with a filter and an option that causes Tanium to match all values supplied:

```
'Computer Name, that contains:Windows, opt:match_all_values'
```

Supplying a sensor with a filter and a set of options that causes Tanium to recognize the value type as String (which is

```
149 the default type for most sensors), re-fetch data older than
150 10 minutes, match any values, and match case:
151
152     'Computer Name', that contains:Windows, ' \
153     opt:value_type:string, opt:max_data_age:600, ' \
154     'opt:match_any_value, opt:match_case'
```

### Invalid ask manual human question filter

Ask a question using an invalid filter.

### Example Python Code

```
1  import os
2  import sys
3  sys.dont_write_bytecode = True
4
5  # Determine our script name, script dir
6  my_file = os.path.abspath(sys.argv[0])
7  my_dir = os.path.dirname(my_file)
8
9  # determine the pytan lib dir and add it to the path
10 parent_dir = os.path.dirname(my_dir)
11 pytan_root_dir = os.path.dirname(parent_dir)
12 lib_dir = os.path.join(pytan_root_dir, 'lib')
13 path_adds = [lib_dir]
14
15 for aa in path_adds:
16     if aa not in sys.path:
17         sys.path.append(aa)
18
19
20 # connection info for Tanium Server
21 USERNAME = "Tanium User"
22 PASSWORD = "T@n!um"
23 HOST = "172.16.31.128"
24 PORT = "444"
25
26 # Logging conrols
27 LOGLEVEL = 2
28 DEBUGFORMAT = False
29
30 import tempfile
31
32 import pytan
33 handler = pytan.Handler(
34     username=USERNAME,
35     password=PASSWORD,
36     host=HOST,
37     port=PORT,
38     loglevel=LOGLEVEL,
39     debugformat=DEBUGFORMAT,
40 )
41
42 print handler
```

```

43
44 # setup the arguments for the handler method
45 kwargs = {}
46 kwargs["sensors"] = u'Computer name, that does not meet:little'
47 kwargs["qtype"] = u'manual_human'
48
49
50 # call the handler with the ask method, passing in kwargs for arguments
51 # this should throw an exception: pytan.utils.HumanParserError
52 import traceback
53 try:
54     handler.ask(**kwargs)
55 except Exception as e:
56     traceback.print_exc(file=sys.stdout)

```

### Output from Python Code

```

1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3279
2 Traceback (most recent call last):
3   File "<string>", line 55, in <module>
4   File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 128, in ask
5     result = getattr(self, q_obj_map['handler'])(**kwargs)
6   File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 399, in ask_manual_human
7     sensor_defs = utils.dehumanize_sensors(sensors)
8   File "/Users/jolsen/gh/pytan/lib/pytan/utils.py", line 1303, in dehumanize_sensors
9     s, parsed_filter = extract_filter(s)
10  File "/Users/jolsen/gh/pytan/lib/pytan/utils.py", line 1671, in extract_filter
11    raise HumanParserError(err(split_filter[1]))
12 HumanParserError: Filter u' does not meet:little' is not a valid filter!

```

### Invalid ask manual question sensor

Ask a question using a sensor that does not exist

### Example Python Code

```

1 import os
2 import sys
3 sys.dont_write_bytecode = True
4
5 # Determine our script name, script dir
6 my_file = os.path.abspath(sys.argv[0])
7 my_dir = os.path.dirname(my_file)
8
9 # determine the pytan lib dir and add it to the path
10 parent_dir = os.path.dirname(my_dir)
11 pytan_root_dir = os.path.dirname(parent_dir)
12 lib_dir = os.path.join(pytan_root_dir, 'lib')
13 path_adds = [lib_dir]
14
15 for aa in path_adds:
16     if aa not in sys.path:
17         sys.path.append(aa)
18

```

```
19
20 # connection info for Tanium Server
21 USERNAME = "Tanium User"
22 PASSWORD = "T@n!um"
23 HOST = "172.16.31.128"
24 PORT = "444"
25
26 # Logging conrols
27 LOGLEVEL = 2
28 DEBUGFORMAT = False
29
30 import tempfile
31
32 import pytan
33 handler = pytan.Handler(
34     username=USERNAME,
35     password=PASSWORD,
36     host=HOST,
37     port=PORT,
38     loglevel=LOGLEVEL,
39     debugformat=DEBUGFORMAT,
40 )
41
42 print handler
43
44 # setup the arguments for the handler method
45 kwargs = {}
46 kwargs["sensor_defs"] = u'Dweedle Dee and Dum'
47 kwargs["qtype"] = u'manual'
48
49
50 # call the handler with the ask method, passing in kwargs for arguments
51 # this should throw an exception: pytan.utils.HandlerError
52 import traceback
53 try:
54     handler.ask(**kwargs)
55 except Exception as e:
56     traceback.print_exc(file=sys.stdout)
```

### Output from Python Code

```
1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3279
2 Traceback (most recent call last):
3   File "<string>", line 55, in <module>
4   File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 128, in ask
5     result = getattr(self, q_obj_map['handler'])(**kwargs)
6   File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 271, in ask_manual
7     sensor_defs = self._get_sensor_defs(sensor_defs)
8   File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 1817, in _get_sensor_defs
9     d['sensor_obj'] = self.get('sensor', **def_search)[0]
10  File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 1600, in get
11  return self._get_multi(obj_map, **kwargs)
12  File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 1762, in _get_multi
13  found = self._find(api_obj_multi, **kwargs)
14  File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 1727, in _find
15  raise HandlerError(err(search_str))
```

```
16 | HandlerError: No results found searching for Sensor, name: u'Dweedle Dee and Dum'!!
```

### Invalid ask manual human question paramater too many

Ask a question that supplies too many parameter blocks ({}).

### Example Python Code

```
1  import os
2  import sys
3  sys.dont_write_bytecode = True
4
5  # Determine our script name, script dir
6  my_file = os.path.abspath(sys.argv[0])
7  my_dir = os.path.dirname(my_file)
8
9  # determine the pytan lib dir and add it to the path
10 parent_dir = os.path.dirname(my_dir)
11 pytan_root_dir = os.path.dirname(parent_dir)
12 lib_dir = os.path.join(pytan_root_dir, 'lib')
13 path_adds = [lib_dir]
14
15 for aa in path_adds:
16     if aa not in sys.path:
17         sys.path.append(aa)
18
19
20 # connection info for Tanium Server
21 USERNAME = "Tanium User"
22 PASSWORD = "T@n!um"
23 HOST = "172.16.31.128"
24 PORT = "444"
25
26 # Logging conrols
27 LOGLEVEL = 2
28 DEBUGFORMAT = False
29
30 import tempfile
31
32 import pytan
33 handler = pytan.Handler(
34     username=USERNAME,
35     password=PASSWORD,
36     host=HOST,
37     port=PORT,
38     loglevel=LOGLEVEL,
39     debugformat=DEBUGFORMAT,
40 )
41
42 print handler
43
44 # setup the arguments for the handler method
45 kwargs = {}
46 kwargs["sensors"] = u'Folder Name Search with RegEx Match(dirname=Program Files,regex=.*){}'
47 kwargs["qtype"] = u'manual_human'
```

```
48
49
50 # call the handler with the ask method, passing in kwargs for arguments
51 # this should throw an exception: pytan.utils.HumanParserError
52 import traceback
53 try:
54     handler.ask(**kwargs)
55 except Exception as e:
56     traceback.print_exc(file=sys.stdout)
```

### Output from Python Code

```
1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3279
2 Traceback (most recent call last):
3   File "<string>", line 55, in <module>
4   File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 128, in ask
5     result = getattr(self, q_obj_map['handler'])(**kwargs)
6   File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 399, in ask_manual_human
7     sensor_defs = utils.dehumanize_sensors(sensors)
8   File "/Users/jolsen/gh/pytan/lib/pytan/utils.py", line 1301, in dehumanize_sensors
9     s, parsed_params = extract_params(s)
10  File "/Users/jolsen/gh/pytan/lib/pytan/utils.py", line 1471, in extract_params
11    raise HumanParserError(err(s))
12 HumanParserError: More than one parameter ({} ) passed in u'Folder Name Search with RegEx Match{dirna
```

### Invalid ask manual human question option

Ask a question using an invalid option.

### Example Python Code

```
1 import os
2 import sys
3 sys.dont_write_bytecode = True
4
5 # Determine our script name, script dir
6 my_file = os.path.abspath(sys.argv[0])
7 my_dir = os.path.dirname(my_file)
8
9 # determine the pytan lib dir and add it to the path
10 parent_dir = os.path.dirname(my_dir)
11 pytan_root_dir = os.path.dirname(parent_dir)
12 lib_dir = os.path.join(pytan_root_dir, 'lib')
13 path_adds = [lib_dir]
14
15 for aa in path_adds:
16     if aa not in sys.path:
17         sys.path.append(aa)
18
19
20 # connection info for Tanium Server
21 USERNAME = "Tanium User"
22 PASSWORD = "T@n!um"
23 HOST = "172.16.31.128"
```

```

24 PORT = "444"
25
26 # Logging conrols
27 LOGLEVEL = 2
28 DEBUGFORMAT = False
29
30 import tempfile
31
32 import pytan
33 handler = pytan.Handler(
34     username=USERNAME,
35     password=PASSWORD,
36     host=HOST,
37     port=PORT,
38     loglevel=LOGLEVEL,
39     debugformat=DEBUGFORMAT,
40 )
41
42 print handler
43
44 # setup the arguments for the handler method
45 kwargs = {}
46 kwargs["sensors"] = u'Operating system, opt:bad'
47 kwargs["qtype"] = u'manual_human'
48
49
50 # call the handler with the ask method, passing in kwargs for arguments
51 # this should throw an exception: pytan.utils.HumanParserError
52 import traceback
53 try:
54     handler.ask(**kwargs)
55 except Exception as e:
56     traceback.print_exc(file=sys.stdout)

```

### Output from Python Code

```

1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3279
2 Traceback (most recent call last):
3   File "<string>", line 55, in <module>
4   File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 128, in ask
5     result = getattr(self, q_obj_map['handler'])(**kwargs)
6   File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 399, in ask_manual_human
7     sensor_defs = utils.dehumanize_sensors(sensors)
8   File "/Users/jolsen/gh/pytan/lib/pytan/utils.py", line 1302, in dehumanize_sensors
9     s, parsed_options = extract_options(s)
10  File "/Users/jolsen/gh/pytan/lib/pytan/utils.py", line 1546, in extract_options
11    parsed_options = map_options(parsed_options, ['filter'])
12  File "/Users/jolsen/gh/pytan/lib/pytan/utils.py", line 1576, in map_options
13    raise HumanParserError(err(option))
14 HumanParserError: Option u'bad' is not a valid option!

```

### Invalid ask manual human question parameter split

Ask a question with parameters that are missing a splitter (=) to designate the key from value.

## Example Python Code

```
1  import os
2  import sys
3  sys.dont_write_bytecode = True
4
5  # Determine our script name, script dir
6  my_file = os.path.abspath(sys.argv[0])
7  my_dir = os.path.dirname(my_file)
8
9  # determine the pytan lib dir and add it to the path
10 parent_dir = os.path.dirname(my_dir)
11 pytan_root_dir = os.path.dirname(parent_dir)
12 lib_dir = os.path.join(pytan_root_dir, 'lib')
13 path_adds = [lib_dir]
14
15 for aa in path_adds:
16     if aa not in sys.path:
17         sys.path.append(aa)
18
19
20 # connection info for Tanium Server
21 USERNAME = "Tanium User"
22 PASSWORD = "T@n!um"
23 HOST = "172.16.31.128"
24 PORT = "444"
25
26 # Logging conrols
27 LOGLEVEL = 2
28 DEBUGFORMAT = False
29
30 import tempfile
31
32 import pytan
33 handler = pytan.Handler(
34     username=USERNAME,
35     password=PASSWORD,
36     host=HOST,
37     port=PORT,
38     loglevel=LOGLEVEL,
39     debugformat=DEBUGFORMAT,
40 )
41
42 print handler
43
44 # setup the arguments for the handler method
45 kwargs = {}
46 kwargs["sensors"] = u'Computer Name{Dweedle}'
47 kwargs["qtype"] = u'manual_human'
48
49
50 # call the handler with the ask method, passing in kwargs for arguments
51 # this should throw an exception: pytan.utils.HumanParserError
52 import traceback
53 try:
54     handler.ask(**kwargs)
55 except Exception as e:
56     traceback.print_exc(file=sys.stdout)
```



## Output from Python Code

```

1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3279
2 Traceback (most recent call last):
3   File "<string>", line 55, in <module>
4   File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 128, in ask
5     result = getattr(self, q_obj_map['handler'])(**kwargs)
6   File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 399, in ask_manual_human
7     sensor_defs = utils.dehumanize_sensors(sensors)
8   File "/Users/jolsen/gh/pytan/lib/pytan/utils.py", line 1301, in dehumanize_sensors
9     s, parsed_params = extract_params(s)
10  File "/Users/jolsen/gh/pytan/lib/pytan/utils.py", line 1489, in extract_params
11    raise HumanParserError(err(sp, constants.PARAM_KEY_SPLIT))
12 HumanParserError: Parameter Dweedle missing key/value seperator (=)

```

## pytan API Valid Get Object Examples

### Get action by id

Get an action by id

### Example Python Code

```

1 import os
2 import sys
3 sys.dont_write_bytecode = True
4
5 # Determine our script name, script dir
6 my_file = os.path.abspath(sys.argv[0])
7 my_dir = os.path.dirname(my_file)
8
9 # determine the pytan lib dir and add it to the path
10 parent_dir = os.path.dirname(my_dir)
11 pytan_root_dir = os.path.dirname(parent_dir)
12 lib_dir = os.path.join(pytan_root_dir, 'lib')
13 path_adds = [lib_dir]
14
15 for aa in path_adds:
16     if aa not in sys.path:
17         sys.path.append(aa)
18
19
20 # connection info for Tanium Server
21 USERNAME = "Tanium User"
22 PASSWORD = "T@n!um"
23 HOST = "172.16.31.128"
24 PORT = "444"
25
26 # Logging conrols
27 LOGLEVEL = 2
28 DEBUGFORMAT = False
29
30 import tempfile
31
32 import pytan
33 handler = pytan.Handler(

```

```
34     username=USERNAME,
35     password=PASSWORD,
36     host=HOST,
37     port=PORT,
38     loglevel=LOGLEVEL,
39     debugformat=DEBUGFORMAT,
40 )
41
42 print handler
43
44 # setup the arguments for the handler method
45 kwargs = {}
46 kwargs["objtype"] = u'action'
47 kwargs["id"] = 1
48
49 # call the handler with the get method, passing in kwargs for arguments
50 response = handler.get(**kwargs)
51
52 print ""
53 print "Type of response: ", type(response)
54
55 print ""
56 print "print of response:"
57 print response
58
59 print ""
60 print "length of response (number of objects returned): "
61 print len(response)
62
63 print ""
64 print "print the first object returned in JSON format:"
65 out = response.to_json(response[0])
66 if len(out.splitlines()) > 15:
67     out = out.splitlines()[0:15]
68     out.append('..trimmed for brevity..')
69     out = '\n'.join(out)
70
71 print out
```

### Output from Python Code

```
1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3279
2
3 Type of response: <class 'taniumpy.object_types.action_list.ActionList'>
4
5 print of response:
6 ActionList, len: 1
7
8 length of response (number of objects returned):
9 1
10
11 print the first object returned in JSON format:
12 {
13     "_type": "action",
14     "action_group": {
15         "_type": "group",
16         "id": 0,
```

```

17     "name": "Default"
18 },
19 "comment": "Scans for unmanaged assets on the network.",
20 "creation_time": "2015-03-03T19:05:56",
21 "distribute_seconds": 600,
22 "expiration_time": "2015-03-03T19:35:56",
23 "expire_seconds": 1800,
24 "history_saved_question": {
25     "_type": "saved_question",
26     "id": 173
27 ..trimmed for brevity..

```

## Get question by id

Get a question by id

## Example Python Code

```

1  import os
2  import sys
3  sys.dont_write_bytecode = True
4
5  # Determine our script name, script dir
6  my_file = os.path.abspath(sys.argv[0])
7  my_dir = os.path.dirname(my_file)
8
9  # determine the pytan lib dir and add it to the path
10 parent_dir = os.path.dirname(my_dir)
11 pytan_root_dir = os.path.dirname(parent_dir)
12 lib_dir = os.path.join(pytan_root_dir, 'lib')
13 path_adds = [lib_dir]
14
15 for aa in path_adds:
16     if aa not in sys.path:
17         sys.path.append(aa)
18
19
20 # connection info for Tanium Server
21 USERNAME = "Tanium User"
22 PASSWORD = "T@n!um"
23 HOST = "172.16.31.128"
24 PORT = "444"
25
26 # Logging conrols
27 LOGLEVEL = 2
28 DEBUGFORMAT = False
29
30 import tempfile
31
32 import pytan
33 handler = pytan.Handler(
34     username=USERNAME,
35     password=PASSWORD,
36     host=HOST,
37     port=PORT,

```

```
38     loglevel=LOGLEVEL,
39     debugformat=DEBUGFORMAT,
40 )
41
42 print handler
43
44 # setup the arguments for the handler method
45 kwargs = {}
46 kwargs["objtype"] = u'question'
47 kwargs["id"] = 1
48
49 # call the handler with the get method, passing in kwargs for arguments
50 response = handler.get(**kwargs)
51
52 print ""
53 print "Type of response: ", type(response)
54
55 print ""
56 print "print of response:"
57 print response
58
59 print ""
60 print "length of response (number of objects returned): "
61 print len(response)
62
63 print ""
64 print "print the first object returned in JSON format:"
65 out = response.to_json(response[0])
66 if len(out.splitlines()) > 15:
67     out = out.splitlines()[0:15]
68     out.append('..trimmed for brevity..')
69     out = '\n'.join(out)
70
71 print out
```

### Output from Python Code

```
1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3279
2
3 Type of response: <class 'taniumpy.object_types.question_list.QuestionList'>
4
5 print of response:
6 QuestionList, len: 1
7
8 length of response (number of objects returned):
9 1
10
11 print the first object returned in JSON format:
12 {
13     "_type": "question",
14     "action_tracking_flag": 0,
15     "context_group": {
16         "_type": "group",
17         "id": 0
18     },
19     "expiration": "2015-03-03T19:13:18",
20     "expire_seconds": 0,
```

```

21     "force_computer_id_flag": 0,
22     "hidden_flag": 0,
23     "id": 1,
24     "management_rights_group": {
25         "_type": "group",
26         "id": 0
27     }..trimmed for brevity..

```

### Get saved question by names

Get two saved questions by name

### Example Python Code

```

1  import os
2  import sys
3  sys.dont_write_bytecode = True
4
5  # Determine our script name, script dir
6  my_file = os.path.abspath(sys.argv[0])
7  my_dir = os.path.dirname(my_file)
8
9  # determine the pytan lib dir and add it to the path
10 parent_dir = os.path.dirname(my_dir)
11 pytan_root_dir = os.path.dirname(parent_dir)
12 lib_dir = os.path.join(pytan_root_dir, 'lib')
13 path_adds = [lib_dir]
14
15 for aa in path_adds:
16     if aa not in sys.path:
17         sys.path.append(aa)
18
19
20 # connection info for Tanium Server
21 USERNAME = "Tanium User"
22 PASSWORD = "T@n!um"
23 HOST = "172.16.31.128"
24 PORT = "444"
25
26 # Logging conrols
27 LOGLEVEL = 2
28 DEBUGFORMAT = False
29
30 import tempfile
31
32 import pytan
33 handler = pytan.Handler(
34     username=USERNAME,
35     password=PASSWORD,
36     host=HOST,
37     port=PORT,
38     loglevel=LOGLEVEL,
39     debugformat=DEBUGFORMAT,
40 )
41

```

```
42 print handler
43
44 # setup the arguments for the handler method
45 kwargs = {}
46 kwargs["objtype"] = u'saved_question'
47 kwargs["name"] = [u'Installed Applications', u'Computer Name']
48
49 # call the handler with the get method, passing in kwargs for arguments
50 response = handler.get(**kwargs)
51
52 print ""
53 print "Type of response: ", type(response)
54
55 print ""
56 print "print of response:"
57 print response
58
59 print ""
60 print "length of response (number of objects returned): "
61 print len(response)
62
63 print ""
64 print "print the first object returned in JSON format:"
65 out = response.to_json(response[0])
66 if len(out.splitlines()) > 15:
67     out = out.splitlines()[0:15]
68     out.append('..trimmed for brevity..')
69     out = '\n'.join(out)
70
71 print out
```

### Output from Python Code

```
1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3279
2
3 Type of response: <class 'taniumpy.object_types.saved_question_list.SavedQuestionList'>
4
5 print of response:
6 SavedQuestionList, len: 2
7
8 length of response (number of objects returned):
9 2
10
11 print the first object returned in JSON format:
12 {
13     "_type": "saved_question",
14     "action_tracking_flag": 0,
15     "archive_enabled_flag": 0,
16     "archive_owner": {
17         "_type": "user",
18         "id": 1,
19         "name": "Jim Olsen"
20     },
21     "expire_seconds": 600,
22     "hidden_flag": 0,
23     "id": 92,
24     "issue_seconds": 120,
```

```

25     "issue_seconds_never_flag": 0,
26     "keep_seconds": 3600,
27     ..trimmed for brevity..

```

### Get userrole by id

Get a user role by id.

### Example Python Code

```

1  import os
2  import sys
3  sys.dont_write_bytecode = True
4
5  # Determine our script name, script dir
6  my_file = os.path.abspath(sys.argv[0])
7  my_dir = os.path.dirname(my_file)
8
9  # determine the pytan lib dir and add it to the path
10 parent_dir = os.path.dirname(my_dir)
11 pytan_root_dir = os.path.dirname(parent_dir)
12 lib_dir = os.path.join(pytan_root_dir, 'lib')
13 path_adds = [lib_dir]
14
15 for aa in path_adds:
16     if aa not in sys.path:
17         sys.path.append(aa)
18
19
20 # connection info for Tanium Server
21 USERNAME = "Tanium User"
22 PASSWORD = "T@n!um"
23 HOST = "172.16.31.128"
24 PORT = "444"
25
26 # Logging conrols
27 LOGLEVEL = 2
28 DEBUGFORMAT = False
29
30 import tempfile
31
32 import pytan
33 handler = pytan.Handler(
34     username=USERNAME,
35     password=PASSWORD,
36     host=HOST,
37     port=PORT,
38     loglevel=LOGLEVEL,
39     debugformat=DEBUGFORMAT,
40 )
41
42 print handler
43
44 # setup the arguments for the handler method
45 kwargs = {}

```

```
46 kwargs["objtype"] = u'userrole'
47 kwargs["id"] = 1
48
49 # call the handler with the get method, passing in kwargs for arguments
50 response = handler.get(**kwargs)
51
52 print ""
53 print "Type of response: ", type(response)
54
55 print ""
56 print "print of response:"
57 print response
58
59 print ""
60 print "length of response (number of objects returned): "
61 print len(response)
62
63 print ""
64 print "print the first object returned in JSON format:"
65 out = response.to_json(response[0])
66 if len(out.splitlines()) > 15:
67     out = out.splitlines()[0:15]
68     out.append('..trimmed for brevity..')
69     out = '\n'.join(out)
70
71 print out
```

### Output from Python Code

```
1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3279
2
3 Type of response: <class 'taniumpy.object_types.user_role_list.UserRoleList'>
4
5 print of response:
6 UserRoleList, len: 1
7
8 length of response (number of objects returned):
9 1
10
11 print the first object returned in JSON format:
12 {
13     "_type": "role",
14     "description": "Administrators can perform all functions in the system, including creating other u
15     "id": 1,
16     "name": "Administrator",
17     "permissions": {
18         "_type": "permissions",
19         "permission": "admin"
20     }
21 }
```

### Get leader clients

Get all clients that are Leader status



## Example Python Code

```

1  import os
2  import sys
3  sys.dont_write_bytecode = True
4
5  # Determine our script name, script dir
6  my_file = os.path.abspath(sys.argv[0])
7  my_dir = os.path.dirname(my_file)
8
9  # determine the pytan lib dir and add it to the path
10 parent_dir = os.path.dirname(my_dir)
11 pytan_root_dir = os.path.dirname(parent_dir)
12 lib_dir = os.path.join(pytan_root_dir, 'lib')
13 path_adds = [lib_dir]
14
15 for aa in path_adds:
16     if aa not in sys.path:
17         sys.path.append(aa)
18
19
20 # connection info for Tanium Server
21 USERNAME = "Tanium User"
22 PASSWORD = "T@n!um"
23 HOST = "172.16.31.128"
24 PORT = "444"
25
26 # Logging conrols
27 LOGLEVEL = 2
28 DEBUGFORMAT = False
29
30 import tempfile
31
32 import pytan
33 handler = pytan.Handler(
34     username=USERNAME,
35     password=PASSWORD,
36     host=HOST,
37     port=PORT,
38     loglevel=LOGLEVEL,
39     debugformat=DEBUGFORMAT,
40 )
41
42 print handler
43
44 # setup the arguments for the handler method
45 kwargs = {}
46 kwargs["objtype"] = u'client'
47 kwargs["status"] = u'Leader'
48
49 # call the handler with the get method, passing in kwargs for arguments
50 response = handler.get(**kwargs)
51
52 print ""
53 print "Type of response: ", type(response)
54
55 print ""
56 print "print of response:"
57 print response

```

```
58
59 print ""
60 print "length of response (number of objects returned): "
61 print len(response)
62
63 print ""
64 print "print the first object returned in JSON format:"
65 out = response.to_json(response[0])
66 if len(out.splitlines()) > 15:
67     out = out.splitlines()[0:15]
68     out.append('..trimmed for brevity..')
69     out = '\n'.join(out)
70
71 print out
```

### Output from Python Code

```
1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3279
2
3 Type of response: <class 'taniumpy.object_types.system_status_list.SystemStatusList'>
4
5 print of response:
6 SystemStatusList, len: 4
7
8 length of response (number of objects returned):
9 4
10
11 print the first object returned in JSON format:
12 {
13     "_type": "client_status",
14     "cache_row_id": 1,
15     "computer_id": "103801052",
16     "full_version": "6.0.314.1195",
17     "host_name": "WIN-A12SC6N6T7Q",
18     "ipaddress_client": "172.16.31.157",
19     "ipaddress_server": "172.16.31.157",
20     "last_registration": "2015-03-11T09:30:02",
21     "port_number": 17472,
22     "protocol_version": 314,
23     "receive_state": "Previous Only",
24     "send_state": "Backward Only",
25     "status": "Leader"
26 }
```

### Get setting by name

Get a system setting by name

### Example Python Code

```
1 import os
2 import sys
3 sys.dont_write_bytecode = True
4
```

```

5  # Determine our script name, script dir
6  my_file = os.path.abspath(sys.argv[0])
7  my_dir = os.path.dirname(my_file)
8
9  # determine the pytan lib dir and add it to the path
10 parent_dir = os.path.dirname(my_dir)
11 pytan_root_dir = os.path.dirname(parent_dir)
12 lib_dir = os.path.join(pytan_root_dir, 'lib')
13 path_adds = [lib_dir]
14
15 for aa in path_adds:
16     if aa not in sys.path:
17         sys.path.append(aa)
18
19
20 # connection info for Tanium Server
21 USERNAME = "Tanium User"
22 PASSWORD = "T@n!um"
23 HOST = "172.16.31.128"
24 PORT = "444"
25
26 # Logging conrols
27 LOGLEVEL = 2
28 DEBUGFORMAT = False
29
30 import tempfile
31
32 import pytan
33 handler = pytan.Handler(
34     username=USERNAME,
35     password=PASSWORD,
36     host=HOST,
37     port=PORT,
38     loglevel=LOGLEVEL,
39     debugformat=DEBUGFORMAT,
40 )
41
42 print handler
43
44 # setup the arguments for the handler method
45 kwargs = {}
46 kwargs["objtype"] = u'setting'
47 kwargs["name"] = u'control_address'
48
49 # call the handler with the get method, passing in kwargs for arguments
50 response = handler.get(**kwargs)
51
52 print ""
53 print "Type of response: ", type(response)
54
55 print ""
56 print "print of response:"
57 print response
58
59 print ""
60 print "length of response (number of objects returned): "
61 print len(response)
62

```

```
63 print ""
64 print "print the first object returned in JSON format:"
65 out = response.to_json(response[0])
66 if len(out.splitlines()) > 15:
67     out = out.splitlines()[0:15]
68     out.append('..trimmed for brevity..')
69     out = '\n'.join(out)
70
71 print out
```

### Output from Python Code

```
1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3279
2
3 Type of response: <class 'taniumpy.object_types.system_settings_list.SystemSettingsList'>
4
5 print of response:
6 SystemSettingsList, len: 1
7
8 length of response (number of objects returned):
9 1
10
11 print the first object returned in JSON format:
12 {
13     "_type": "system_setting",
14     "default_value": "512:17473:127.0.0.1",
15     "hidden_flag": 0,
16     "id": 57,
17     "name": "control_address",
18     "read_only_flag": 0,
19     "setting_type": "Server",
20     "value": "512:17473:127.0.0.1",
21     "value_type": "Text"
22 }
```

### Get user by name

Get a user by name

### Example Python Code

```
1 import os
2 import sys
3 sys.dont_write_bytecode = True
4
5 # Determine our script name, script dir
6 my_file = os.path.abspath(sys.argv[0])
7 my_dir = os.path.dirname(my_file)
8
9 # determine the pytan lib dir and add it to the path
10 parent_dir = os.path.dirname(my_dir)
11 pytan_root_dir = os.path.dirname(parent_dir)
12 lib_dir = os.path.join(pytan_root_dir, 'lib')
13 path_adds = [lib_dir]
```

```

14
15 for aa in path_adds:
16     if aa not in sys.path:
17         sys.path.append(aa)
18
19
20 # connection info for Tanium Server
21 USERNAME = "Tanium User"
22 PASSWORD = "T@n!um"
23 HOST = "172.16.31.128"
24 PORT = "444"
25
26 # Logging conrols
27 LOGLEVEL = 2
28 DEBUGFORMAT = False
29
30 import tempfile
31
32 import pytan
33 handler = pytan.Handler(
34     username=USERNAME,
35     password=PASSWORD,
36     host=HOST,
37     port=PORT,
38     loglevel=LOGLEVEL,
39     debugformat=DEBUGFORMAT,
40 )
41
42 print handler
43
44 # setup the arguments for the handler method
45 kwargs = {}
46 kwargs["objtype"] = u'user'
47 kwargs["name"] = u'Tanium User'
48
49 # call the handler with the get method, passing in kwargs for arguments
50 response = handler.get(**kwargs)
51
52 print ""
53 print "Type of response: ", type(response)
54
55 print ""
56 print "print of response:"
57 print response
58
59 print ""
60 print "length of response (number of objects returned): "
61 print len(response)
62
63 print ""
64 print "print the first object returned in JSON format:"
65 out = response.to_json(response[0])
66 if len(out.splitlines()) > 15:
67     out = out.splitlines()[0:15]
68     out.append('..trimmed for brevity..')
69     out = '\n'.join(out)
70

```

```
71 print out
```

### Output from Python Code

```
1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3279
2
3 Type of response: <class 'taniumpy.object_types.user_list.UserList'>
4
5 print of response:
6 UserList, len: 1
7
8 length of response (number of objects returned):
9 1
10
11 print the first object returned in JSON format:
12 {
13     "_type": "user",
14     "deleted_flag": 0,
15     "group_id": 0,
16     "id": 2,
17     "last_login": "2015-03-26T08:08:24",
18     "name": "Tanium User",
19     "permissions": {
20         "_type": "permissions",
21         "permission": "admin"
22     },
23     "roles": {
24         "_type": "roles",
25         "role": [
26             {
27 ..trimmed for brevity..
```

### Get sensor by id

Get a sensor by id

### Example Python Code

```
1 import os
2 import sys
3 sys.dont_write_bytecode = True
4
5 # Determine our script name, script dir
6 my_file = os.path.abspath(sys.argv[0])
7 my_dir = os.path.dirname(my_file)
8
9 # determine the pytan lib dir and add it to the path
10 parent_dir = os.path.dirname(my_dir)
11 pytan_root_dir = os.path.dirname(parent_dir)
12 lib_dir = os.path.join(pytan_root_dir, 'lib')
13 path_adds = [lib_dir]
14
15 for aa in path_adds:
16     if aa not in sys.path:
```

```

17         sys.path.append(aa)
18
19
20     # connection info for Tanium Server
21     USERNAME = "Tanium User"
22     PASSWORD = "T@n!um"
23     HOST = "172.16.31.128"
24     PORT = "444"
25
26     # Logging conrols
27     LOGLEVEL = 2
28     DEBUGFORMAT = False
29
30     import tempfile
31
32     import pytan
33     handler = pytan.Handler(
34         username=USERNAME,
35         password=PASSWORD,
36         host=HOST,
37         port=PORT,
38         loglevel=LOGLEVEL,
39         debugformat=DEBUGFORMAT,
40     )
41
42     print handler
43
44     # setup the arguments for the handler method
45     kwargs = {}
46     kwargs["objtype"] = u'sensor'
47     kwargs["id"] = 1
48
49     # call the handler with the get method, passing in kwargs for arguments
50     response = handler.get(**kwargs)
51
52     print ""
53     print "Type of response: ", type(response)
54
55     print ""
56     print "print of response:"
57     print response
58
59     print ""
60     print "length of response (number of objects returned): "
61     print len(response)
62
63     print ""
64     print "print the first object returned in JSON format:"
65     out = response.to_json(response[0])
66     if len(out.splitlines()) > 15:
67         out = out.splitlines()[0:15]
68         out.append('..trimmed for brevity..')
69         out = '\n'.join(out)
70
71     print out

```

## Output from Python Code

```
1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3279
2
3 Type of response: <class 'taniumpy.object_types.sensor_list.SensorList'>
4
5 print of response:
6 SensorList, len: 1
7
8 length of response (number of objects returned):
9 1
10
11 print the first object returned in JSON format:
12 {
13     "_type": "sensor",
14     "category": "Reserved",
15     "description": "The recorded state of each action a client has taken recently in the form of id:st
16     "exclude_from_parse_flag": 1,
17     "hash": 1792443391,
18     "hidden_flag": 0,
19     "id": 1,
20     "ignore_case_flag": 1,
21     "max_age_seconds": 3600,
22     "name": "Action Statuses",
23     "queries": {
24         "_type": "queries",
25         "query": [
26             {
27 ..trimmed for brevity..
```

### Get sensor by mixed

Get multiple sensors by id, name, and hash

### Example Python Code

```
1 import os
2 import sys
3 sys.dont_write_bytecode = True
4
5 # Determine our script name, script dir
6 my_file = os.path.abspath(sys.argv[0])
7 my_dir = os.path.dirname(my_file)
8
9 # determine the pytan lib dir and add it to the path
10 parent_dir = os.path.dirname(my_dir)
11 pytan_root_dir = os.path.dirname(parent_dir)
12 lib_dir = os.path.join(pytan_root_dir, 'lib')
13 path_adds = [lib_dir]
14
15 for aa in path_adds:
16     if aa not in sys.path:
17         sys.path.append(aa)
18
19
20 # connection info for Tanium Server
21 USERNAME = "Tanium User"
```



```

22 PASSWORD = "T@n!um"
23 HOST = "172.16.31.128"
24 PORT = "444"
25
26 # Logging conrols
27 LOGLEVEL = 2
28 DEBUGFORMAT = False
29
30 import tempfile
31
32 import pytan
33 handler = pytan.Handler(
34     username=USERNAME,
35     password=PASSWORD,
36     host=HOST,
37     port=PORT,
38     loglevel=LOGLEVEL,
39     debugformat=DEBUGFORMAT,
40 )
41
42 print handler
43
44 # setup the arguments for the handler method
45 kwargs = {}
46 kwargs["objtype"] = u'sensor'
47 kwargs["hash"] = [u'322086833']
48 kwargs["name"] = [u'Computer Name']
49 kwargs["id"] = [1, 2]
50
51 # call the handler with the get method, passing in kwargs for arguments
52 response = handler.get(**kwargs)
53
54 print ""
55 print "Type of response: ", type(response)
56
57 print ""
58 print "print of response:"
59 print response
60
61 print ""
62 print "length of response (number of objects returned): "
63 print len(response)
64
65 print ""
66 print "print the first object returned in JSON format:"
67 out = response.to_json(response[0])
68 if len(out.splitlines()) > 15:
69     out = out.splitlines()[0:15]
70     out.append('..trimmed for brevity..')
71     out = '\n'.join(out)
72
73 print out

```

### Output from Python Code

```

1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3279
2

```

```
3 | Type of response: <class 'taniumpy.object_types.sensor_list.SensorList'>
4 |
5 | print of response:
6 | SensorList, len: 4
7 |
8 | length of response (number of objects returned):
9 | 4
10 |
11 | print the first object returned in JSON format:
12 | {
13 |     "_type": "sensor",
14 |     "category": "Reserved",
15 |     "description": "The recorded state of each download a client has made recently in the form of hash",
16 |     "exclude_from_parse_flag": 0,
17 |     "hash": 322086833,
18 |     "hidden_flag": 0,
19 |     "id": 4,
20 |     "ignore_case_flag": 1,
21 |     "max_age_seconds": 900,
22 |     "name": "Download Statuses",
23 |     "queries": {
24 |         "_type": "queries",
25 |         "query": [
26 |             {
27 |                 ..trimmed for brevity..
```

### Get whitelisted url by id

Get a whitelisted url by id

### Example Python Code

```
1 | import os
2 | import sys
3 | sys.dont_write_bytecode = True
4 |
5 | # Determine our script name, script dir
6 | my_file = os.path.abspath(sys.argv[0])
7 | my_dir = os.path.dirname(my_file)
8 |
9 | # determine the pytan lib dir and add it to the path
10 | parent_dir = os.path.dirname(my_dir)
11 | pytan_root_dir = os.path.dirname(parent_dir)
12 | lib_dir = os.path.join(pytan_root_dir, 'lib')
13 | path_adds = [lib_dir]
14 |
15 | for aa in path_adds:
16 |     if aa not in sys.path:
17 |         sys.path.append(aa)
18 |
19 |
20 | # connection info for Tanium Server
21 | USERNAME = "Tanium User"
22 | PASSWORD = "T@n!um"
23 | HOST = "172.16.31.128"
```

```

24 PORT = "444"
25
26 # Logging controls
27 LOGLEVEL = 2
28 DEBUGFORMAT = False
29
30 import tempfile
31
32 import pytan
33 handler = pytan.Handler(
34     username=USERNAME,
35     password=PASSWORD,
36     host=HOST,
37     port=PORT,
38     loglevel=LOGLEVEL,
39     debugformat=DEBUGFORMAT,
40 )
41
42 print handler
43
44 # setup the arguments for the handler method
45 kwargs = {}
46 kwargs["objtype"] = u'whitelisted_url'
47 kwargs["id"] = 1
48
49 # call the handler with the get method, passing in kwargs for arguments
50 response = handler.get(**kwargs)
51
52 print ""
53 print "Type of response: ", type(response)
54
55 print ""
56 print "print of response:"
57 print response
58
59 print ""
60 print "length of response (number of objects returned): "
61 print len(response)
62
63 print ""
64 print "print the first object returned in JSON format:"
65 out = response.to_json(response[0])
66 if len(out.splitlines()) > 15:
67     out = out.splitlines()[0:15]
68     out.append('..trimmed for brevity..')
69     out = '\n'.join(out)
70
71 print out

```

### Output from Python Code

```

1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3279
2
3 Type of response: <class 'taniumpy.object_types.white_listed_url_list.WhiteListedUrlList'>
4
5 print of response:
6 WhiteListedUrlList, len: 1

```

```
7
8 length of response (number of objects returned):
9 1
10
11 print the first object returned in JSON format:
12 {
13     "_type": "white_listed_url",
14     "download_seconds": 86400,
15     "id": 1,
16     "url_regex": "test1"
17 }
```

## Get group by name

Get a group by name

## Example Python Code

```
1 import os
2 import sys
3 sys.dont_write_bytecode = True
4
5 # Determine our script name, script dir
6 my_file = os.path.abspath(sys.argv[0])
7 my_dir = os.path.dirname(my_file)
8
9 # determine the pytan lib dir and add it to the path
10 parent_dir = os.path.dirname(my_dir)
11 pytan_root_dir = os.path.dirname(parent_dir)
12 lib_dir = os.path.join(pytan_root_dir, 'lib')
13 path_adds = [lib_dir]
14
15 for aa in path_adds:
16     if aa not in sys.path:
17         sys.path.append(aa)
18
19
20 # connection info for Tanium Server
21 USERNAME = "Tanium User"
22 PASSWORD = "T@n!um"
23 HOST = "172.16.31.128"
24 PORT = "444"
25
26 # Logging conrols
27 LOGLEVEL = 2
28 DEBUGFORMAT = False
29
30 import tempfile
31
32 import pytan
33 handler = pytan.Handler(
34     username=USERNAME,
35     password=PASSWORD,
36     host=HOST,
37     port=PORT,
```

```

38     loglevel=LOGLEVEL,
39     debugformat=DEBUGFORMAT,
40 )
41
42 print handler
43
44 # setup the arguments for the handler method
45 kwargs = {}
46 kwargs["objtype"] = u'group'
47 kwargs["name"] = u'All Computers'
48
49 # call the handler with the get method, passing in kwargs for arguments
50 response = handler.get(**kwargs)
51
52 print ""
53 print "Type of response: ", type(response)
54
55 print ""
56 print "print of response:"
57 print response
58
59 print ""
60 print "length of response (number of objects returned): "
61 print len(response)
62
63 print ""
64 print "print the first object returned in JSON format:"
65 out = response.to_json(response[0])
66 if len(out.splitlines()) > 15:
67     out = out.splitlines()[0:15]
68     out.append('..trimmed for brevity..')
69     out = '\n'.join(out)
70
71 print out

```

### Output from Python Code

```

1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3279
2
3 Type of response: <class 'taniumpy.object_types.group_list.GroupList'>
4
5 print of response:
6 GroupList, len: 1
7
8 length of response (number of objects returned):
9 1
10
11 print the first object returned in JSON format:
12 {
13     "_type": "group",
14     "and_flag": 0,
15     "deleted_flag": 0,
16     "filters": {
17         "_type": "filters",
18         "filter": []
19     },
20     "id": 218,

```

```
21     "name": "All Computers",
22     "not_flag": 0,
23     "sub_groups": {
24         "_type": "groups",
25         "group": []
26     },
27     ..trimmed for brevity..
```

## Get sensor by hash

Get a sensor by hash

## Example Python Code

```
1  import os
2  import sys
3  sys.dont_write_bytecode = True
4
5  # Determine our script name, script dir
6  my_file = os.path.abspath(sys.argv[0])
7  my_dir = os.path.dirname(my_file)
8
9  # determine the pytan lib dir and add it to the path
10 parent_dir = os.path.dirname(my_dir)
11 pytan_root_dir = os.path.dirname(parent_dir)
12 lib_dir = os.path.join(pytan_root_dir, 'lib')
13 path_adds = [lib_dir]
14
15 for aa in path_adds:
16     if aa not in sys.path:
17         sys.path.append(aa)
18
19
20 # connection info for Tanium Server
21 USERNAME = "Tanium User"
22 PASSWORD = "T@n!um"
23 HOST = "172.16.31.128"
24 PORT = "444"
25
26 # Logging conrols
27 LOGLEVEL = 2
28 DEBUGFORMAT = False
29
30 import tempfile
31
32 import pytan
33 handler = pytan.Handler(
34     username=USERNAME,
35     password=PASSWORD,
36     host=HOST,
37     port=PORT,
38     loglevel=LOGLEVEL,
39     debugformat=DEBUGFORMAT,
40 )
41
```

```

42 print handler
43
44 # setup the arguments for the handler method
45 kwargs = {}
46 kwargs["objtype"] = u'sensor'
47 kwargs["hash"] = u'322086833'
48
49 # call the handler with the get method, passing in kwargs for arguments
50 response = handler.get(**kwargs)
51
52 print ""
53 print "Type of response: ", type(response)
54
55 print ""
56 print "print of response:"
57 print response
58
59 print ""
60 print "length of response (number of objects returned): "
61 print len(response)
62
63 print ""
64 print "print the first object returned in JSON format:"
65 out = response.to_json(response[0])
66 if len(out.splitlines()) > 15:
67     out = out.splitlines()[0:15]
68     out.append('..trimmed for brevity..')
69     out = '\n'.join(out)
70
71 print out

```

### Output from Python Code

```

1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3279
2
3 Type of response: <class 'taniumpy.object_types.sensor_list.SensorList'>
4
5 print of response:
6 SensorList, len: 1
7
8 length of response (number of objects returned):
9 1
10
11 print the first object returned in JSON format:
12 {
13     "_type": "sensor",
14     "category": "Reserved",
15     "description": "The recorded state of each download a client has made recently in the form of hash
16     "exclude_from_parse_flag": 0,
17     "hash": 322086833,
18     "hidden_flag": 0,
19     "id": 4,
20     "ignore_case_flag": 1,
21     "max_age_seconds": 900,
22     "name": "Download Statuses",
23     "queries": {
24         "_type": "queries",

```

```
25     "query": [  
26         {  
27     ..trimmed for brevity..
```

## Get package by name

Get a package by name

## Example Python Code

```
1  import os  
2  import sys  
3  sys.dont_write_bytecode = True  
4  
5  # Determine our script name, script dir  
6  my_file = os.path.abspath(sys.argv[0])  
7  my_dir = os.path.dirname(my_file)  
8  
9  # determine the pytan lib dir and add it to the path  
10 parent_dir = os.path.dirname(my_dir)  
11 pytan_root_dir = os.path.dirname(parent_dir)  
12 lib_dir = os.path.join(pytan_root_dir, 'lib')  
13 path_adds = [lib_dir]  
14  
15 for aa in path_adds:  
16     if aa not in sys.path:  
17         sys.path.append(aa)  
18  
19  
20 # connection info for Tanium Server  
21 USERNAME = "Tanium User"  
22 PASSWORD = "T@n!um"  
23 HOST = "172.16.31.128"  
24 PORT = "444"  
25  
26 # Logging conrols  
27 LOGLEVEL = 2  
28 DEBUGFORMAT = False  
29  
30 import tempfile  
31  
32 import pytan  
33 handler = pytan.Handler(  
34     username=USERNAME,  
35     password=PASSWORD,  
36     host=HOST,  
37     port=PORT,  
38     loglevel=LOGLEVEL,  
39     debugformat=DEBUGFORMAT,  
40 )  
41  
42 print handler  
43  
44 # setup the arguments for the handler method  
45 kwargs = {}
```



```

46 kwargs["objtype"] = u'package'
47 kwargs["name"] = u'Distribute Patch Tools'
48
49 # call the handler with the get method, passing in kwargs for arguments
50 response = handler.get(**kwargs)
51
52 print ""
53 print "Type of response: ", type(response)
54
55 print ""
56 print "print of response:"
57 print response
58
59 print ""
60 print "length of response (number of objects returned): "
61 print len(response)
62
63 print ""
64 print "print the first object returned in JSON format:"
65 out = response.to_json(response[0])
66 if len(out.splitlines()) > 15:
67     out = out.splitlines()[0:15]
68     out.append('..trimmed for brevity..')
69     out = '\n'.join(out)
70
71 print out

```

### Output from Python Code

```

1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3279
2
3 Type of response: <class 'taniumpy.object_types.package_spec_list.PackageSpecList'>
4
5 print of response:
6 PackageSpecList, len: 1
7
8 length of response (number of objects returned):
9 1
10
11 print the first object returned in JSON format:
12 {
13     "_type": "package_spec",
14     "available_time": "2015-03-03T19:11:01",
15     "command": "cmd /c cscript //T:1800 copy-to-tanium-dir.vbs \"Tools\\\"",
16     "command_timeout": 1800,
17     "creation_time": "2015-03-03T19:04:09",
18     "deleted_flag": 0,
19     "display_name": "Distribute Patch Tools",
20     "expire_seconds": 2400,
21     "files": {
22         "_type": "package_files",
23         "file": [
24             {
25                 "_type": "file",
26                 "bytes_downloaded": 3041,
27                 ..trimmed for brevity..

```

## Get sensor by names

Get multiple sensors by name

### Example Python Code

```
1  import os
2  import sys
3  sys.dont_write_bytecode = True
4
5  # Determine our script name, script dir
6  my_file = os.path.abspath(sys.argv[0])
7  my_dir = os.path.dirname(my_file)
8
9  # determine the pytan lib dir and add it to the path
10 parent_dir = os.path.dirname(my_dir)
11 pytan_root_dir = os.path.dirname(parent_dir)
12 lib_dir = os.path.join(pytan_root_dir, 'lib')
13 path_adds = [lib_dir]
14
15 for aa in path_adds:
16     if aa not in sys.path:
17         sys.path.append(aa)
18
19
20 # connection info for Tanium Server
21 USERNAME = "Tanium User"
22 PASSWORD = "T@n!um"
23 HOST = "172.16.31.128"
24 PORT = "444"
25
26 # Logging conrols
27 LOGLEVEL = 2
28 DEBUGFORMAT = False
29
30 import tempfile
31
32 import pytan
33 handler = pytan.Handler(
34     username=USERNAME,
35     password=PASSWORD,
36     host=HOST,
37     port=PORT,
38     loglevel=LOGLEVEL,
39     debugformat=DEBUGFORMAT,
40 )
41
42 print handler
43
44 # setup the arguments for the handler method
45 kwargs = {}
46 kwargs["objtype"] = u'sensor'
47 kwargs["name"] = [u'Computer Name', u'Action Statuses']
48
49 # call the handler with the get method, passing in kwargs for arguments
50 response = handler.get(**kwargs)
51
52 print ""
```

```

53 print "Type of response: ", type(response)
54
55 print ""
56 print "print of response:"
57 print response
58
59 print ""
60 print "length of response (number of objects returned): "
61 print len(response)
62
63 print ""
64 print "print the first object returned in JSON format:"
65 out = response.to_json(response[0])
66 if len(out.splitlines()) > 15:
67     out = out.splitlines()[0:15]
68     out.append('..trimmed for brevity..')
69     out = '\n'.join(out)
70
71 print out

```

### Output from Python Code

```

1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3279
2
3 Type of response: <class 'taniumpy.object_types.sensor_list.SensorList'>
4
5 print of response:
6 SensorList, len: 2
7
8 length of response (number of objects returned):
9 2
10
11 print the first object returned in JSON format:
12 {
13     "_type": "sensor",
14     "category": "Reserved",
15     "description": "The assigned name of the client machine.\nExample: workstation-1.company.com",
16     "exclude_from_parse_flag": 0,
17     "hash": 3409330187,
18     "hidden_flag": 0,
19     "id": 3,
20     "ignore_case_flag": 1,
21     "max_age_seconds": 86400,
22     "name": "Computer Name",
23     "queries": {
24         "_type": "queries",
25         "query": [
26             {
27                 ..trimmed for brevity..

```

### Get saved question by name

Get saved question by name

## Example Python Code

```
1  import os
2  import sys
3  sys.dont_write_bytecode = True
4
5  # Determine our script name, script dir
6  my_file = os.path.abspath(sys.argv[0])
7  my_dir = os.path.dirname(my_file)
8
9  # determine the pytan lib dir and add it to the path
10 parent_dir = os.path.dirname(my_dir)
11 pytan_root_dir = os.path.dirname(parent_dir)
12 lib_dir = os.path.join(pytan_root_dir, 'lib')
13 path_adds = [lib_dir]
14
15 for aa in path_adds:
16     if aa not in sys.path:
17         sys.path.append(aa)
18
19
20 # connection info for Tanium Server
21 USERNAME = "Tanium User"
22 PASSWORD = "T@n!um"
23 HOST = "172.16.31.128"
24 PORT = "444"
25
26 # Logging conrols
27 LOGLEVEL = 2
28 DEBUGFORMAT = False
29
30 import tempfile
31
32 import pytan
33 handler = pytan.Handler(
34     username=USERNAME,
35     password=PASSWORD,
36     host=HOST,
37     port=PORT,
38     loglevel=LOGLEVEL,
39     debugformat=DEBUGFORMAT,
40 )
41
42 print handler
43
44 # setup the arguments for the handler method
45 kwargs = {}
46 kwargs["objtype"] = u'saved_question'
47 kwargs["name"] = u'Installed Applications'
48
49 # call the handler with the get method, passing in kwargs for arguments
50 response = handler.get(**kwargs)
51
52 print ""
53 print "Type of response: ", type(response)
54
55 print ""
56 print "print of response:"
57 print response
```

```

58
59 print ""
60 print "length of response (number of objects returned): "
61 print len(response)
62
63 print ""
64 print "print the first object returned in JSON format:"
65 out = response.to_json(response[0])
66 if len(out.splitlines()) > 15:
67     out = out.splitlines()[0:15]
68     out.append('..trimmed for brevity..')
69     out = '\n'.join(out)
70
71 print out

```

### Output from Python Code

```

1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3279
2
3 Type of response: <class 'taniumpy.object_types.saved_question_list.SavedQuestionList'>
4
5 print of response:
6 SavedQuestionList, len: 1
7
8 length of response (number of objects returned):
9 1
10
11 print the first object returned in JSON format:
12 {
13     "_type": "saved_question",
14     "action_tracking_flag": 0,
15     "archive_enabled_flag": 0,
16     "archive_owner": {
17         "_type": "user",
18         "id": 1,
19         "name": "Jim Olsen"
20     },
21     "expire_seconds": 600,
22     "hidden_flag": 0,
23     "id": 92,
24     "issue_seconds": 120,
25     "issue_seconds_never_flag": 0,
26     "keep_seconds": 3600,
27     ..trimmed for brevity..

```

### Get user by id

Get a user by id

### Example Python Code

```

1 import os
2 import sys
3 sys.dont_write_bytecode = True

```

```
4
5 # Determine our script name, script dir
6 my_file = os.path.abspath(sys.argv[0])
7 my_dir = os.path.dirname(my_file)
8
9 # determine the pytan lib dir and add it to the path
10 parent_dir = os.path.dirname(my_dir)
11 pytan_root_dir = os.path.dirname(parent_dir)
12 lib_dir = os.path.join(pytan_root_dir, 'lib')
13 path_adds = [lib_dir]
14
15 for aa in path_adds:
16     if aa not in sys.path:
17         sys.path.append(aa)
18
19
20 # connection info for Tanium Server
21 USERNAME = "Tanium User"
22 PASSWORD = "T@n!um"
23 HOST = "172.16.31.128"
24 PORT = "444"
25
26 # Logging conrols
27 LOGLEVEL = 2
28 DEBUGFORMAT = False
29
30 import tempfile
31
32 import pytan
33 handler = pytan.Handler(
34     username=USERNAME,
35     password=PASSWORD,
36     host=HOST,
37     port=PORT,
38     loglevel=LOGLEVEL,
39     debugformat=DEBUGFORMAT,
40 )
41
42 print handler
43
44 # setup the arguments for the handler method
45 kwargs = {}
46 kwargs["objtype"] = u'user'
47 kwargs["id"] = 1
48
49 # call the handler with the get method, passing in kwargs for arguments
50 response = handler.get(**kwargs)
51
52 print ""
53 print "Type of response: ", type(response)
54
55 print ""
56 print "print of response:"
57 print response
58
59 print ""
60 print "length of response (number of objects returned): "
61 print len(response)
```

```

62
63 print ""
64 print "print the first object returned in JSON format:"
65 out = response.to_json(response[0])
66 if len(out.splitlines()) > 15:
67     out = out.splitlines()[0:15]
68     out.append('..trimmed for brevity..')
69     out = '\n'.join(out)
70
71 print out

```

### Output from Python Code

```

1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3279
2
3 Type of response: <class 'taniumpy.object_types.user_list.UserList'>
4
5 print of response:
6 UserList, len: 1
7
8 length of response (number of objects returned):
9 1
10
11 print the first object returned in JSON format:
12 {
13     "_type": "user",
14     "deleted_flag": 0,
15     "group_id": 0,
16     "id": 1,
17     "last_login": "2015-03-25T13:19:16",
18     "metadata": {
19         "_type": "metadata",
20         "item": [
21             {
22                 "_type": "item",
23                 "admin_flag": 0,
24                 "name": "TConsole.User.Preference.FilterClientsPeriod",
25                 "value": "43200"
26             }
27         ]
28     }
29 }
30 ..trimmed for brevity..

```

### Get sensor by name

Get a sensor by name

### Example Python Code

```

1 import os
2 import sys
3 sys.dont_write_bytecode = True
4
5 # Determine our script name, script dir
6 my_file = os.path.abspath(sys.argv[0])
7 my_dir = os.path.dirname(my_file)

```

```
8
9  # determine the pytan lib dir and add it to the path
10 parent_dir = os.path.dirname(my_dir)
11 pytan_root_dir = os.path.dirname(parent_dir)
12 lib_dir = os.path.join(pytan_root_dir, 'lib')
13 path_adds = [lib_dir]
14
15 for aa in path_adds:
16     if aa not in sys.path:
17         sys.path.append(aa)
18
19
20 # connection info for Tanium Server
21 USERNAME = "Tanium User"
22 PASSWORD = "T@n!um"
23 HOST = "172.16.31.128"
24 PORT = "444"
25
26 # Logging conrols
27 LOGLEVEL = 2
28 DEBUGFORMAT = False
29
30 import tempfile
31
32 import pytan
33 handler = pytan.Handler(
34     username=USERNAME,
35     password=PASSWORD,
36     host=HOST,
37     port=PORT,
38     loglevel=LOGLEVEL,
39     debugformat=DEBUGFORMAT,
40 )
41
42 print handler
43
44 # setup the arguments for the handler method
45 kwargs = {}
46 kwargs["objtype"] = u'sensor'
47 kwargs["name"] = u'Computer Name'
48
49 # call the handler with the get method, passing in kwargs for arguments
50 response = handler.get(**kwargs)
51
52 print ""
53 print "Type of response: ", type(response)
54
55 print ""
56 print "print of response:"
57 print response
58
59 print ""
60 print "length of response (number of objects returned): "
61 print len(response)
62
63 print ""
64 print "print the first object returned in JSON format:"
65 out = response.to_json(response[0])
```



```

66 if len(out.splitlines()) > 15:
67     out = out.splitlines()[0:15]
68     out.append('..trimmed for brevity..')
69     out = '\n'.join(out)
70
71 print out

```

### Output from Python Code

```

1  Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3279
2
3  Type of response: <class 'taniumpy.object_types.sensor_list.SensorList'>
4
5  print of response:
6  SensorList, len: 1
7
8  length of response (number of objects returned):
9  1
10
11 print the first object returned in JSON format:
12 {
13     "_type": "sensor",
14     "category": "Reserved",
15     "description": "The assigned name of the client machine.\nExample: workstation-1.company.com",
16     "exclude_from_parse_flag": 0,
17     "hash": 3409330187,
18     "hidden_flag": 0,
19     "id": 3,
20     "ignore_case_flag": 1,
21     "max_age_seconds": 86400,
22     "name": "Computer Name",
23     "queries": {
24         "_type": "queries",
25         "query": [
26             {
27 ..trimmed for brevity..

```

### Get saved action by name

Get a saved action by name

### Example Python Code

```

1  import os
2  import sys
3  sys.dont_write_bytecode = True
4
5  # Determine our script name, script dir
6  my_file = os.path.abspath(sys.argv[0])
7  my_dir = os.path.dirname(my_file)
8
9  # determine the pytan lib dir and add it to the path
10 parent_dir = os.path.dirname(my_dir)
11 pytan_root_dir = os.path.dirname(parent_dir)

```

```
12 lib_dir = os.path.join(pytan_root_dir, 'lib')
13 path_adds = [lib_dir]
14
15 for aa in path_adds:
16     if aa not in sys.path:
17         sys.path.append(aa)
18
19
20 # connection info for Tanium Server
21 USERNAME = "Tanium User"
22 PASSWORD = "T@n!um"
23 HOST = "172.16.31.128"
24 PORT = "444"
25
26 # Logging conrols
27 LOGLEVEL = 2
28 DEBUGFORMAT = False
29
30 import tempfile
31
32 import pytan
33 handler = pytan.Handler(
34     username=USERNAME,
35     password=PASSWORD,
36     host=HOST,
37     port=PORT,
38     loglevel=LOGLEVEL,
39     debugformat=DEBUGFORMAT,
40 )
41
42 print handler
43
44 # setup the arguments for the handler method
45 kwargs = {}
46 kwargs["objtype"] = u'saved_action'
47 kwargs["name"] = u'Distribute Tanium Standard Utilities'
48
49 # call the handler with the get method, passing in kwargs for arguments
50 response = handler.get(**kwargs)
51
52 print ""
53 print "Type of response: ", type(response)
54
55 print ""
56 print "print of response:"
57 print response
58
59 print ""
60 print "length of response (number of objects returned): "
61 print len(response)
62
63 print ""
64 print "print the first object returned in JSON format:"
65 out = response.to_json(response[0])
66 if len(out.splitlines()) > 15:
67     out = out.splitlines()[0:15]
68     out.append('..trimmed for brevity..')
69     out = '\n'.join(out)
```

```
print out
```

### Output from Python Code

```

1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3279
2
3 Type of response: <class 'taniumpy.object_types.saved_action_list.SavedActionList'>
4
5 print of response:
6 SavedActionList, len: 1
7
8 length of response (number of objects returned):
9 1
10
11 print the first object returned in JSON format:
12 {
13     "_type": "saved_action",
14     "action_group_id": 0,
15     "comment": "Distributes the Hardware Tools used for hardware identification.",
16     "creation_time": "2015-03-03T19:06:00",
17     "distribute_seconds": 0,
18     "end_time": "Never",
19     "expire_seconds": 660,
20     "id": 14,
21     "issue_count": 4,
22     "issue_seconds": 86400,
23     "last_action": {
24         "_type": "action",
25         "id": 15985,
26         "start_time": "2015-03-13T19:06:00"
27     }
28 }
29 ..trimmed for brevity..

```

### Get all users

Get all users

### Example Python Code

```

1 import os
2 import sys
3 sys.dont_write_bytecode = True
4
5 # Determine our script name, script dir
6 my_file = os.path.abspath(sys.argv[0])
7 my_dir = os.path.dirname(my_file)
8
9 # determine the pytan lib dir and add it to the path
10 parent_dir = os.path.dirname(my_dir)
11 pytan_root_dir = os.path.dirname(parent_dir)
12 lib_dir = os.path.join(pytan_root_dir, 'lib')
13 path_adds = [lib_dir]
14
15 for aa in path_adds:

```

```
16     if aa not in sys.path:
17         sys.path.append(aa)
18
19
20     # connection info for Tanium Server
21     USERNAME = "Tanium User"
22     PASSWORD = "T@n!um"
23     HOST = "172.16.31.128"
24     PORT = "444"
25
26     # Logging conrols
27     LOGLEVEL = 2
28     DEBUGFORMAT = False
29
30     import tempfile
31
32     import pytan
33     handler = pytan.Handler(
34         username=USERNAME,
35         password=PASSWORD,
36         host=HOST,
37         port=PORT,
38         loglevel=LOGLEVEL,
39         debugformat=DEBUGFORMAT,
40     )
41
42     print handler
43
44     # setup the arguments for the handler method
45     kwargs = {}
46     kwargs["objtype"] = u'user'
47
48     # call the handler with the get_all method, passing in kwargs for arguments
49     response = handler.get_all(**kwargs)
50
51     print ""
52     print "Type of response: ", type(response)
53
54     print ""
55     print "print of response:"
56     print response
57
58     print ""
59     print "length of response (number of objects returned): "
60     print len(response)
61
62     print ""
63     print "print the first object returned in JSON format:"
64     out = response.to_json(response[0])
65     if len(out.splitlines()) > 15:
66         out = out.splitlines()[0:15]
67         out.append('..trimmed for brevity..')
68         out = '\n'.join(out)
69
70     print out
```

## Output from Python Code

```

1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3279
2
3 Type of response: <class 'taniumpy.object_types.user_list.UserList'>
4
5 print of response:
6 UserList, len: 5
7
8 length of response (number of objects returned):
9 5
10
11 print the first object returned in JSON format:
12 {
13     "_type": "user",
14     "deleted_flag": 0,
15     "group_id": 0,
16     "id": 1,
17     "last_login": "2015-03-25T13:19:16",
18     "metadata": {
19         "_type": "metadata",
20         "item": [
21             {
22                 "_type": "item",
23                 "admin_flag": 0,
24                 "name": "TConsole.User.Preference.FilterClientsPeriod",
25                 "value": "43200"
26             }
27         ]
28     }
29 }
30 ..trimmed for brevity..

```

## Get all saved actions

Get all saved actions

## Example Python Code

```

1 import os
2 import sys
3 sys.dont_write_bytecode = True
4
5 # Determine our script name, script dir
6 my_file = os.path.abspath(sys.argv[0])
7 my_dir = os.path.dirname(my_file)
8
9 # determine the pytan lib dir and add it to the path
10 parent_dir = os.path.dirname(my_dir)
11 pytan_root_dir = os.path.dirname(parent_dir)
12 lib_dir = os.path.join(pytan_root_dir, 'lib')
13 path_adds = [lib_dir]
14
15 for aa in path_adds:
16     if aa not in sys.path:
17         sys.path.append(aa)
18
19
20 # connection info for Tanium Server
21 USERNAME = "Tanium User"

```

```
22 PASSWORD = "T@n!um"
23 HOST = "172.16.31.128"
24 PORT = "444"
25
26 # Logging conrols
27 LOGLEVEL = 2
28 DEBUGFORMAT = False
29
30 import tempfile
31
32 import pytan
33 handler = pytan.Handler(
34     username=USERNAME,
35     password=PASSWORD,
36     host=HOST,
37     port=PORT,
38     loglevel=LOGLEVEL,
39     debugformat=DEBUGFORMAT,
40 )
41
42 print handler
43
44 # setup the arguments for the handler method
45 kwargs = {}
46 kwargs["objtype"] = u'saved_action'
47
48 # call the handler with the get_all method, passing in kwargs for arguments
49 response = handler.get_all(**kwargs)
50
51 print ""
52 print "Type of response: ", type(response)
53
54 print ""
55 print "print of response:"
56 print response
57
58 print ""
59 print "length of response (number of objects returned): "
60 print len(response)
61
62 print ""
63 print "print the first object returned in JSON format:"
64 out = response.to_json(response[0])
65 if len(out.splitlines()) > 15:
66     out = out.splitlines()[0:15]
67     out.append('..trimmed for brevity..')
68     out = '\n'.join(out)
69
70 print out
```

### Output from Python Code

```
1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3279
2
3 Type of response:  <class 'taniumpy.object_types.saved_action_list.SavedActionList'>
4
5 print of response:
```

```

6 SavedActionList, len: 1688
7
8 length of response (number of objects returned):
9 1688
10
11 print the first object returned in JSON format:
12 {
13     "_type": "saved_action",
14     "action_group_id": 0,
15     "cache_row_id": 0,
16     "comment": "Scans for unmanaged assets on the network.",
17     "creation_time": "2015-03-03T19:05:56",
18     "distribute_seconds": 600,
19     "end_time": "Never",
20     "expire_seconds": 1800,
21     "id": 1,
22     "issue_count": 224,
23     "issue_seconds": 3600,
24     "last_action": {
25         "_type": "action",
26         "id": 21075,
27     ..trimmed for brevity..

```

## Get all settings

Get all system settings

## Example Python Code

```

1 import os
2 import sys
3 sys.dont_write_bytecode = True
4
5 # Determine our script name, script dir
6 my_file = os.path.abspath(sys.argv[0])
7 my_dir = os.path.dirname(my_file)
8
9 # determine the pytan lib dir and add it to the path
10 parent_dir = os.path.dirname(my_dir)
11 pytan_root_dir = os.path.dirname(parent_dir)
12 lib_dir = os.path.join(pytan_root_dir, 'lib')
13 path_adds = [lib_dir]
14
15 for aa in path_adds:
16     if aa not in sys.path:
17         sys.path.append(aa)
18
19
20 # connection info for Tanium Server
21 USERNAME = "Tanium User"
22 PASSWORD = "T@n!um"
23 HOST = "172.16.31.128"
24 PORT = "444"
25
26 # Logging conrols

```

```
27 LOGLEVEL = 2
28 DEBUGFORMAT = False
29
30 import tempfile
31
32 import pytan
33 handler = pytan.Handler(
34     username=USERNAME,
35     password=PASSWORD,
36     host=HOST,
37     port=PORT,
38     loglevel=LOGLEVEL,
39     debugformat=DEBUGFORMAT,
40 )
41
42 print handler
43
44 # setup the arguments for the handler method
45 kwargs = {}
46 kwargs["objtype"] = u'setting'
47
48 # call the handler with the get_all method, passing in kwargs for arguments
49 response = handler.get_all(**kwargs)
50
51 print ""
52 print "Type of response: ", type(response)
53
54 print ""
55 print "print of response:"
56 print response
57
58 print ""
59 print "length of response (number of objects returned): "
60 print len(response)
61
62 print ""
63 print "print the first object returned in JSON format:"
64 out = response.to_json(response[0])
65 if len(out.splitlines()) > 15:
66     out = out.splitlines()[0:15]
67     out.append('..trimmed for brevity..')
68     out = '\n'.join(out)
69
70 print out
```

### Output from Python Code

```
1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3279
2
3 Type of response:  <class 'taniumpy.object_types.system_settings_list.SystemSettingsList'>
4
5 print of response:
6 SystemSettingsList, len: 88
7
8 length of response (number of objects returned):
9 88
10
```



```

11 print the first object returned in JSON format:
12 {
13     "_type": "system_setting",
14     "audit_data": {
15         "_type": "audit_data",
16         "creation_time": "2015-03-03T19:06:08",
17         "last_modified_by": "Jim Olsen",
18         "modification_time": "2015-03-03T19:06:08"
19     },
20     "cache_row_id": 0,
21     "default_value": "0",
22     "hidden_flag": 0,
23     "id": 1,
24     "name": "load_initial_content",
25     "read_only_flag": 0,
26     "setting_type": "Server",
27     ..trimmed for brevity..

```

### Get all saved questions

Get all saved questions

### Example Python Code

```

1  import os
2  import sys
3  sys.dont_write_bytecode = True
4
5  # Determine our script name, script dir
6  my_file = os.path.abspath(sys.argv[0])
7  my_dir = os.path.dirname(my_file)
8
9  # determine the pytan lib dir and add it to the path
10 parent_dir = os.path.dirname(my_dir)
11 pytan_root_dir = os.path.dirname(parent_dir)
12 lib_dir = os.path.join(pytan_root_dir, 'lib')
13 path_adds = [lib_dir]
14
15 for aa in path_adds:
16     if aa not in sys.path:
17         sys.path.append(aa)
18
19
20 # connection info for Tanium Server
21 USERNAME = "Tanium User"
22 PASSWORD = "T@n!um"
23 HOST = "172.16.31.128"
24 PORT = "444"
25
26 # Logging conrols
27 LOGLEVEL = 2
28 DEBUGFORMAT = False
29
30 import tempfile
31

```

```
32 import pytan
33 handler = pytan.Handler(
34     username=USERNAME,
35     password=PASSWORD,
36     host=HOST,
37     port=PORT,
38     loglevel=LOGLEVEL,
39     debugformat=DEBUGFORMAT,
40 )
41
42 print handler
43
44 # setup the arguments for the handler method
45 kwargs = {}
46 kwargs["objtype"] = u'saved_question'
47
48 # call the handler with the get_all method, passing in kwargs for arguments
49 response = handler.get_all(**kwargs)
50
51 print ""
52 print "Type of response: ", type(response)
53
54 print ""
55 print "print of response:"
56 print response
57
58 print ""
59 print "length of response (number of objects returned): "
60 print len(response)
61
62 print ""
63 print "print the first object returned in JSON format:"
64 out = response.to_json(response[0])
65 if len(out.splitlines()) > 15:
66     out = out.splitlines()[0:15]
67     out.append('..trimmed for brevity..')
68     out = '\n'.join(out)
69
70 print out
```

### Output from Python Code

```
1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3279
2
3 Type of response: <class 'taniumpy.object_types.saved_question_list.SavedQuestionList'>
4
5 print of response:
6 SavedQuestionList, len: 175
7
8 length of response (number of objects returned):
9 175
10
11 print the first object returned in JSON format:
12 {
13     "_type": "saved_question",
14     "action_tracking_flag": 0,
15     "archive_enabled_flag": 0,
```

```

16     "archive_owner": {
17         "_type": "user",
18         "id": 1,
19         "name": "Jim Olsen"
20     },
21     "cache_row_id": 0,
22     "expire_seconds": 600,
23     "hidden_flag": 0,
24     "id": 1,
25     "issue_seconds": 120,
26     "issue_seconds_never_flag": 0,
27     ..trimmed for brevity..

```

### Get all userroles

Get all user roles

### Example Python Code

```

1  import os
2  import sys
3  sys.dont_write_bytecode = True
4
5  # Determine our script name, script dir
6  my_file = os.path.abspath(sys.argv[0])
7  my_dir = os.path.dirname(my_file)
8
9  # determine the pytan lib dir and add it to the path
10 parent_dir = os.path.dirname(my_dir)
11 pytan_root_dir = os.path.dirname(parent_dir)
12 lib_dir = os.path.join(pytan_root_dir, 'lib')
13 path_adds = [lib_dir]
14
15 for aa in path_adds:
16     if aa not in sys.path:
17         sys.path.append(aa)
18
19
20 # connection info for Tanium Server
21 USERNAME = "Tanium User"
22 PASSWORD = "T@n!um"
23 HOST = "172.16.31.128"
24 PORT = "444"
25
26 # Logging conrols
27 LOGLEVEL = 2
28 DEBUGFORMAT = False
29
30 import tempfile
31
32 import pytan
33 handler = pytan.Handler(
34     username=USERNAME,
35     password=PASSWORD,
36     host=HOST,

```

```
37     port=PORT,
38     loglevel=LOGLEVEL,
39     debugformat=DEBUGFORMAT,
40 )
41
42 print handler
43
44 # setup the arguments for the handler method
45 kwargs = {}
46 kwargs["objtype"] = u'userrole'
47
48 # call the handler with the get_all method, passing in kwargs for arguments
49 response = handler.get_all(**kwargs)
50
51 print ""
52 print "Type of response: ", type(response)
53
54 print ""
55 print "print of response:"
56 print response
57
58 print ""
59 print "length of response (number of objects returned): "
60 print len(response)
61
62 print ""
63 print "print the first object returned in JSON format:"
64 out = response.to_json(response[0])
65 if len(out.splitlines()) > 15:
66     out = out.splitlines()[0:15]
67     out.append('..trimmed for brevity..')
68     out = '\n'.join(out)
69
70 print out
```

### Output from Python Code

```
1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3279
2
3 Type of response: <class 'taniumpy.object_types.user_role_list.UserRoleList'>
4
5 print of response:
6 UserRoleList, len: 9
7
8 length of response (number of objects returned):
9 9
10
11 print the first object returned in JSON format:
12 {
13     "_type": "role",
14     "description": "Administrators can perform all functions in the system, including creating other u
15     "id": 1,
16     "name": "Administrator",
17     "permissions": {
18         "_type": "permissions",
19         "permission": "admin"
20     }
```

```
21 }
```

## Get all questions

Get all questions

## Example Python Code

```
1 import os
2 import sys
3 sys.dont_write_bytecode = True
4
5 # Determine our script name, script dir
6 my_file = os.path.abspath(sys.argv[0])
7 my_dir = os.path.dirname(my_file)
8
9 # determine the pytan lib dir and add it to the path
10 parent_dir = os.path.dirname(my_dir)
11 pytan_root_dir = os.path.dirname(parent_dir)
12 lib_dir = os.path.join(pytan_root_dir, 'lib')
13 path_adds = [lib_dir]
14
15 for aa in path_adds:
16     if aa not in sys.path:
17         sys.path.append(aa)
18
19
20 # connection info for Tanium Server
21 USERNAME = "Tanium User"
22 PASSWORD = "T@n!um"
23 HOST = "172.16.31.128"
24 PORT = "444"
25
26 # Logging conrols
27 LOGLEVEL = 2
28 DEBUGFORMAT = False
29
30 import tempfile
31
32 import pytan
33 handler = pytan.Handler(
34     username=USERNAME,
35     password=PASSWORD,
36     host=HOST,
37     port=PORT,
38     loglevel=LOGLEVEL,
39     debugformat=DEBUGFORMAT,
40 )
41
42 print handler
43
44 # setup the arguments for the handler method
45 kwargs = {}
46 kwargs["objtype"] = u'question'
47
```

```
48 # call the handler with the get_all method, passing in kwargs for arguments
49 response = handler.get_all(**kwargs)
50
51 print ""
52 print "Type of response: ", type(response)
53
54 print ""
55 print "print of response:"
56 print response
57
58 print ""
59 print "length of response (number of objects returned): "
60 print len(response)
61
62 print ""
63 print "print the first object returned in JSON format:"
64 out = response.to_json(response[0])
65 if len(out.splitlines()) > 15:
66     out = out.splitlines()[0:15]
67     out.append('..trimmed for brevity..')
68     out = '\n'.join(out)
69
70 print out
```

### Output from Python Code

```
1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3279
2
3 Type of response: <class 'taniumpy.object_types.question_list.QuestionList'>
4
5 print of response:
6 QuestionList, len: 1997
7
8 length of response (number of objects returned):
9 1997
10
11 print the first object returned in JSON format:
12 {
13     "_type": "question",
14     "action_tracking_flag": 0,
15     "cache_row_id": 1,
16     "context_group": {
17         "_type": "group",
18         "id": 0
19     },
20     "expiration": "2015-03-19T00:07:36",
21     "expire_seconds": 600,
22     "hidden_flag": 0,
23     "id": 26988,
24     "management_rights_group": {
25         "_type": "group",
26         "id": 0
27     }
28     ..trimmed for brevity..
```

## Get all groups

Get all groups

### Example Python Code

```

1  import os
2  import sys
3  sys.dont_write_bytecode = True
4
5  # Determine our script name, script dir
6  my_file = os.path.abspath(sys.argv[0])
7  my_dir = os.path.dirname(my_file)
8
9  # determine the pytan lib dir and add it to the path
10 parent_dir = os.path.dirname(my_dir)
11 pytan_root_dir = os.path.dirname(parent_dir)
12 lib_dir = os.path.join(pytan_root_dir, 'lib')
13 path_adds = [lib_dir]
14
15 for aa in path_adds:
16     if aa not in sys.path:
17         sys.path.append(aa)
18
19
20 # connection info for Tanium Server
21 USERNAME = "Tanium User"
22 PASSWORD = "T@n!um"
23 HOST = "172.16.31.128"
24 PORT = "444"
25
26 # Logging conrols
27 LOGLEVEL = 2
28 DEBUGFORMAT = False
29
30 import tempfile
31
32 import pytan
33 handler = pytan.Handler(
34     username=USERNAME,
35     password=PASSWORD,
36     host=HOST,
37     port=PORT,
38     loglevel=LOGLEVEL,
39     debugformat=DEBUGFORMAT,
40 )
41
42 print handler
43
44 # setup the arguments for the handler method
45 kwargs = {}
46 kwargs["objtype"] = u'group'
47
48 # call the handler with the get_all method, passing in kwargs for arguments
49 response = handler.get_all(**kwargs)
50
51 print ""
52 print "Type of response: ", type(response)

```

```
53
54 print ""
55 print "print of response:"
56 print response
57
58 print ""
59 print "length of response (number of objects returned): "
60 print len(response)
61
62 print ""
63 print "print the first object returned in JSON format:"
64 out = response.to_json(response[0])
65 if len(out.splitlines()) > 15:
66     out = out.splitlines()[0:15]
67     out.append('..trimmed for brevity..')
68     out = '\n'.join(out)
69
70 print out
```

### Output from Python Code

```
1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3279
2
3 Type of response: <class 'taniumpy.object_types.group_list.GroupList'>
4
5 print of response:
6 GroupList, len: 2
7
8 length of response (number of objects returned):
9 2
10
11 print the first object returned in JSON format:
12 {
13     "_type": "group",
14     "and_flag": 0,
15     "deleted_flag": 0,
16     "filters": {
17         "_type": "filters",
18         "filter": []
19     },
20     "id": 218,
21     "name": "All Computers",
22     "not_flag": 0,
23     "sub_groups": {
24         "_type": "groups",
25         "group": []
26     },
27     ..trimmed for brevity..
```

### Get all sensors

Get all sensors

### Example Python Code



```

1  import os
2  import sys
3  sys.dont_write_bytecode = True
4
5  # Determine our script name, script dir
6  my_file = os.path.abspath(sys.argv[0])
7  my_dir = os.path.dirname(my_file)
8
9  # determine the pytan lib dir and add it to the path
10 parent_dir = os.path.dirname(my_dir)
11 pytan_root_dir = os.path.dirname(parent_dir)
12 lib_dir = os.path.join(pytan_root_dir, 'lib')
13 path_adds = [lib_dir]
14
15 for aa in path_adds:
16     if aa not in sys.path:
17         sys.path.append(aa)
18
19
20 # connection info for Tanium Server
21 USERNAME = "Tanium User"
22 PASSWORD = "T@n!um"
23 HOST = "172.16.31.128"
24 PORT = "444"
25
26 # Logging conrols
27 LOGLEVEL = 2
28 DEBUGFORMAT = False
29
30 import tempfile
31
32 import pytan
33 handler = pytan.Handler(
34     username=USERNAME,
35     password=PASSWORD,
36     host=HOST,
37     port=PORT,
38     loglevel=LOGLEVEL,
39     debugformat=DEBUGFORMAT,
40 )
41
42 print handler
43
44 # setup the arguments for the handler method
45 kwargs = {}
46 kwargs["objtype"] = u'sensor'
47
48 # call the handler with the get_all method, passing in kwargs for arguments
49 response = handler.get_all(**kwargs)
50
51 print ""
52 print "Type of response: ", type(response)
53
54 print ""
55 print "print of response:"
56 print response
57
58 print ""

```

```
59 print "length of response (number of objects returned): "  
60 print len(response)  
61  
62 print ""  
63 print "print the first object returned in JSON format:"  
64 out = response.to_json(response[0])  
65 if len(out.splitlines()) > 15:  
66     out = out.splitlines()[0:15]  
67     out.append('..trimmed for brevity..')  
68     out = '\n'.join(out)  
69  
70 print out
```

### Output from Python Code

```
1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3279  
2  
3 Type of response: <class 'taniumpy.object_types.sensor_list.SensorList'>  
4  
5 print of response:  
6 SensorList, len: 728  
7  
8 length of response (number of objects returned):  
9 728  
10  
11 print the first object returned in JSON format:  
12 {  
13     "_type": "sensor",  
14     "cache_row_id": 0,  
15     "category": "Reserved",  
16     "description": "The recorded state of each action a client has taken recently in the form of id:st  
17     "exclude_from_parse_flag": 1,  
18     "hash": 1792443391,  
19     "hidden_flag": 0,  
20     "id": 1,  
21     "ignore_case_flag": 1,  
22     "max_age_seconds": 3600,  
23     "name": "Action Statuses",  
24     "queries": {  
25         "_type": "queries",  
26         "query": [  
27 ..trimmed for brevity..
```

### Get all whitelisted urls

Get all whitelisted urls

### Example Python Code

```
1 import os  
2 import sys  
3 sys.dont_write_bytecode = True  
4  
5 # Determine our script name, script dir
```

```

6 my_file = os.path.abspath(sys.argv[0])
7 my_dir = os.path.dirname(my_file)
8
9 # determine the pytan lib dir and add it to the path
10 parent_dir = os.path.dirname(my_dir)
11 pytan_root_dir = os.path.dirname(parent_dir)
12 lib_dir = os.path.join(pytan_root_dir, 'lib')
13 path_adds = [lib_dir]
14
15 for aa in path_adds:
16     if aa not in sys.path:
17         sys.path.append(aa)
18
19
20 # connection info for Tanium Server
21 USERNAME = "Tanium User"
22 PASSWORD = "T@n!um"
23 HOST = "172.16.31.128"
24 PORT = "444"
25
26 # Logging conrols
27 LOGLEVEL = 2
28 DEBUGFORMAT = False
29
30 import tempfile
31
32 import pytan
33 handler = pytan.Handler(
34     username=USERNAME,
35     password=PASSWORD,
36     host=HOST,
37     port=PORT,
38     loglevel=LOGLEVEL,
39     debugformat=DEBUGFORMAT,
40 )
41
42 print handler
43
44 # setup the arguments for the handler method
45 kwargs = {}
46 kwargs["objtype"] = u'whitelisted_url'
47
48 # call the handler with the get_all method, passing in kwargs for arguments
49 response = handler.get_all(**kwargs)
50
51 print ""
52 print "Type of response: ", type(response)
53
54 print ""
55 print "print of response:"
56 print response
57
58 print ""
59 print "length of response (number of objects returned): "
60 print len(response)
61
62 print ""
63 print "print the first object returned in JSON format:"

```

```
64 out = response.to_json(response[0])
65 if len(out.splitlines()) > 15:
66     out = out.splitlines()[0:15]
67     out.append('..trimmed for brevity..')
68     out = '\n'.join(out)
69
70 print out
```

### Output from Python Code

```
1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3279
2
3 Type of response: <class 'taniumpy.object_types.white_listed_url_list.WhiteListedUrlList'>
4
5 print of response:
6 WhiteListedUrlList, len: 22
7
8 length of response (number of objects returned):
9 22
10
11 print the first object returned in JSON format:
12 {
13     "_type": "white_listed_url",
14     "download_seconds": 86400,
15     "id": 1,
16     "url_regex": "test1"
17 }
```

### Get all clients

Get all clients

### Example Python Code

```
1 import os
2 import sys
3 sys.dont_write_bytecode = True
4
5 # Determine our script name, script dir
6 my_file = os.path.abspath(sys.argv[0])
7 my_dir = os.path.dirname(my_file)
8
9 # determine the pytan lib dir and add it to the path
10 parent_dir = os.path.dirname(my_dir)
11 pytan_root_dir = os.path.dirname(parent_dir)
12 lib_dir = os.path.join(pytan_root_dir, 'lib')
13 path_adds = [lib_dir]
14
15 for aa in path_adds:
16     if aa not in sys.path:
17         sys.path.append(aa)
18
19
20 # connection info for Tanium Server
```

```

21 USERNAME = "Tanium User"
22 PASSWORD = "T@n!um"
23 HOST = "172.16.31.128"
24 PORT = "444"
25
26 # Logging conrols
27 LOGLEVEL = 2
28 DEBUGFORMAT = False
29
30 import tempfile
31
32 import pytan
33 handler = pytan.Handler(
34     username=USERNAME,
35     password=PASSWORD,
36     host=HOST,
37     port=PORT,
38     loglevel=LOGLEVEL,
39     debugformat=DEBUGFORMAT,
40 )
41
42 print handler
43
44 # setup the arguments for the handler method
45 kwargs = {}
46 kwargs["objtype"] = u'client'
47
48 # call the handler with the get_all method, passing in kwargs for arguments
49 response = handler.get_all(**kwargs)
50
51 print ""
52 print "Type of response: ", type(response)
53
54 print ""
55 print "print of response:"
56 print response
57
58 print ""
59 print "length of response (number of objects returned): "
60 print len(response)
61
62 print ""
63 print "print the first object returned in JSON format:"
64 out = response.to_json(response[0])
65 if len(out.splitlines()) > 15:
66     out = out.splitlines()[0:15]
67     out.append('..trimmed for brevity..')
68     out = '\n'.join(out)
69
70 print out

```

### Output from Python Code

```

1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3279
2
3 Type of response: <class 'taniumpy.object_types.system_status_list.SystemStatusList'>
4

```

```
5 print of response:
6 SystemStatusList, len: 5
7
8 length of response (number of objects returned):
9 5
10
11 print the first object returned in JSON format:
12 {
13     "_type": "client_status",
14     "cache_row_id": 0,
15     "computer_id": "1320430098",
16     "full_version": "5.1.314.7724",
17     "host_name": "Casus-Belli.local",
18     "ipaddress_client": "172.16.31.1",
19     "ipaddress_server": "172.16.31.1",
20     "last_registration": "2015-03-26T08:08:09",
21     "port_number": 17472,
22     "protocol_version": 314,
23     "send_state": "Forward Only",
24     "status": "Leader, Slow Link Behind"
25 }
```

## Get all packages

Get all packages

## Example Python Code

```
1 import os
2 import sys
3 sys.dont_write_bytecode = True
4
5 # Determine our script name, script dir
6 my_file = os.path.abspath(sys.argv[0])
7 my_dir = os.path.dirname(my_file)
8
9 # determine the pytan lib dir and add it to the path
10 parent_dir = os.path.dirname(my_dir)
11 pytan_root_dir = os.path.dirname(parent_dir)
12 lib_dir = os.path.join(pytan_root_dir, 'lib')
13 path_adds = [lib_dir]
14
15 for aa in path_adds:
16     if aa not in sys.path:
17         sys.path.append(aa)
18
19
20 # connection info for Tanium Server
21 USERNAME = "Tanium User"
22 PASSWORD = "T@n!um"
23 HOST = "172.16.31.128"
24 PORT = "444"
25
26 # Logging conrols
27 LOGLEVEL = 2
```

```

28 DEBUGFORMAT = False
29
30 import tempfile
31
32 import pytan
33 handler = pytan.Handler(
34     username=USERNAME,
35     password=PASSWORD,
36     host=HOST,
37     port=PORT,
38     loglevel=LOGLEVEL,
39     debugformat=DEBUGFORMAT,
40 )
41
42 print handler
43
44 # setup the arguments for the handler method
45 kwargs = {}
46 kwargs["objtype"] = u'package'
47
48 # call the handler with the get_all method, passing in kwargs for arguments
49 response = handler.get_all(**kwargs)
50
51 print ""
52 print "Type of response: ", type(response)
53
54 print ""
55 print "print of response:"
56 print response
57
58 print ""
59 print "length of response (number of objects returned): "
60 print len(response)
61
62 print ""
63 print "print the first object returned in JSON format:"
64 out = response.to_json(response[0])
65 if len(out.splitlines()) > 15:
66     out = out.splitlines()[0:15]
67     out.append('..trimmed for brevity..')
68     out = '\n'.join(out)
69
70 print out

```

### Output from Python Code

```

1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3279
2
3 Type of response: <class 'taniumpy.object_types.package_spec_list.PackageSpecList'>
4
5 print of response:
6 PackageSpecList, len: 224
7
8 length of response (number of objects returned):
9 224
10
11 print the first object returned in JSON format:

```

```
12 {
13     "_type": "package_spec",
14     "available_time": "2015-03-03T19:06:35",
15     "cache_row_id": 0,
16     "command": "cmd /c cscript //T:900 java-installer.vbs /KillAppsUsingJava:Yes /RebootIfNeeded:Yes /",
17     "command_timeout": 900,
18     "creation_time": "2015-03-03T19:03:39",
19     "deleted_flag": 0,
20     "display_name": "Update Java 64-bit - Kill / Reboot",
21     "expire_seconds": 1500,
22     "files": {
23         "_type": "package_files",
24         "file": [
25             {
26                 "_type": "file",
27                 ..trimmed for brevity..

```

### Get all actions

Get all actions

### Example Python Code

```
1  import os
2  import sys
3  sys.dont_write_bytecode = True
4
5  # Determine our script name, script dir
6  my_file = os.path.abspath(sys.argv[0])
7  my_dir = os.path.dirname(my_file)
8
9  # determine the pytan lib dir and add it to the path
10 parent_dir = os.path.dirname(my_dir)
11 pytan_root_dir = os.path.dirname(parent_dir)
12 lib_dir = os.path.join(pytan_root_dir, 'lib')
13 path_adds = [lib_dir]
14
15 for aa in path_adds:
16     if aa not in sys.path:
17         sys.path.append(aa)
18
19
20 # connection info for Tanium Server
21 USERNAME = "Tanium User"
22 PASSWORD = "T@n!um"
23 HOST = "172.16.31.128"
24 PORT = "444"
25
26 # Logging conrols
27 LOGLEVEL = 2
28 DEBUGFORMAT = False
29
30 import tempfile
31
32 import pytan

```



```

33 handler = pytan.Handler(
34     username=USERNAME,
35     password=PASSWORD,
36     host=HOST,
37     port=PORT,
38     loglevel=LOGLEVEL,
39     debugformat=DEBUGFORMAT,
40 )
41
42 print handler
43
44 # setup the arguments for the handler method
45 kwargs = {}
46 kwargs["objtype"] = u'action'
47
48 # call the handler with the get_all method, passing in kwargs for arguments
49 response = handler.get_all(**kwargs)
50
51 print ""
52 print "Type of response: ", type(response)
53
54 print ""
55 print "print of response:"
56 print response
57
58 print ""
59 print "length of response (number of objects returned): "
60 print len(response)
61
62 print ""
63 print "print the first object returned in JSON format:"
64 out = response.to_json(response[0])
65 if len(out.splitlines()) > 15:
66     out = out.splitlines()[0:15]
67     out.append('..trimmed for brevity..')
68     out = '\n'.join(out)
69
70 print out

```

### Output from Python Code

```

1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3279
2
3 Type of response: <class 'taniumpy.object_types.action_list.ActionList'>
4
5 print of response:
6 ActionList, len: 2025
7
8 length of response (number of objects returned):
9 2025
10
11 print the first object returned in JSON format:
12 {
13     "_type": "action",
14     "action_group": {
15         "_type": "group",
16         "id": 0,

```

```
17     "name": "Default"
18 },
19 "cache_row_id": 0,
20 "comment": "Scans for unmanaged assets on the network.",
21 "creation_time": "2015-03-03T19:05:56",
22 "distribute_seconds": 600,
23 "expiration_time": "2015-03-03T19:35:56",
24 "expire_seconds": 1800,
25 "history_saved_question": {
26     "_type": "saved_question",
27 ..trimmed for brevity..
```

## pytan API Invalid Get Object Examples

### Invalid get action single by name

Get an action by name (name is not a supported selector for action)

### Example Python Code

```
1  import os
2  import sys
3  sys.dont_write_bytecode = True
4
5  # Determine our script name, script dir
6  my_file = os.path.abspath(sys.argv[0])
7  my_dir = os.path.dirname(my_file)
8
9  # determine the pytan lib dir and add it to the path
10 parent_dir = os.path.dirname(my_dir)
11 pytan_root_dir = os.path.dirname(parent_dir)
12 lib_dir = os.path.join(pytan_root_dir, 'lib')
13 path_adds = [lib_dir]
14
15 for aa in path_adds:
16     if aa not in sys.path:
17         sys.path.append(aa)
18
19
20 # connection info for Tanium Server
21 USERNAME = "Tanium User"
22 PASSWORD = "T@n!um"
23 HOST = "172.16.31.128"
24 PORT = "444"
25
26 # Logging conrols
27 LOGLEVEL = 2
28 DEBUGFORMAT = False
29
30 import tempfile
31
32 import pytan
33 handler = pytan.Handler(
34     username=USERNAME,
35     password=PASSWORD,
```

```

36     host=HOST,
37     port=PORT,
38     loglevel=LOGLEVEL,
39     debugformat=DEBUGFORMAT,
40 )
41
42 print handler
43
44 # setup the arguments for the handler method
45 kwargs = {}
46 kwargs["objtype"] = u'action'
47 kwargs["name"] = u'Distribute Tanium Standard Utilities'
48
49
50 # call the handler with the get method, passing in kwargs for arguments
51 # this should throw an exception: pytan.utils.HandlerError
52 import traceback
53 try:
54     handler.get(**kwargs)
55 except Exception as e:
56     traceback.print_exc(file=sys.stdout)

```

### Output from Python Code

```

1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3279
2 Traceback (most recent call last):
3   File "<string>", line 55, in <module>
4   File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 1595, in get
5     raise HandlerError(err(objtype, api_attrs))
6 HandlerError: Getting a action requires at least one filter: ['id']

```

### Invalid get question by name

Get a question by name (name is not a supported selector for question)

### Example Python Code

```

1 import os
2 import sys
3 sys.dont_write_bytecode = True
4
5 # Determine our script name, script dir
6 my_file = os.path.abspath(sys.argv[0])
7 my_dir = os.path.dirname(my_file)
8
9 # determine the pytan lib dir and add it to the path
10 parent_dir = os.path.dirname(my_dir)
11 pytan_root_dir = os.path.dirname(parent_dir)
12 lib_dir = os.path.join(pytan_root_dir, 'lib')
13 path_adds = [lib_dir]
14
15 for aa in path_adds:
16     if aa not in sys.path:
17         sys.path.append(aa)

```

```
18
19
20 # connection info for Tanium Server
21 USERNAME = "Tanium User"
22 PASSWORD = "T@n!um"
23 HOST = "172.16.31.128"
24 PORT = "444"
25
26 # Logging conrols
27 LOGLEVEL = 2
28 DEBUGFORMAT = False
29
30 import tempfile
31
32 import pytan
33 handler = pytan.Handler(
34     username=USERNAME,
35     password=PASSWORD,
36     host=HOST,
37     port=PORT,
38     loglevel=LOGLEVEL,
39     debugformat=DEBUGFORMAT,
40 )
41
42 print handler
43
44 # setup the arguments for the handler method
45 kwargs = {}
46 kwargs["objtype"] = u'question'
47 kwargs["name"] = u'dweedle'
48
49
50 # call the handler with the get method, passing in kwargs for arguments
51 # this should throw an exception: pytan.utils.HandlerError
52 import traceback
53 try:
54     handler.get(**kwargs)
55 except Exception as e:
56     traceback.print_exc(file=sys.stdout)
```

### Output from Python Code

```
1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3279
2 Traceback (most recent call last):
3   File "<string>", line 55, in <module>
4   File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 1595, in get
5     raise HandlerError(err(objtype, api_attrs))
6 HandlerError: Getting a question requires at least one filter: ['id']
```

## pytan API Valid Deploy Action Examples

### Deploy action simple

Deploy an action against all computers using human strings.

## Example Python Code

```

1  import os
2  import sys
3  sys.dont_write_bytecode = True
4
5  # Determine our script name, script dir
6  my_file = os.path.abspath(sys.argv[0])
7  my_dir = os.path.dirname(my_file)
8
9  # determine the pytan lib dir and add it to the path
10 parent_dir = os.path.dirname(my_dir)
11 pytan_root_dir = os.path.dirname(parent_dir)
12 lib_dir = os.path.join(pytan_root_dir, 'lib')
13 path_adds = [lib_dir]
14
15 for aa in path_adds:
16     if aa not in sys.path:
17         sys.path.append(aa)
18
19
20 # connection info for Tanium Server
21 USERNAME = "Tanium User"
22 PASSWORD = "T@n!um"
23 HOST = "172.16.31.128"
24 PORT = "444"
25
26 # Logging conrols
27 LOGLEVEL = 2
28 DEBUGFORMAT = False
29
30 import tempfile
31
32 import pytan
33 handler = pytan.Handler(
34     username=USERNAME,
35     password=PASSWORD,
36     host=HOST,
37     port=PORT,
38     loglevel=LOGLEVEL,
39     debugformat=DEBUGFORMAT,
40 )
41
42 print handler
43
44 # setup the arguments for the handler method
45 kwargs = {}
46 kwargs["run"] = True
47 kwargs["package"] = u'Distribute Tanium Standard Utilities'
48
49 # call the handler with the deploy_action_human method, passing in kwargs for arguments
50 response = handler.deploy_action_human(**kwargs)
51 import pprint, io
52
53 print ""
54 print "Type of response: ", type(response)
55
56 print ""
57 print "Pretty print of response:"

```

```

58 print pprint.pformat(response)
59
60 print ""
61 print "Print of action object: "
62 print response['action_object']
63
64 # create an IO stream to store CSV results to
65 out = io.BytesIO()
66
67 # if results were returned (i.e. get_results=True was one of the kwargs passed in):
68 if response['action_results']:
69     # call the write_csv() method to convert response to CSV and store it in out
70     response['action_results'].write_csv(out, response['action_results'])
71
72     print ""
73     print "CSV Results of response: "
74     print out.getvalue()

```

### Output from Python Code

```

1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3279
2 2015-03-26 11:45:58,435 INFO question_progress: Results 0% (Get Online = "True" from all machine
3 2015-03-26 11:46:03,455 INFO question_progress: Results 100% (Get Online = "True" from all machi
4 2015-03-26 11:46:03,540 INFO action_progress: Action Results Passed: 0% (API Deploy Distribute T
5 2015-03-26 11:46:04,579 INFO action_progress: Action Results Passed: 0% (API Deploy Distribute T
6 2015-03-26 11:46:05,615 INFO action_progress: Action Results Passed: 0% (API Deploy Distribute T
7 2015-03-26 11:46:06,649 INFO action_progress: Action Results Passed: 0% (API Deploy Distribute T
8 2015-03-26 11:46:07,682 INFO action_progress: Action Results Passed: 0% (API Deploy Distribute T
9 2015-03-26 11:46:08,717 INFO action_progress: Action Results Passed: 0% (API Deploy Distribute T
10 2015-03-26 11:46:09,752 INFO action_progress: Action Results Passed: 0% (API Deploy Distribute T
11 2015-03-26 11:46:10,788 INFO action_progress: Action Results Passed: 0% (API Deploy Distribute T
12 2015-03-26 11:46:11,819 INFO action_progress: Action Results Passed: 0% (API Deploy Distribute T
13 2015-03-26 11:46:12,851 INFO action_progress: Action Results Passed: 0% (API Deploy Distribute T
14 2015-03-26 11:46:13,886 INFO action_progress: Action Results Passed: 0% (API Deploy Distribute T
15 2015-03-26 11:46:14,922 INFO action_progress: Action Results Passed: 0% (API Deploy Distribute T
16 2015-03-26 11:46:15,954 INFO action_progress: Action Results Passed: 0% (API Deploy Distribute T
17 2015-03-26 11:46:16,988 INFO action_progress: Action Results Passed: 0% (API Deploy Distribute T
18 2015-03-26 11:46:18,020 INFO action_progress: Action Results Passed: 0% (API Deploy Distribute T
19 2015-03-26 11:46:19,054 INFO action_progress: Action Results Passed: 0% (API Deploy Distribute T
20 2015-03-26 11:46:20,095 INFO action_progress: Action Results Passed: 0% (API Deploy Distribute T
21 2015-03-26 11:46:21,131 INFO action_progress: Action Results Passed: 0% (API Deploy Distribute T
22 2015-03-26 11:46:22,164 INFO action_progress: Action Results Passed: 0% (API Deploy Distribute T
23 2015-03-26 11:46:23,199 INFO action_progress: Action Results Passed: 0% (API Deploy Distribute T
24 2015-03-26 11:46:24,233 INFO action_progress: Action Results Passed: 0% (API Deploy Distribute T
25 2015-03-26 11:46:25,270 INFO action_progress: Action Results Passed: 0% (API Deploy Distribute T
26 2015-03-26 11:46:26,307 INFO action_progress: Action Results Passed: 0% (API Deploy Distribute T
27 2015-03-26 11:46:27,342 INFO action_progress: Action Results Passed: 0% (API Deploy Distribute T
28 2015-03-26 11:46:28,375 INFO action_progress: Action Results Passed: 0% (API Deploy Distribute T
29 2015-03-26 11:46:29,414 INFO action_progress: Action Results Passed: 0% (API Deploy Distribute T
30 2015-03-26 11:46:30,476 INFO action_progress: Action Results Passed: 0% (API Deploy Distribute T
31 2015-03-26 11:46:31,513 INFO action_progress: Action Results Passed: 0% (API Deploy Distribute T
32 2015-03-26 11:46:32,552 INFO action_progress: Action Results Passed: 0% (API Deploy Distribute T
33 2015-03-26 11:46:33,592 INFO action_progress: Action Results Passed: 0% (API Deploy Distribute T
34 2015-03-26 11:46:34,626 INFO action_progress: Action Results Passed: 0% (API Deploy Distribute T
35 2015-03-26 11:46:35,663 INFO action_progress: Action Results Passed: 0% (API Deploy Distribute T
36 2015-03-26 11:46:36,697 INFO action_progress: Action Results Passed: 0% (API Deploy Distribute T
37 2015-03-26 11:46:37,734 INFO action_progress: Action Results Passed: 0% (API Deploy Distribute T

```

38	2015-03-26 11:46:38,768	INFO	action_progress: Action Results Passed: 0%	(API Deploy Distribute T
39	2015-03-26 11:46:39,803	INFO	action_progress: Action Results Passed: 0%	(API Deploy Distribute T
40	2015-03-26 11:46:40,836	INFO	action_progress: Action Results Passed: 0%	(API Deploy Distribute T
41	2015-03-26 11:46:41,874	INFO	action_progress: Action Results Passed: 0%	(API Deploy Distribute T
42	2015-03-26 11:46:42,913	INFO	action_progress: Action Results Passed: 0%	(API Deploy Distribute T
43	2015-03-26 11:46:43,947	INFO	action_progress: Action Results Passed: 0%	(API Deploy Distribute T
44	2015-03-26 11:46:44,984	INFO	action_progress: Action Results Passed: 0%	(API Deploy Distribute T
45	2015-03-26 11:46:46,015	INFO	action_progress: Action Results Passed: 0%	(API Deploy Distribute T
46	2015-03-26 11:46:47,043	INFO	action_progress: Action Results Passed: 0%	(API Deploy Distribute T
47	2015-03-26 11:46:48,070	INFO	action_progress: Action Results Passed: 0%	(API Deploy Distribute T
48	2015-03-26 11:46:49,104	INFO	action_progress: Action Results Passed: 0%	(API Deploy Distribute T
49	2015-03-26 11:46:50,136	INFO	action_progress: Action Results Passed: 0%	(API Deploy Distribute T
50	2015-03-26 11:46:51,175	INFO	action_progress: Action Results Passed: 0%	(API Deploy Distribute T
51	2015-03-26 11:46:52,208	INFO	action_progress: Action Results Passed: 0%	(API Deploy Distribute T
52	2015-03-26 11:46:53,248	INFO	action_progress: Action Results Passed: 0%	(API Deploy Distribute T
53	2015-03-26 11:46:54,285	INFO	action_progress: Action Results Passed: 0%	(API Deploy Distribute T
54	2015-03-26 11:46:55,324	INFO	action_progress: Action Results Passed: 0%	(API Deploy Distribute T
55	2015-03-26 11:46:56,365	INFO	action_progress: Action Results Passed: 0%	(API Deploy Distribute T
56	2015-03-26 11:46:57,400	INFO	action_progress: Action Results Passed: 0%	(API Deploy Distribute T
57	2015-03-26 11:46:58,435	INFO	action_progress: Action Results Passed: 0%	(API Deploy Distribute T
58	2015-03-26 11:46:59,471	INFO	action_progress: Action Results Passed: 0%	(API Deploy Distribute T
59	2015-03-26 11:47:00,506	INFO	action_progress: Action Results Passed: 0%	(API Deploy Distribute T
60	2015-03-26 11:47:01,537	INFO	action_progress: Action Results Passed: 0%	(API Deploy Distribute T
61	2015-03-26 11:47:02,577	INFO	action_progress: Action Results Passed: 0%	(API Deploy Distribute T
62	2015-03-26 11:47:03,614	INFO	action_progress: Action Results Passed: 0%	(API Deploy Distribute T
63	2015-03-26 11:47:04,649	INFO	action_progress: Action Results Passed: 0%	(API Deploy Distribute T
64	2015-03-26 11:47:05,686	INFO	action_progress: Action Results Passed: 0%	(API Deploy Distribute T
65	2015-03-26 11:47:06,722	INFO	action_progress: Action Results Passed: 0%	(API Deploy Distribute T
66	2015-03-26 11:47:07,757	INFO	action_progress: Action Results Passed: 0%	(API Deploy Distribute T
67	2015-03-26 11:47:08,793	INFO	action_progress: Action Results Passed: 0%	(API Deploy Distribute T
68	2015-03-26 11:47:09,828	INFO	action_progress: Action Results Passed: 0%	(API Deploy Distribute T
69	2015-03-26 11:47:10,867	INFO	action_progress: Action Results Passed: 0%	(API Deploy Distribute T
70	2015-03-26 11:47:11,904	INFO	action_progress: Action Results Passed: 0%	(API Deploy Distribute T
71	2015-03-26 11:47:12,942	INFO	action_progress: Action Results Passed: 0%	(API Deploy Distribute T
72	2015-03-26 11:47:13,983	INFO	action_progress: Action Results Passed: 0%	(API Deploy Distribute T
73	2015-03-26 11:47:15,019	INFO	action_progress: Action Results Passed: 0%	(API Deploy Distribute T
74	2015-03-26 11:47:16,056	INFO	action_progress: Action Results Passed: 0%	(API Deploy Distribute T
75	2015-03-26 11:47:17,095	INFO	action_progress: Action Results Passed: 0%	(API Deploy Distribute T
76	2015-03-26 11:47:18,132	INFO	action_progress: Action Results Passed: 0%	(API Deploy Distribute T
77	2015-03-26 11:47:19,171	INFO	action_progress: Action Results Passed: 0%	(API Deploy Distribute T
78	2015-03-26 11:47:20,204	INFO	action_progress: Action Results Passed: 0%	(API Deploy Distribute T
79	2015-03-26 11:47:21,241	INFO	action_progress: Action Results Passed: 0%	(API Deploy Distribute T
80	2015-03-26 11:47:22,282	INFO	action_progress: Action Results Passed: 0%	(API Deploy Distribute T
81	2015-03-26 11:47:23,318	INFO	action_progress: Action Results Passed: 0%	(API Deploy Distribute T
82	2015-03-26 11:47:24,356	INFO	action_progress: Action Results Passed: 0%	(API Deploy Distribute T
83	2015-03-26 11:47:25,396	INFO	action_progress: Action Results Passed: 0%	(API Deploy Distribute T
84	2015-03-26 11:47:26,434	INFO	action_progress: Action Results Passed: 0%	(API Deploy Distribute T
85	2015-03-26 11:47:27,471	INFO	action_progress: Action Results Passed: 0%	(API Deploy Distribute T
86	2015-03-26 11:47:28,509	INFO	action_progress: Action Results Passed: 0%	(API Deploy Distribute T
87	2015-03-26 11:47:29,545	INFO	action_progress: Action Results Passed: 0%	(API Deploy Distribute T
88	2015-03-26 11:47:30,582	INFO	action_progress: Action Results Passed: 0%	(API Deploy Distribute T
89	2015-03-26 11:47:31,618	INFO	action_progress: Action Results Passed: 0%	(API Deploy Distribute T
90	2015-03-26 11:47:32,654	INFO	action_progress: Action Results Passed: 0%	(API Deploy Distribute T
91	2015-03-26 11:47:33,693	INFO	action_progress: Action Results Passed: 0%	(API Deploy Distribute T
92	2015-03-26 11:47:34,733	INFO	action_progress: Action Results Passed: 0%	(API Deploy Distribute T
93	2015-03-26 11:47:35,765	INFO	action_progress: Action Results Passed: 0%	(API Deploy Distribute T
94	2015-03-26 11:47:36,804	INFO	action_progress: Action Results Passed: 0%	(API Deploy Distribute T
95	2015-03-26 11:47:37,838	INFO	action_progress: Action Results Passed: 0%	(API Deploy Distribute T

```

96 2015-03-26 11:47:38,875 INFO action_progress: Action Results Passed: 0% (API Deploy Distribute T
97 2015-03-26 11:47:39,907 INFO action_progress: Action Results Passed: 0% (API Deploy Distribute T
98 2015-03-26 11:47:40,945 INFO action_progress: Action Results Passed: 0% (API Deploy Distribute T
99 2015-03-26 11:47:41,985 INFO action_progress: Action Results Passed: 0% (API Deploy Distribute T
100 2015-03-26 11:47:43,020 INFO action_progress: Action Results Passed: 0% (API Deploy Distribute T
101 2015-03-26 11:47:44,066 INFO action_progress: Action Results Passed: 0% (API Deploy Distribute T
102 2015-03-26 11:47:45,103 INFO action_progress: Action Results Passed: 0% (API Deploy Distribute T
103 2015-03-26 11:47:46,138 INFO action_progress: Action Results Passed: 0% (API Deploy Distribute T
104 2015-03-26 11:47:47,180 INFO action_progress: Action Results Passed: 0% (API Deploy Distribute T
105 2015-03-26 11:47:48,220 INFO action_progress: Action Results Passed: 0% (API Deploy Distribute T
106 2015-03-26 11:47:49,263 INFO action_progress: Action Results Passed: 50% (API Deploy Distribute
107 2015-03-26 11:47:50,300 INFO action_progress: Action Results Passed: 50% (API Deploy Distribute
108 2015-03-26 11:47:51,338 INFO action_progress: Action Results Passed: 50% (API Deploy Distribute
109 2015-03-26 11:47:52,379 INFO action_progress: Action Results Passed: 50% (API Deploy Distribute
110 2015-03-26 11:47:53,466 INFO action_progress: Action Results Passed: 100% (API Deploy Distribute
111 2015-03-26 11:47:53,499 INFO action_progress: Action Results Completed: 0% (API Deploy Distribut
112 2015-03-26 11:47:54,539 INFO action_progress: Action Results Completed: 0% (API Deploy Distribut
113 2015-03-26 11:47:55,581 INFO action_progress: Action Results Completed: 0% (API Deploy Distribut
114 2015-03-26 11:47:56,621 INFO action_progress: Action Results Completed: 0% (API Deploy Distribut
115 2015-03-26 11:47:57,666 INFO action_progress: Action Results Completed: 0% (API Deploy Distribut
116 2015-03-26 11:47:58,702 INFO action_progress: Action Results Completed: 0% (API Deploy Distribut
117 2015-03-26 11:47:59,739 INFO action_progress: Action Results Completed: 0% (API Deploy Distribut
118 2015-03-26 11:48:00,778 INFO action_progress: Action Results Completed: 0% (API Deploy Distribut
119 2015-03-26 11:48:01,819 INFO action_progress: Action Results Completed: 0% (API Deploy Distribut
120 2015-03-26 11:48:02,854 INFO action_progress: Action Results Completed: 0% (API Deploy Distribut
121 2015-03-26 11:48:03,895 INFO action_progress: Action Results Completed: 100% (API Deploy Distrib
122 2015-03-26 11:48:03,895 INFO action_progress: API Deploy Distribute Tanium Standard Utilities Re
123 Running Count: 0
124 Success Count: 2
125 Failed Count: 0
126 Unknown Count: 0
127 Finished Count: 2
128 Total Count: 2
129 Finished Count must equal: 2
130
131 Type of response: <type 'dict'>
132
133 Pretty print of response:
134 {'action_object': <taniumpy.object_types.action.Action object at 0x10856d890>,
135  'action_progress_human': 'API Deploy Distribute Tanium Standard Utilities Result Counts:\n\tRunning
136  'action_progress_map': {'Completed.': ['Casus-Belli.local',
137                                     'jtanium1.localdomain']},
138  'action_results': <taniumpy.object_types.result_set.ResultSet object at 0x10769cfd0>,
139  'pre_action_question_results': {'question_object': <taniumpy.object_types.question.Question object
140                                     'question_results': <taniumpy.object_types.result_set.ResultSet obj
141
142 Print of action object:
143 Action, name: 'API Deploy Distribute Tanium Standard Utilities'
144
145 CSV Results of response:
146 Action Statuses,Computer Name
147 21076:Completed.,Casus-Belli.local
148 21076:Completed.,jtanium1.localdomain

```



## Deploy action simple without results

Deploy an action against all computers using human strings, but do not get the completed results of the job – return right away with the deploy action object.

## Example Python Code

```

1  import os
2  import sys
3  sys.dont_write_bytecode = True
4
5  # Determine our script name, script dir
6  my_file = os.path.abspath(sys.argv[0])
7  my_dir = os.path.dirname(my_file)
8
9  # determine the pytan lib dir and add it to the path
10 parent_dir = os.path.dirname(my_dir)
11 pytan_root_dir = os.path.dirname(parent_dir)
12 lib_dir = os.path.join(pytan_root_dir, 'lib')
13 path_adds = [lib_dir]
14
15 for aa in path_adds:
16     if aa not in sys.path:
17         sys.path.append(aa)
18
19
20 # connection info for Tanium Server
21 USERNAME = "Tanium User"
22 PASSWORD = "T@n!um"
23 HOST = "172.16.31.128"
24 PORT = "444"
25
26 # Logging conrols
27 LOGLEVEL = 2
28 DEBUGFORMAT = False
29
30 import tempfile
31
32 import pytan
33 handler = pytan.Handler(
34     username=USERNAME,
35     password=PASSWORD,
36     host=HOST,
37     port=PORT,
38     loglevel=LOGLEVEL,
39     debugformat=DEBUGFORMAT,
40 )
41
42 print handler
43
44 # setup the arguments for the handler method
45 kwargs = {}
46 kwargs["get_results"] = False
47 kwargs["run"] = True
48 kwargs["package"] = u'Distribute Tanium Standard Utilities'
49
50 # call the handler with the deploy_action_human method, passing in kwargs for arguments

```

```
51 response = handler.deploy_action_human(**kwargs)
52 import pprint, io
53
54 print ""
55 print "Type of response: ", type(response)
56
57 print ""
58 print "Pretty print of response:"
59 print pprint.pformat(response)
60
61 print ""
62 print "Print of action object: "
63 print response['action_object']
64
65 # create an IO stream to store CSV results to
66 out = io.BytesIO()
67
68 # if results were returned (i.e. get_results=True was one of the kwargs passed in):
69 if response['action_results']:
70     # call the write_csv() method to convert response to CSV and store it in out
71     response['action_results'].write_csv(out, response['action_results'])
72
73     print ""
74     print "CSV Results of response: "
75     print out.getvalue()
```

### Output from Python Code

```
1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3279
2
3 Type of response: <type 'dict'>
4
5 Pretty print of response:
6 {'action_object': <taniumpy.object_types.action.Action object at 0x107808f10>,
7  'action_progress_human': None,
8  'action_progress_map': None,
9  'action_results': None,
10 'pre_action_question_results': None}
11
12 Print of action object:
13 Action, name: 'API Deploy Distribute Tanium Standard Utilities'
```

### Deploy action simple against windows computers

Deploy an action against only windows computers using human strings. This requires passing in an action filter

### Example Python Code

```
1 import os
2 import sys
3 sys.dont_write_bytecode = True
4
5 # Determine our script name, script dir
6 my_file = os.path.abspath(sys.argv[0])
```

```

7 my_dir = os.path.dirname(my_file)
8
9 # determine the pytan lib dir and add it to the path
10 parent_dir = os.path.dirname(my_dir)
11 pytan_root_dir = os.path.dirname(parent_dir)
12 lib_dir = os.path.join(pytan_root_dir, 'lib')
13 path_adds = [lib_dir]
14
15 for aa in path_adds:
16     if aa not in sys.path:
17         sys.path.append(aa)
18
19
20 # connection info for Tanium Server
21 USERNAME = "Tanium User"
22 PASSWORD = "T@n!um"
23 HOST = "172.16.31.128"
24 PORT = "444"
25
26 # Logging conrols
27 LOGLEVEL = 2
28 DEBUGFORMAT = False
29
30 import tempfile
31
32 import pytan
33 handler = pytan.Handler(
34     username=USERNAME,
35     password=PASSWORD,
36     host=HOST,
37     port=PORT,
38     loglevel=LOGLEVEL,
39     debugformat=DEBUGFORMAT,
40 )
41
42 print handler
43
44 # setup the arguments for the handler method
45 kwargs = {}
46 kwargs["run"] = True
47 kwargs["action_filters"] = u'Operating System, that contains:Windows'
48 kwargs["package"] = u'Distribute Tanium Standard Utilities'
49
50 # call the handler with the deploy_action_human method, passing in kwargs for arguments
51 response = handler.deploy_action_human(**kwargs)
52 import pprint, io
53
54 print ""
55 print "Type of response: ", type(response)
56
57 print ""
58 print "Pretty print of response:"
59 print pprint.pformat(response)
60
61 print ""
62 print "Print of action object: "
63 print response['action_object']
64

```

```

65 # create an IO stream to store CSV results to
66 out = io.BytesIO()
67
68 # if results were returned (i.e. get_results=True was one of the kwargs passed in):
69 if response['action_results']:
70     # call the write_csv() method to convert response to CSV and store it in out
71     response['action_results'].write_csv(out, response['action_results'])
72
73     print ""
74     print "CSV Results of response: "
75     print out.getvalue()

```

### Output from Python Code

```

1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3279
2 2015-03-26 11:48:04,109 INFO question_progress: Results 0% (Get Online = "True" from all machine
3 2015-03-26 11:48:09,130 INFO question_progress: Results 100% (Get Online = "True" from all machine
4 2015-03-26 11:48:09,233 INFO action_progress: Action Results Passed: 0% (API Deploy Distribute T
5 2015-03-26 11:48:10,272 INFO action_progress: Action Results Passed: 0% (API Deploy Distribute T
6 2015-03-26 11:48:11,310 INFO action_progress: Action Results Passed: 0% (API Deploy Distribute T
7 2015-03-26 11:48:12,348 INFO action_progress: Action Results Passed: 0% (API Deploy Distribute T
8 2015-03-26 11:48:13,396 INFO action_progress: Action Results Passed: 0% (API Deploy Distribute T
9 2015-03-26 11:48:14,439 INFO action_progress: Action Results Passed: 0% (API Deploy Distribute T
10 2015-03-26 11:48:15,482 INFO action_progress: Action Results Passed: 0% (API Deploy Distribute T
11 2015-03-26 11:48:16,525 INFO action_progress: Action Results Passed: 0% (API Deploy Distribute T
12 2015-03-26 11:48:17,562 INFO action_progress: Action Results Passed: 0% (API Deploy Distribute T
13 2015-03-26 11:48:18,613 INFO action_progress: Action Results Passed: 0% (API Deploy Distribute T
14 2015-03-26 11:48:19,657 INFO action_progress: Action Results Passed: 0% (API Deploy Distribute T
15 2015-03-26 11:48:20,695 INFO action_progress: Action Results Passed: 100% (API Deploy Distribute
16 2015-03-26 11:48:20,733 INFO action_progress: Action Results Completed: 0% (API Deploy Distribut
17 2015-03-26 11:48:21,772 INFO action_progress: Action Results Completed: 0% (API Deploy Distribut
18 2015-03-26 11:48:22,807 INFO action_progress: Action Results Completed: 0% (API Deploy Distribut
19 2015-03-26 11:48:23,849 INFO action_progress: Action Results Completed: 0% (API Deploy Distribut
20 2015-03-26 11:48:24,887 INFO action_progress: Action Results Completed: 0% (API Deploy Distribut
21 2015-03-26 11:48:25,927 INFO action_progress: Action Results Completed: 0% (API Deploy Distribut
22 2015-03-26 11:48:26,965 INFO action_progress: Action Results Completed: 0% (API Deploy Distribut
23 2015-03-26 11:48:28,004 INFO action_progress: Action Results Completed: 0% (API Deploy Distribut
24 2015-03-26 11:48:29,042 INFO action_progress: Action Results Completed: 0% (API Deploy Distribut
25 2015-03-26 11:48:30,077 INFO action_progress: Action Results Completed: 0% (API Deploy Distribut
26 2015-03-26 11:48:31,105 INFO action_progress: Action Results Completed: 0% (API Deploy Distribut
27 2015-03-26 11:48:32,140 INFO action_progress: Action Results Completed: 0% (API Deploy Distribut
28 2015-03-26 11:48:33,178 INFO action_progress: Action Results Completed: 0% (API Deploy Distribut
29 2015-03-26 11:48:34,216 INFO action_progress: Action Results Completed: 0% (API Deploy Distribut
30 2015-03-26 11:48:35,252 INFO action_progress: Action Results Completed: 0% (API Deploy Distribut
31 2015-03-26 11:48:36,290 INFO action_progress: Action Results Completed: 100% (API Deploy Distrib
32 2015-03-26 11:48:36,290 INFO action_progress: API Deploy Distribute Tanium Standard Utilities Re
33 Running Count: 0
34 Success Count: 1
35 Failed Count: 0
36 Unknown Count: 0
37 Finished Count: 1
38 Total Count: 1
39 Finished Count must equal: 1
40
41 Type of response: <type 'dict'>
42
43 Pretty print of response:

```

```

44 {'action_object': <taniumpy.object_types.action.Action object at 0x108567450>,
45  'action_progress_human': 'API Deploy Distribute Tanium Standard Utilities Result Counts:\n\tRunning
46  'action_progress_map': {'Completed.': ['jtanium1.localdomain']},
47  'action_results': <taniumpy.object_types.result_set.ResultSet object at 0x1077fa950>,
48  'pre_action_question_results': {'question_object': <taniumpy.object_types.question.Question object
49                                  'question_results': <taniumpy.object_types.result_set.ResultSet obj
50
51 Print of action object:
52 Action, name: 'API Deploy Distribute Tanium Standard Utilities'
53
54 CSV Results of response:
55 Action Statuses,Computer Name
56 21078:Completed.,jtanium1.localdomain

```

### Deploy action with params against windows computers

Deploy an action with parameters against only windows computers using human strings.

This will use the Package ‘Custom Tagging - Add Tags’ and supply two parameters. The second parameter will be ignored because the package in question only requires one parameter.

### Example Python Code

```

1  import os
2  import sys
3  sys.dont_write_bytecode = True
4
5  # Determine our script name, script dir
6  my_file = os.path.abspath(sys.argv[0])
7  my_dir = os.path.dirname(my_file)
8
9  # determine the pytan lib dir and add it to the path
10 parent_dir = os.path.dirname(my_dir)
11 pytan_root_dir = os.path.dirname(parent_dir)
12 lib_dir = os.path.join(pytan_root_dir, 'lib')
13 path_adds = [lib_dir]
14
15 for aa in path_adds:
16     if aa not in sys.path:
17         sys.path.append(aa)
18
19
20 # connection info for Tanium Server
21 USERNAME = "Tanium User"
22 PASSWORD = "T@n!um"
23 HOST = "172.16.31.128"
24 PORT = "444"
25
26 # Logging conrols
27 LOGLEVEL = 2
28 DEBUGFORMAT = False
29
30 import tempfile
31
32 import pytan
33 handler = pytan.Handler(

```

```

34     username=USERNAME,
35     password=PASSWORD,
36     host=HOST,
37     port=PORT,
38     loglevel=LOGLEVEL,
39     debugformat=DEBUGFORMAT,
40 )
41
42 print handler
43
44 # setup the arguments for the handler method
45 kwargs = {}
46 kwargs["run"] = True
47 kwargs["action_filters"] = u'Operating System, that contains:Windows'
48 kwargs["package"] = u'Custom Tagging - Add Tags{$1=tag_should_be_added,$2=tag_should_be_ignore}'
49
50 # call the handler with the deploy_action_human method, passing in kwargs for arguments
51 response = handler.deploy_action_human(**kwargs)
52 import pprint, io
53
54 print ""
55 print "Type of response: ", type(response)
56
57 print ""
58 print "Pretty print of response:"
59 print pprint.pformat(response)
60
61 print ""
62 print "Print of action object: "
63 print response['action_object']
64
65 # create an IO stream to store CSV results to
66 out = io.BytesIO()
67
68 # if results were returned (i.e. get_results=True was one of the kwargs passed in):
69 if response['action_results']:
70     # call the write_csv() method to convert response to CSV and store it in out
71     response['action_results'].write_csv(out, response['action_results'])
72
73     print ""
74     print "CSV Results of response: "
75     print out.getvalue()

```

### Output from Python Code

```

1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3279
2 2015-03-26 11:48:36,436 INFO      question_progress: Results 0% (Get Online = "True" from all machine
3 2015-03-26 11:48:41,456 INFO      question_progress: Results 50% (Get Online = "True" from all machine
4 2015-03-26 11:48:46,478 INFO      question_progress: Results 100% (Get Online = "True" from all machine
5 2015-03-26 11:48:46,628 INFO      action_progress: Action Results Passed: 0% (API Deploy Custom Taggi
6 2015-03-26 11:48:47,667 INFO      action_progress: Action Results Passed: 0% (API Deploy Custom Taggi
7 2015-03-26 11:48:48,712 INFO      action_progress: Action Results Passed: 0% (API Deploy Custom Taggi
8 2015-03-26 11:48:49,747 INFO      action_progress: Action Results Passed: 0% (API Deploy Custom Taggi
9 2015-03-26 11:48:50,784 INFO      action_progress: Action Results Passed: 0% (API Deploy Custom Taggi
10 2015-03-26 11:48:51,830 INFO      action_progress: Action Results Passed: 0% (API Deploy Custom Taggi
11 2015-03-26 11:48:52,867 INFO      action_progress: Action Results Passed: 0% (API Deploy Custom Taggi
12 2015-03-26 11:48:53,907 INFO      action_progress: Action Results Passed: 0% (API Deploy Custom Taggi

```

```

13 2015-03-26 11:48:54,946 INFO      action_progress: Action Results Passed: 0% (API Deploy Custom Taggi
14 2015-03-26 11:48:55,985 INFO      action_progress: Action Results Passed: 0% (API Deploy Custom Taggi
15 2015-03-26 11:48:57,023 INFO      action_progress: Action Results Passed: 0% (API Deploy Custom Taggi
16 2015-03-26 11:48:58,064 INFO      action_progress: Action Results Passed: 100% (API Deploy Custom Tag
17 2015-03-26 11:48:58,099 INFO      action_progress: Action Results Completed: 100% (API Deploy Custom
18 2015-03-26 11:48:58,100 INFO      action_progress: API Deploy Custom Tagging - Add Tags Result Counts
19      Running Count: 0
20      Success Count: 1
21      Failed Count: 0
22      Unknown Count: 0
23      Finished Count: 1
24      Total Count: 1
25      Finished Count must equal: 1
26
27 Type of response: <type 'dict'>
28
29 Pretty print of response:
30 {'action_object': <taniumpy.object_types.action.Action object at 0x10756c410>,
31  'action_progress_human': 'API Deploy Custom Tagging - Add Tags Result Counts:\n\tRunning Count: 0\n\t
32  'action_progress_map': {'Completed.': ['jtanium1.localdomain']},
33  'action_results': <taniumpy.object_types.result_set.ResultSet object at 0x107563bd0>,
34  'pre_action_question_results': {'question_object': <taniumpy.object_types.question.Question object
35                                'question_results': <taniumpy.object_types.result_set.ResultSet obj
36
37 Print of action object:
38 Action, name: 'API Deploy Custom Tagging - Add Tags'
39
40 CSV Results of response:
41 Action Statuses,Computer Name
42 21079:Completed.,jtanium1.localdomain

```

## pytan API Invalid Deploy Action Examples

### Invalid deploy action run false

Deploy an action without `run=True`, which will only run the pre-deploy action question that matches `action_filters`, export the results to a file, and raise a `RunFalse` exception

### Example Python Code

```

1  import os
2  import sys
3  sys.dont_write_bytecode = True
4
5  # Determine our script name, script dir
6  my_file = os.path.abspath(sys.argv[0])
7  my_dir = os.path.dirname(my_file)
8
9  # determine the pytan lib dir and add it to the path
10 parent_dir = os.path.dirname(my_dir)
11 pytan_root_dir = os.path.dirname(parent_dir)
12 lib_dir = os.path.join(pytan_root_dir, 'lib')
13 path_adds = [lib_dir]
14
15 for aa in path_adds:

```

```
16     if aa not in sys.path:
17         sys.path.append(aa)
18
19
20     # connection info for Tanium Server
21     USERNAME = "Tanium User"
22     PASSWORD = "T@n!um"
23     HOST = "172.16.31.128"
24     PORT = "444"
25
26     # Logging conrols
27     LOGLEVEL = 2
28     DEBUGFORMAT = False
29
30     import tempfile
31
32     import pytan
33     handler = pytan.Handler(
34         username=USERNAME,
35         password=PASSWORD,
36         host=HOST,
37         port=PORT,
38         loglevel=LOGLEVEL,
39         debugformat=DEBUGFORMAT,
40     )
41
42     print handler
43
44     # setup the arguments for the handler method
45     kwargs = {}
46     kwargs['report_dir'] = tempfile.gettempdir()
47     kwargs["package"] = u'Distribute Tanium Standard Utilities'
48
49
50     # call the handler with the deploy_action_human method, passing in kwargs for arguments
51     # this should throw an exception: pytan.utils.RunFalse
52     import traceback
53     try:
54         handler.deploy_action_human(**kwargs)
55     except Exception as e:
56         traceback.print_exc(file=sys.stdout)
```

### Output from Python Code

```
1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3279
2 2015-03-26 11:48:58,305 INFO      question_progress: Results 0% (Get Computer Name and Online = "True
3 2015-03-26 11:49:03,333 INFO      question_progress: Results 50% (Get Computer Name and Online = "True
4 2015-03-26 11:49:08,354 INFO      question_progress: Results 100% (Get Computer Name and Online = "True
5 2015-03-26 11:49:08,376 INFO      handler: Report file '/var/folders/dk/vjrlr_c53yx6k6gzp2bbt_c40000gn
6 Traceback (most recent call last):
7   File "<string>", line 55, in <module>
8   File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 1193, in deploy_action_human
9     **kwargs
10  File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 1034, in deploy_action
11    raise RunFalse(m(report_path, len(result)))
12 RunFalse: 'Run' is not True!!
13 View and verify the contents of /var/folders/dk/vjrlr_c53yx6k6gzp2bbt_c40000gn/T/VERIFY_BEFORE_DEPLO
```



14 | Re-run this deploy action with run=True after verifying

### Invalid deploy action package help

Have deploy\_action\_human() return the help for package

### Example Python Code

```

1  import os
2  import sys
3  sys.dont_write_bytecode = True
4
5  # Determine our script name, script dir
6  my_file = os.path.abspath(sys.argv[0])
7  my_dir = os.path.dirname(my_file)
8
9  # determine the pytan lib dir and add it to the path
10 parent_dir = os.path.dirname(my_dir)
11 pytan_root_dir = os.path.dirname(parent_dir)
12 lib_dir = os.path.join(pytan_root_dir, 'lib')
13 path_adds = [lib_dir]
14
15 for aa in path_adds:
16     if aa not in sys.path:
17         sys.path.append(aa)
18
19
20 # connection info for Tanium Server
21 USERNAME = "Tanium User"
22 PASSWORD = "T@n!um"
23 HOST = "172.16.31.128"
24 PORT = "444"
25
26 # Logging conrols
27 LOGLEVEL = 2
28 DEBUGFORMAT = False
29
30 import tempfile
31
32 import pytan
33 handler = pytan.Handler(
34     username=USERNAME,
35     password=PASSWORD,
36     host=HOST,
37     port=PORT,
38     loglevel=LOGLEVEL,
39     debugformat=DEBUGFORMAT,
40 )
41
42 print handler
43
44 # setup the arguments for the handler method
45 kwargs = {}
46 kwargs['report_dir'] = tempfile.gettempdir()
47 kwargs["package_help"] = True

```

```
48
49
50 # call the handler with the deploy_action_human method, passing in kwargs for arguments
51 # this should throw an exception: pytan.utils.PytanHelp
52 import traceback
53 try:
54     handler.deploy_action_human(**kwargs)
55 except Exception as e:
56     traceback.print_exc(file=sys.stdout)
```

## Output from Python Code

```
1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3279
2 Traceback (most recent call last):
3   File "<string>", line 55, in <module>
4   File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 1157, in deploy_action_human
5     raise PytanHelp(utils.help_package())
6 PytanHelp:
7 Package Help
8 =====
9
10 Supplying package defines what package will be deployed as part of the
11 action.
12
13 A package string is a human string that describes, at a minimum, a
14 package. It can also optionally define a selector for the package,
15 and/or parameters for the package. A package must be provided as a string.
16
17 Examples for package
18 -----
19
20 Supplying a package:
21
22     'Distribute Tanium Standard Utilities'
23
24 Supplying a package by id:
25
26     'id:1'
27
28 Supplying a package by hash:
29
30     'hash:123456789'
31
32 Supplying a package by name:
33
34     'name:Distribute Tanium Standard Utilities'
35
36 Package Parameters
37 -----
38
39 Supplying parameters to a package can control the arguments
40 that are supplied to a package, if that package takes any arguments.
41
42 Package parameters must be surrounded with curly braces '{}',
43 and must have a key and value specified that is separated by
44 an equals '='. Multiple parameters must be seperated by
45 a comma ','. The key should match up to a valid parameter key
```

```

46 for the package in question.
47
48 If a parameter is supplied and the package doesn't have a
49 corresponding key name, it will be ignored. If the package has
50 parameters and a parameter is NOT supplied then an exception
51 will be raised, printing out the JSON of the missing parameter
52 for the package in question.
53
54 Examples for package with parameters
55 -----
56
57 Supplying a package with a single parameter '$1':
58
59     'Package With Params{$1=value1}'
60
61 Supplying a package with two parameters, '$1' and '$2':
62
63     'Package With Params{$1=value1,$2=value2}'

```

### Invalid deploy action package

Deploy an action using a non-existing package.

### Example Python Code

```

1  import os
2  import sys
3  sys.dont_write_bytecode = True
4
5  # Determine our script name, script dir
6  my_file = os.path.abspath(sys.argv[0])
7  my_dir = os.path.dirname(my_file)
8
9  # determine the pytan lib dir and add it to the path
10 parent_dir = os.path.dirname(my_dir)
11 pytan_root_dir = os.path.dirname(parent_dir)
12 lib_dir = os.path.join(pytan_root_dir, 'lib')
13 path_adds = [lib_dir]
14
15 for aa in path_adds:
16     if aa not in sys.path:
17         sys.path.append(aa)
18
19
20 # connection info for Tanium Server
21 USERNAME = "Tanium User"
22 PASSWORD = "T@n!um"
23 HOST = "172.16.31.128"
24 PORT = "444"
25
26 # Logging controls
27 LOGLEVEL = 2
28 DEBUGFORMAT = False
29
30 import tempfile

```

```
31
32 import pytan
33 handler = pytan.Handler(
34     username=USERNAME,
35     password=PASSWORD,
36     host=HOST,
37     port=PORT,
38     loglevel=LOGLEVEL,
39     debugformat=DEBUGFORMAT,
40 )
41
42 print handler
43
44 # setup the arguments for the handler method
45 kwargs = {}
46 kwargs['report_dir'] = tempfile.gettempdir()
47 kwargs["run"] = True
48 kwargs["package"] = u'Invalid Package'
49
50
51 # call the handler with the deploy_action_human method, passing in kwargs for arguments
52 # this should throw an exception: pytan.utils.HandlerError
53 import traceback
54 try:
55     handler.deploy_action_human(**kwargs)
56 except Exception as e:
57     traceback.print_exc(file=sys.stdout)
```

### Output from Python Code

```
1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3279
2 Traceback (most recent call last):
3   File "<string>", line 56, in <module>
4   File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 1193, in deploy_action_human
5     **kwargs
6   File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 971, in deploy_action
7     package_def = self._get_package_def(package_def)
8   File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 1829, in _get_package_def
9     d['package_obj'] = self.get('package', **def_search)[0]
10  File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 1605, in get
11    return self._get_single(obj_map, **kwargs)
12  File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 1789, in _get_single
13    for x in self._single_find(obj_map, k, v, **kwargs):
14  File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 1799, in _single_find
15    obj_ret = self._find(api_obj_single, **kwargs)
16  File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 1727, in _find
17    raise HandlerError(err(search_str))
18 HandlerError: No results found searching for PackageSpec, name: u'Invalid Package'!!
```

### Invalid deploy action options help

Have `deploy_action_human()` return the help for options

### Example Python Code

```

1  import os
2  import sys
3  sys.dont_write_bytecode = True
4
5  # Determine our script name, script dir
6  my_file = os.path.abspath(sys.argv[0])
7  my_dir = os.path.dirname(my_file)
8
9  # determine the pytan lib dir and add it to the path
10 parent_dir = os.path.dirname(my_dir)
11 pytan_root_dir = os.path.dirname(parent_dir)
12 lib_dir = os.path.join(pytan_root_dir, 'lib')
13 path_adds = [lib_dir]
14
15 for aa in path_adds:
16     if aa not in sys.path:
17         sys.path.append(aa)
18
19
20 # connection info for Tanium Server
21 USERNAME = "Tanium User"
22 PASSWORD = "T@n!um"
23 HOST = "172.16.31.128"
24 PORT = "444"
25
26 # Logging conrols
27 LOGLEVEL = 2
28 DEBUGFORMAT = False
29
30 import tempfile
31
32 import pytan
33 handler = pytan.Handler(
34     username=USERNAME,
35     password=PASSWORD,
36     host=HOST,
37     port=PORT,
38     loglevel=LOGLEVEL,
39     debugformat=DEBUGFORMAT,
40 )
41
42 print handler
43
44 # setup the arguments for the handler method
45 kwargs = {}
46 kwargs['report_dir'] = tempfile.gettempdir()
47 kwargs["options_help"] = True
48
49
50 # call the handler with the deploy_action_human method, passing in kwargs for arguments
51 # this should throw an exception: pytan.utils.PytanHelp
52 import traceback
53 try:
54     handler.deploy_action_human(**kwargs)
55 except Exception as e:
56     traceback.print_exc(file=sys.stdout)

```

## Output from Python Code

```
1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3279
2 Traceback (most recent call last):
3   File "<string>", line 55, in <module>
4   File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 1163, in deploy_action_human
5     raise PytanHelp(utils.help_options())
6 PytanHelp:
7 Options Help
8 =====
9
10 Options are used for controlling how filters act. When options are
11 used as part of a sensor string, they change how the filters
12 supplied as part of that sensor operate. When options are used for
13 whole question options, they change how all of the question filters
14 operate.
15
16 When options are supplied for a sensor string, they must be
17 supplied as ', opt:OPTION' or ', opt:OPTION:VALUE' for options
18 that require a value.
19
20 When options are supplied for question options, they must be
21 supplied as 'OPTION' or 'OPTION:VALUE' for options that require
22 a value.
23
24 Options can be used on 'filter' or 'group', where 'group' pertains
25 to group filters or question filters. All 'filter' options are also
26 applicable to 'group' for question options.
27
28 Valid Options
29 -----
30
31 'ignore_case'
32   Help: Make the filter do a case insensitive match
33   Usable on: filter
34   Example for sensor: "Sensor1, opt:ignore_case"
35   Example for question: "ignore_case"
36
37 'match_case'
38   Help: Make the filter do a case sensitive match
39   Usable on: filter
40   Example for sensor: "Sensor1, opt:match_case"
41   Example for question: "match_case"
42
43 'match_any_value'
44   Help: Make the filter match any value
45   Usable on: filter
46   Example for sensor: "Sensor1, opt:match_any_value"
47   Example for question: "match_any_value"
48
49 'match_all_values'
50   Help: Make the filter match all values
51   Usable on: filter
52   Example for sensor: "Sensor1, opt:match_all_values"
53   Example for question: "match_all_values"
54
55 'max_data_age'
56   Help: Re-fetch cached values older than N seconds
57   Usable on: filter
```

```

58     VALUE description and type: seconds, <type 'int'>
59     Example for sensor: "Sensor1, opt:max_data_age:seconds"
60     Example for question: "max_data_age:seconds"
61
62     'value_type'
63     Help: Make the filter consider the value type as VALUE_TYPE
64     Usable on: filter
65     VALUE description and type: value_type, <type 'str'>
66     Example for sensor: "Sensor1, opt:value_type:value_type"
67     Example for question: "value_type:value_type"
68
69     'and'
70     Help: Use 'and' for all of the filters supplied
71     Usable on: group
72     Example for sensor: "Sensor1, opt:and"
73     Example for question: "and"
74
75     'or'
76     Help: Use 'or' for all of the filters supplied
77     Usable on: group
78     Example for sensor: "Sensor1, opt:or"
79     Example for question: "or"
80
81     'ignore_case'
82     Help: Make the filter do a case insensitive match
83     Usable on: filter
84     Example for sensor: "Sensor1, opt:ignore_case"
85     Example for question: "ignore_case"
86
87     'match_case'
88     Help: Make the filter do a case sensitive match
89     Usable on: filter
90     Example for sensor: "Sensor1, opt:match_case"
91     Example for question: "match_case"
92
93     'match_any_value'
94     Help: Make the filter match any value
95     Usable on: filter
96     Example for sensor: "Sensor1, opt:match_any_value"
97     Example for question: "match_any_value"
98
99     'match_all_values'
100    Help: Make the filter match all values
101    Usable on: filter
102    Example for sensor: "Sensor1, opt:match_all_values"
103    Example for question: "match_all_values"
104
105    'max_data_age'
106    Help: Re-fetch cached values older than N seconds
107    Usable on: filter
108    VALUE description and type: seconds, <type 'int'>
109    Example for sensor: "Sensor1, opt:max_data_age:seconds"
110    Example for question: "max_data_age:seconds"
111
112    'value_type'
113    Help: Make the filter consider the value type as VALUE_TYPE
114    Usable on: filter
115    VALUE description and type: value_type, <type 'str'>

```

```
116         Example for sensor: "Sensor1, opt:value_type:value_type"
117         Example for question: "value_type:value_type"
118
119     'and'
120     Help: Use 'and' for all of the filters supplied
121     Usable on: group
122     Example for sensor: "Sensor1, opt:and"
123     Example for question: "and"
124
125     'or'
126     Help: Use 'or' for all of the filters supplied
127     Usable on: group
128     Example for sensor: "Sensor1, opt:or"
129     Example for question: "or"
```

### Invalid deploy action empty package

Deploy an action using an empty package string.

### Example Python Code

```
1  import os
2  import sys
3  sys.dont_write_bytecode = True
4
5  # Determine our script name, script dir
6  my_file = os.path.abspath(sys.argv[0])
7  my_dir = os.path.dirname(my_file)
8
9  # determine the pytan lib dir and add it to the path
10 parent_dir = os.path.dirname(my_dir)
11 pytan_root_dir = os.path.dirname(parent_dir)
12 lib_dir = os.path.join(pytan_root_dir, 'lib')
13 path_adds = [lib_dir]
14
15 for aa in path_adds:
16     if aa not in sys.path:
17         sys.path.append(aa)
18
19
20 # connection info for Tanium Server
21 USERNAME = "Tanium User"
22 PASSWORD = "T@n!um"
23 HOST = "172.16.31.128"
24 PORT = "444"
25
26 # Logging conrols
27 LOGLEVEL = 2
28 DEBUGFORMAT = False
29
30 import tempfile
31
32 import pytan
33 handler = pytan.Handler(
34     username=USERNAME,
```



```

35     password=PASSWORD,
36     host=HOST,
37     port=PORT,
38     loglevel=LOGLEVEL,
39     debugformat=DEBUGFORMAT,
40 )
41
42 print handler
43
44 # setup the arguments for the handler method
45 kwargs = {}
46 kwargs['report_dir'] = tempfile.gettempdir()
47 kwargs["run"] = True
48 kwargs["package"] = u''
49
50
51 # call the handler with the deploy_action_human method, passing in kwargs for arguments
52 # this should throw an exception: pytan.utils.HumanParserError
53 import traceback
54 try:
55     handler.deploy_action_human(**kwargs)
56 except Exception as e:
57     traceback.print_exc(file=sys.stdout)

```

### Output from Python Code

```

1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3279
2 Traceback (most recent call last):
3   File "<string>", line 56, in <module>
4   File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 1187, in deploy_action_human
5     package_def = utils.dehumanize_package(package)
6   File "/Users/jolsen/gh/pytan/lib/pytan/utils.py", line 1333, in dehumanize_package
7     raise HumanParserError(err(package))
8 HumanParserError: u'' must be a string supplied as 'package'

```

### Invalid deploy action filters help

Have `deploy_action_human()` return the help for filters

### Example Python Code

```

1 import os
2 import sys
3 sys.dont_write_bytecode = True
4
5 # Determine our script name, script dir
6 my_file = os.path.abspath(sys.argv[0])
7 my_dir = os.path.dirname(my_file)
8
9 # determine the pytan lib dir and add it to the path
10 parent_dir = os.path.dirname(my_dir)
11 pytan_root_dir = os.path.dirname(parent_dir)
12 lib_dir = os.path.join(pytan_root_dir, 'lib')
13 path_adds = [lib_dir]

```

```
14
15 for aa in path_adds:
16     if aa not in sys.path:
17         sys.path.append(aa)
18
19
20 # connection info for Tanium Server
21 USERNAME = "Tanium User"
22 PASSWORD = "T@n!um"
23 HOST = "172.16.31.128"
24 PORT = "444"
25
26 # Logging conrols
27 LOGLEVEL = 2
28 DEBUGFORMAT = False
29
30 import tempfile
31
32 import pytan
33 handler = pytan.Handler(
34     username=USERNAME,
35     password=PASSWORD,
36     host=HOST,
37     port=PORT,
38     loglevel=LOGLEVEL,
39     debugformat=DEBUGFORMAT,
40 )
41
42 print handler
43
44 # setup the arguments for the handler method
45 kwargs = {}
46 kwargs['report_dir'] = tempfile.gettempdir()
47 kwargs["filters_help"] = True
48
49
50 # call the handler with the deploy_action_human method, passing in kwargs for arguments
51 # this should throw an exception: pytan.utils.PytanHelp
52 import traceback
53 try:
54     handler.deploy_action_human(**kwargs)
55 except Exception as e:
56     traceback.print_exc(file=sys.stdout)
```

### Output from Python Code

```
1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3279
2 Traceback (most recent call last):
3   File "<string>", line 55, in <module>
4   File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 1160, in deploy_action_human
5     raise PytanHelp(utils.help_filters())
6 PytanHelp:
7 Filters Help
8 =====
9
10 Filters are used generously throughout pytan. When used as part of a
11 sensor string, they control what data is shown for the columns that
```

the sensor returns. When filters are used for whole question filters, they control what rows will be returned. They are used by Groups to define group membership, deploy actions to determine which machines should have the action deployed to it, and more.

A filter string is a human string that describes, a sensor followed by ', that FILTER:VALUE', where FILTER is a valid filter string, and VALUE is the string that you want FILTER to match on.

#### Valid Filters

-----

```
'<'
    Help: Filter for less than VALUE
    Example: "Sensor1, that <:VALUE"

'less'
    Help: Filter for less than VALUE
    Example: "Sensor1, that less:VALUE"

'lt'
    Help: Filter for less than VALUE
    Example: "Sensor1, that lt:VALUE"

'less than'
    Help: Filter for less than VALUE
    Example: "Sensor1, that less than:VALUE"

'!<'
    Help: Filter for not less than VALUE
    Example: "Sensor1, that !<:VALUE"

'notless'
    Help: Filter for not less than VALUE
    Example: "Sensor1, that notless:VALUE"

'not less'
    Help: Filter for not less than VALUE
    Example: "Sensor1, that not less:VALUE"

'not less than'
    Help: Filter for not less than VALUE
    Example: "Sensor1, that not less than:VALUE"

'<='
    Help: Filter for less than or equal to VALUE
    Example: "Sensor1, that <=:VALUE"

'less equal'
    Help: Filter for less than or equal to VALUE
    Example: "Sensor1, that less equal:VALUE"

'lessequal'
    Help: Filter for less than or equal to VALUE
    Example: "Sensor1, that lessequal:VALUE"

'le'
    Help: Filter for less than or equal to VALUE
```

```
70         Example: "Sensor1, that le:VALUE"
71
72     '!=>'
73         Help: Filter for not less than or equal to VALUE
74         Example: "Sensor1, that !=>:VALUE"
75
76     'not less equal'
77         Help: Filter for not less than or equal to VALUE
78         Example: "Sensor1, that not less equal:VALUE"
79
80     'not lessequal'
81         Help: Filter for not less than or equal to VALUE
82         Example: "Sensor1, that not lessequal:VALUE"
83
84     '>'
85         Help: Filter for greater than VALUE
86         Example: "Sensor1, that >:VALUE"
87
88     'greater'
89         Help: Filter for greater than VALUE
90         Example: "Sensor1, that greater:VALUE"
91
92     'gt'
93         Help: Filter for greater than VALUE
94         Example: "Sensor1, that gt:VALUE"
95
96     'greater than'
97         Help: Filter for greater than VALUE
98         Example: "Sensor1, that greater than:VALUE"
99
100    '!=>'
101        Help: Filter for not greater than VALUE
102        Example: "Sensor1, that !=>:VALUE"
103
104    'not greater'
105        Help: Filter for not greater than VALUE
106        Example: "Sensor1, that not greater:VALUE"
107
108    'notgreater'
109        Help: Filter for not greater than VALUE
110        Example: "Sensor1, that notgreater:VALUE"
111
112    'not greater than'
113        Help: Filter for not greater than VALUE
114        Example: "Sensor1, that not greater than:VALUE"
115
116    '=>'
117        Help: Filter for greater than or equal to VALUE
118        Example: "Sensor1, that =>:VALUE"
119
120    'greater equal'
121        Help: Filter for greater than or equal to VALUE
122        Example: "Sensor1, that greater equal:VALUE"
123
124    'greaterequal'
125        Help: Filter for greater than or equal to VALUE
126        Example: "Sensor1, that greaterequal:VALUE"
127
```

```

128 'ge'
129     Help: Filter for greater than or equal to VALUE
130     Example: "Sensor1, that ge:VALUE"
131
132 '!=>'
133     Help: Filter for not greater than VALUE
134     Example: "Sensor1, that !=>:VALUE"
135
136 'not greater equal'
137     Help: Filter for not greater than VALUE
138     Example: "Sensor1, that not greater equal:VALUE"
139
140 'notgreaterequal'
141     Help: Filter for not greater than VALUE
142     Example: "Sensor1, that notgreaterequal:VALUE"
143
144 '='
145     Help: Filter for equals to VALUE
146     Example: "Sensor1, that =:VALUE"
147
148 'equal'
149     Help: Filter for equals to VALUE
150     Example: "Sensor1, that equal:VALUE"
151
152 'equals'
153     Help: Filter for equals to VALUE
154     Example: "Sensor1, that equals:VALUE"
155
156 'eq'
157     Help: Filter for equals to VALUE
158     Example: "Sensor1, that eq:VALUE"
159
160 '!='
161     Help: Filter for not equals to VALUE
162     Example: "Sensor1, that !=:VALUE"
163
164 'not equal'
165     Help: Filter for not equals to VALUE
166     Example: "Sensor1, that not equal:VALUE"
167
168 'notequal'
169     Help: Filter for not equals to VALUE
170     Example: "Sensor1, that notequal:VALUE"
171
172 'not equals'
173     Help: Filter for not equals to VALUE
174     Example: "Sensor1, that not equals:VALUE"
175
176 'notequals'
177     Help: Filter for not equals to VALUE
178     Example: "Sensor1, that notequals:VALUE"
179
180 'ne'
181     Help: Filter for not equals to VALUE
182     Example: "Sensor1, that ne:VALUE"
183
184 'contains'
185     Help: Filter for contains VALUE (adds .* before and after VALUE)

```

```
186     Example: "Sensor1, that contains:VALUE"
187
188 'does not contain'
189     Help: Filter for does not contain VALUE (adds .* before and after VALUE)
190     Example: "Sensor1, that does not contain:VALUE"
191
192 'doesnotcontain'
193     Help: Filter for does not contain VALUE (adds .* before and after VALUE)
194     Example: "Sensor1, that doesnotcontain:VALUE"
195
196 'not contains'
197     Help: Filter for does not contain VALUE (adds .* before and after VALUE)
198     Example: "Sensor1, that not contains:VALUE"
199
200 'notcontains'
201     Help: Filter for does not contain VALUE (adds .* before and after VALUE)
202     Example: "Sensor1, that notcontains:VALUE"
203
204 'starts with'
205     Help: Filter for starts with VALUE (adds .* after VALUE)
206     Example: "Sensor1, that starts with:VALUE"
207
208 'startswith'
209     Help: Filter for starts with VALUE (adds .* after VALUE)
210     Example: "Sensor1, that startswith:VALUE"
211
212 'does not start with'
213     Help: Filter for does not start with VALUE (adds .* after VALUE)
214     Example: "Sensor1, that does not start with:VALUE"
215
216 'doesnotstartswith'
217     Help: Filter for does not start with VALUE (adds .* after VALUE)
218     Example: "Sensor1, that doesnotstartswith:VALUE"
219
220 'not starts with'
221     Help: Filter for does not start with VALUE (adds .* after VALUE)
222     Example: "Sensor1, that not starts with:VALUE"
223
224 'notstartswith'
225     Help: Filter for does not start with VALUE (adds .* after VALUE)
226     Example: "Sensor1, that notstartswith:VALUE"
227
228 'ends with'
229     Help: Filter for ends with VALUE (adds .* before VALUE)
230     Example: "Sensor1, that ends with:VALUE"
231
232 'endswith'
233     Help: Filter for ends with VALUE (adds .* before VALUE)
234     Example: "Sensor1, that endswith:VALUE"
235
236 'does not end with'
237     Help: Filter for does bit end with VALUE (adds .* before VALUE)
238     Example: "Sensor1, that does not end with:VALUE"
239
240 'doesnotendwith'
241     Help: Filter for does bit end with VALUE (adds .* before VALUE)
242     Example: "Sensor1, that doesnotendwith:VALUE"
243
```

```

244 'not ends with'
245     Help: Filter for does bit end with VALUE (adds .* before VALUE)
246     Example: "Sensor1, that not ends with:VALUE"
247
248 'notstartswith'
249     Help: Filter for does bit end with VALUE (adds .* before VALUE)
250     Example: "Sensor1, that notstartswith:VALUE"
251
252 'is not'
253     Help: Filter for non regular expression match for VALUE
254     Example: "Sensor1, that is not:VALUE"
255
256 'not regex'
257     Help: Filter for non regular expression match for VALUE
258     Example: "Sensor1, that not regex:VALUE"
259
260 'notregex'
261     Help: Filter for non regular expression match for VALUE
262     Example: "Sensor1, that notregex:VALUE"
263
264 'not regex match'
265     Help: Filter for non regular expression match for VALUE
266     Example: "Sensor1, that not regex match:VALUE"
267
268 'notregexmatch'
269     Help: Filter for non regular expression match for VALUE
270     Example: "Sensor1, that notregexmatch:VALUE"
271
272 'nre'
273     Help: Filter for non regular expression match for VALUE
274     Example: "Sensor1, that nre:VALUE"
275
276 'is'
277     Help: Filter for regular expression match for VALUE
278     Example: "Sensor1, that is:VALUE"
279
280 'regex'
281     Help: Filter for regular expression match for VALUE
282     Example: "Sensor1, that regex:VALUE"
283
284 'regex match'
285     Help: Filter for regular expression match for VALUE
286     Example: "Sensor1, that regex match:VALUE"
287
288 'regexmatch'
289     Help: Filter for regular expression match for VALUE
290     Example: "Sensor1, that regexmatch:VALUE"
291
292 're'
293     Help: Filter for regular expression match for VALUE
294     Example: "Sensor1, that re:VALUE"
295
296 '<'
297     Help: Filter for less than VALUE
298     Example: "Sensor1, that <:VALUE"
299
300 'less'
301     Help: Filter for less than VALUE

```

```
302         Example: "Sensor1, that less:VALUE"
303
304     'lt'
305         Help: Filter for less than VALUE
306         Example: "Sensor1, that lt:VALUE"
307
308     'less than'
309         Help: Filter for less than VALUE
310         Example: "Sensor1, that less than:VALUE"
311
312     '!'<'
313         Help: Filter for not less than VALUE
314         Example: "Sensor1, that !<:VALUE"
315
316     'notless'
317         Help: Filter for not less than VALUE
318         Example: "Sensor1, that notless:VALUE"
319
320     'not less'
321         Help: Filter for not less than VALUE
322         Example: "Sensor1, that not less:VALUE"
323
324     'not less than'
325         Help: Filter for not less than VALUE
326         Example: "Sensor1, that not less than:VALUE"
327
328     '<='
329         Help: Filter for less than or equal to VALUE
330         Example: "Sensor1, that <=:VALUE"
331
332     'less equal'
333         Help: Filter for less than or equal to VALUE
334         Example: "Sensor1, that less equal:VALUE"
335
336     'lessequal'
337         Help: Filter for less than or equal to VALUE
338         Example: "Sensor1, that lessequal:VALUE"
339
340     'le'
341         Help: Filter for less than or equal to VALUE
342         Example: "Sensor1, that le:VALUE"
343
344     '!'<='
345         Help: Filter for not less than or equal to VALUE
346         Example: "Sensor1, that !<=:VALUE"
347
348     'not less equal'
349         Help: Filter for not less than or equal to VALUE
350         Example: "Sensor1, that not less equal:VALUE"
351
352     'not lessequal'
353         Help: Filter for not less than or equal to VALUE
354         Example: "Sensor1, that not lessequal:VALUE"
355
356     '>'
357         Help: Filter for greater than VALUE
358         Example: "Sensor1, that >:VALUE"
359
```



```

360 'greater'
361     Help: Filter for greater than VALUE
362     Example: "Sensor1, that greater:VALUE"
363
364 'gt'
365     Help: Filter for greater than VALUE
366     Example: "Sensor1, that gt:VALUE"
367
368 'greater than'
369     Help: Filter for greater than VALUE
370     Example: "Sensor1, that greater than:VALUE"
371
372 '!'>'
373     Help: Filter for not greater than VALUE
374     Example: "Sensor1, that !>:VALUE"
375
376 'not greater'
377     Help: Filter for not greater than VALUE
378     Example: "Sensor1, that not greater:VALUE"
379
380 'notgreater'
381     Help: Filter for not greater than VALUE
382     Example: "Sensor1, that notgreater:VALUE"
383
384 'not greater than'
385     Help: Filter for not greater than VALUE
386     Example: "Sensor1, that not greater than:VALUE"
387
388 '=>'
389     Help: Filter for greater than or equal to VALUE
390     Example: "Sensor1, that =>:VALUE"
391
392 'greater equal'
393     Help: Filter for greater than or equal to VALUE
394     Example: "Sensor1, that greater equal:VALUE"
395
396 'greaterequal'
397     Help: Filter for greater than or equal to VALUE
398     Example: "Sensor1, that greaterequal:VALUE"
399
400 'ge'
401     Help: Filter for greater than or equal to VALUE
402     Example: "Sensor1, that ge:VALUE"
403
404 '!'>='
405     Help: Filter for not greater than VALUE
406     Example: "Sensor1, that !>=:VALUE"
407
408 'not greater equal'
409     Help: Filter for not greater than VALUE
410     Example: "Sensor1, that not greater equal:VALUE"
411
412 'notgreaterequal'
413     Help: Filter for not greater than VALUE
414     Example: "Sensor1, that notgreaterequal:VALUE"
415
416 '='
417     Help: Filter for equals to VALUE

```

```
418         Example: "Sensor1, that =:VALUE"
419
420     'equal'
421         Help: Filter for equals to VALUE
422         Example: "Sensor1, that equal:VALUE"
423
424     'equals'
425         Help: Filter for equals to VALUE
426         Example: "Sensor1, that equals:VALUE"
427
428     'eq'
429         Help: Filter for equals to VALUE
430         Example: "Sensor1, that eq:VALUE"
431
432     '!='
433         Help: Filter for not equals to VALUE
434         Example: "Sensor1, that !=:VALUE"
435
436     'not equal'
437         Help: Filter for not equals to VALUE
438         Example: "Sensor1, that not equal:VALUE"
439
440     'notequal'
441         Help: Filter for not equals to VALUE
442         Example: "Sensor1, that notequal:VALUE"
443
444     'not equals'
445         Help: Filter for not equals to VALUE
446         Example: "Sensor1, that not equals:VALUE"
447
448     'notequals'
449         Help: Filter for not equals to VALUE
450         Example: "Sensor1, that notequals:VALUE"
451
452     'ne'
453         Help: Filter for not equals to VALUE
454         Example: "Sensor1, that ne:VALUE"
455
456     'contains'
457         Help: Filter for contains VALUE (adds .* before and after VALUE)
458         Example: "Sensor1, that contains:VALUE"
459
460     'does not contain'
461         Help: Filter for does not contain VALUE (adds .* before and after VALUE)
462         Example: "Sensor1, that does not contain:VALUE"
463
464     'doesnotcontain'
465         Help: Filter for does not contain VALUE (adds .* before and after VALUE)
466         Example: "Sensor1, that doesnotcontain:VALUE"
467
468     'not contains'
469         Help: Filter for does not contain VALUE (adds .* before and after VALUE)
470         Example: "Sensor1, that not contains:VALUE"
471
472     'notcontains'
473         Help: Filter for does not contain VALUE (adds .* before and after VALUE)
474         Example: "Sensor1, that notcontains:VALUE"
475
```

```

476 'starts with'
477     Help: Filter for starts with VALUE (adds .* after VALUE)
478     Example: "Sensor1, that starts with:VALUE"
479
480 'startswith'
481     Help: Filter for starts with VALUE (adds .* after VALUE)
482     Example: "Sensor1, that startswith:VALUE"
483
484 'does not start with'
485     Help: Filter for does not start with VALUE (adds .* after VALUE)
486     Example: "Sensor1, that does not start with:VALUE"
487
488 'doesnotstartswith'
489     Help: Filter for does not start with VALUE (adds .* after VALUE)
490     Example: "Sensor1, that doesnotstartswith:VALUE"
491
492 'not starts with'
493     Help: Filter for does not start with VALUE (adds .* after VALUE)
494     Example: "Sensor1, that not starts with:VALUE"
495
496 'notstartswith'
497     Help: Filter for does not start with VALUE (adds .* after VALUE)
498     Example: "Sensor1, that notstartswith:VALUE"
499
500 'ends with'
501     Help: Filter for ends with VALUE (adds .* before VALUE)
502     Example: "Sensor1, that ends with:VALUE"
503
504 'endswith'
505     Help: Filter for ends with VALUE (adds .* before VALUE)
506     Example: "Sensor1, that endswith:VALUE"
507
508 'does not end with'
509     Help: Filter for does bit end with VALUE (adds .* before VALUE)
510     Example: "Sensor1, that does not end with:VALUE"
511
512 'doesnotendwith'
513     Help: Filter for does bit end with VALUE (adds .* before VALUE)
514     Example: "Sensor1, that doesnotendwith:VALUE"
515
516 'not ends with'
517     Help: Filter for does bit end with VALUE (adds .* before VALUE)
518     Example: "Sensor1, that not ends with:VALUE"
519
520 'notstartswith'
521     Help: Filter for does bit end with VALUE (adds .* before VALUE)
522     Example: "Sensor1, that notstartswith:VALUE"
523
524 'is not'
525     Help: Filter for non regular expression match for VALUE
526     Example: "Sensor1, that is not:VALUE"
527
528 'not regex'
529     Help: Filter for non regular expression match for VALUE
530     Example: "Sensor1, that not regex:VALUE"
531
532 'notregex'
533     Help: Filter for non regular expression match for VALUE

```

```
534         Example: "Sensor1, that notregex:VALUE"
535
536     'not regex match'
537         Help: Filter for non regular expression match for VALUE
538         Example: "Sensor1, that not regex match:VALUE"
539
540     'notregexmatch'
541         Help: Filter for non regular expression match for VALUE
542         Example: "Sensor1, that notregexmatch:VALUE"
543
544     'nre'
545         Help: Filter for non regular expression match for VALUE
546         Example: "Sensor1, that nre:VALUE"
547
548     'is'
549         Help: Filter for regular expression match for VALUE
550         Example: "Sensor1, that is:VALUE"
551
552     'regex'
553         Help: Filter for regular expression match for VALUE
554         Example: "Sensor1, that regex:VALUE"
555
556     'regex match'
557         Help: Filter for regular expression match for VALUE
558         Example: "Sensor1, that regex match:VALUE"
559
560     'regexmatch'
561         Help: Filter for regular expression match for VALUE
562         Example: "Sensor1, that regexmatch:VALUE"
563
564     're'
565         Help: Filter for regular expression match for VALUE
566         Example: "Sensor1, that re:VALUE"
```

### Invalid deploy action missing parameters

Deploy an action using a package that requires parameters but do not supply any parameters.

### Example Python Code

```
1  import os
2  import sys
3  sys.dont_write_bytecode = True
4
5  # Determine our script name, script dir
6  my_file = os.path.abspath(sys.argv[0])
7  my_dir = os.path.dirname(my_file)
8
9  # determine the pytan lib dir and add it to the path
10 parent_dir = os.path.dirname(my_dir)
11 pytan_root_dir = os.path.dirname(parent_dir)
12 lib_dir = os.path.join(pytan_root_dir, 'lib')
13 path_adds = [lib_dir]
14
15 for aa in path_adds:
```

```

16     if aa not in sys.path:
17         sys.path.append(aa)
18
19
20     # connection info for Tanium Server
21     USERNAME = "Tanium User"
22     PASSWORD = "T@n!um"
23     HOST = "172.16.31.128"
24     PORT = "444"
25
26     # Logging conrols
27     LOGLEVEL = 2
28     DEBUGFORMAT = False
29
30     import tempfile
31
32     import pytan
33     handler = pytan.Handler(
34         username=USERNAME,
35         password=PASSWORD,
36         host=HOST,
37         port=PORT,
38         loglevel=LOGLEVEL,
39         debugformat=DEBUGFORMAT,
40     )
41
42     print handler
43
44     # setup the arguments for the handler method
45     kwargs = {}
46     kwargs['report_dir'] = tempfile.gettempdir()
47     kwargs["run"] = True
48     kwargs["package"] = u'Custom Tagging - Add Tags'
49
50
51     # call the handler with the deploy_action_human method, passing in kwargs for arguments
52     # this should throw an exception: pytan.utils.HandlerError
53     import traceback
54     try:
55         handler.deploy_action_human(**kwargs)
56     except Exception as e:
57         traceback.print_exc(file=sys.stdout)

```

## Output from Python Code

```

1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3279
2 2015-03-26 11:49:08,639 INFO      question_progress: Results 0% (Get Online = "True" from all machine
3 2015-03-26 11:49:13,664 INFO      question_progress: Results 0% (Get Online = "True" from all machine
4 2015-03-26 11:49:18,683 INFO      question_progress: Results 100% (Get Online = "True" from all machi
5 Traceback (most recent call last):
6   File "<string>", line 56, in <module>
7   File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 1193, in deploy_action_human
8     **kwargs
9   File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 1047, in deploy_action
10     empty_ok=False,
11   File "/Users/jolsen/gh/pytan/lib/pytan/utils.py", line 2154, in build_param_objlist
12     raise HandlerError(err(obj_name, p_key, jsonify(obj_param)))

```

```
13 HandlerError: PackageSpec, name: 'Custom Tagging - Add Tags' parameter key '$1' requires a value, pa
14 {
15     "defaultValue": "",
16     "helpString": "Enter tags space-delimited.",
17     "key": "$1",
18     "label": "Add tags (space-delimited)",
19     "maxChars": 0,
20     "model": "com.tanium.components.parameters::TextInputParameter",
21     "parameterType": "com.tanium.components.parameters::TextInputParameter",
22     "promptText": "e.g. PCI DMZ Decomm",
23     "restrict": null,
24     "validationExpressions": [
25         {
26             "expression": "\\S",
27             "flags": "",
28             "helpString": "You must enter a value",
29             "model": "com.tanium.models::ValidationExpression",
30             "parameterType": "com.tanium.models::ValidationExpression"
31         }
32     ],
33     "value": ""
34 }
```

## pytan API Valid Create Object Examples

### Create user

Create a user called API Test User

### Example Python Code

```
1 import os
2 import sys
3 sys.dont_write_bytecode = True
4
5 # Determine our script name, script dir
6 my_file = os.path.abspath(sys.argv[0])
7 my_dir = os.path.dirname(my_file)
8
9 # determine the pytan lib dir and add it to the path
10 parent_dir = os.path.dirname(my_dir)
11 pytan_root_dir = os.path.dirname(parent_dir)
12 lib_dir = os.path.join(pytan_root_dir, 'lib')
13 path_adds = [lib_dir]
14
15 for aa in path_adds:
16     if aa not in sys.path:
17         sys.path.append(aa)
18
19
20 # connection info for Tanium Server
21 USERNAME = "Tanium User"
22 PASSWORD = "T@n!um"
23 HOST = "172.16.31.128"
24 PORT = "444"
```

```

25
26 # Logging conrols
27 LOGLEVEL = 2
28 DEBUGFORMAT = False
29
30 import tempfile
31
32 import pytan
33 handler = pytan.Handler(
34     username=USERNAME,
35     password=PASSWORD,
36     host=HOST,
37     port=PORT,
38     loglevel=LOGLEVEL,
39     debugformat=DEBUGFORMAT,
40 )
41
42 print handler
43
44 # setup the arguments for the delete method (to remove the package in case it exists)
45 delete_kwargs = {}
46 delete_kwargs["objtype"] = 'user'
47 delete_kwargs["name"] = 'API Test User'
48
49
50 # setup the arguments for the handler method
51 kwargs = {}
52 kwargs["username"] = u'API Test User'
53 kwargs["rolename"] = u'Administrator'
54 kwargs["properties"] = [[u'property1', u'value1']]
55
56 # delete the object in case it already exists
57 try:
58     handler.delete(**delete_kwargs)
59 except Exception as e:
60     print e
61
62 # call the handler with the create_user method, passing in kwargs for arguments
63 response = handler.create_user(**kwargs)
64
65
66 print ""
67 print "Type of response: ", type(response)
68
69 print ""
70 print "print of response:"
71 print response
72
73 print ""
74 print "print the object returned in JSON format:"
75 print response.to_json(response)
76
77 # delete the object, we are done with it now
78 try:
79     handler.delete(**delete_kwargs)
80 except Exception as e:
81     print e

```

## Output from Python Code

```

1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3279
2 No results found searching for user with {'name': 'API Test User'}!!
3 2015-03-26 11:49:18,767 INFO      handler: New user 'API Test User' created with ID 1015, roles: ['Ad
4
5 Type of response:  <class 'taniumpy.object_types.user.User'>
6
7 print of response:
8 User, name: 'API Test User'
9
10 print the object returned in JSON format:
11 {
12     "_type": "user",
13     "deleted_flag": 0,
14     "group_id": 0,
15     "id": 1015,
16     "last_login": "2001-01-01T00:00:00",
17     "metadata": {
18         "_type": "metadata",
19         "item": [
20             {
21                 "_type": "item",
22                 "admin_flag": 0,
23                 "name": "TConsole.User.Property.property1",
24                 "value": "value1"
25             }
26         ]
27     },
28     "name": "API Test User",
29     "permissions": {
30         "_type": "permissions",
31         "permission": "admin"
32     },
33     "roles": {
34         "_type": "roles",
35         "role": [
36             {
37                 "_type": "role",
38                 "description": "Administrators can perform all functions in the system, including creating o
39                 "id": 1,
40                 "name": "Administrator",
41                 "permissions": {
42                     "_type": "permissions",
43                     "permission": "admin"
44                 }
45             }
46         ]
47     }
48 }
49 2015-03-26 11:49:18,797 INFO      handler: Deleted "User, name: 'API Test User'"

```

## Create package

Create a package called package49



## Example Python Code

```

1  import os
2  import sys
3  sys.dont_write_bytecode = True
4
5  # Determine our script name, script dir
6  my_file = os.path.abspath(sys.argv[0])
7  my_dir = os.path.dirname(my_file)
8
9  # determine the pytan lib dir and add it to the path
10 parent_dir = os.path.dirname(my_dir)
11 pytan_root_dir = os.path.dirname(parent_dir)
12 lib_dir = os.path.join(pytan_root_dir, 'lib')
13 path_adds = [lib_dir]
14
15 for aa in path_adds:
16     if aa not in sys.path:
17         sys.path.append(aa)
18
19
20 # connection info for Tanium Server
21 USERNAME = "Tanium User"
22 PASSWORD = "T@n!um"
23 HOST = "172.16.31.128"
24 PORT = "444"
25
26 # Logging conrols
27 LOGLEVEL = 2
28 DEBUGFORMAT = False
29
30 import tempfile
31
32 import pytan
33 handler = pytan.Handler(
34     username=USERNAME,
35     password=PASSWORD,
36     host=HOST,
37     port=PORT,
38     loglevel=LOGLEVEL,
39     debugformat=DEBUGFORMAT,
40 )
41
42 print handler
43
44 # setup the arguments for the delete method (to remove the package in case it exists)
45 delete_kwargs = {}
46 delete_kwargs["objtype"] = 'package'
47 delete_kwargs["name"] = 'package49'
48
49
50 # setup the arguments for the handler method
51 kwargs = {}
52 kwargs["expire_seconds"] = 1500
53 kwargs["display_name"] = u'package49 API test'
54 kwargs["name"] = u'package49'
55 kwargs["parameters_json_file"] = u'../doc/example_of_all_package_parameters.json'
56 kwargs["verify_expire_seconds"] = 3600
57 kwargs["command"] = u'package49 $1 $2 $3 $4 $5 $6 $7 $8'

```

```
58 kwargs["file_urls"] = [u'3600::testing.vbs||https://content.tanium.com/files/initialcontent/bundles/
59 kwargs["verify_filter_options"] = [u'and']
60 kwargs["verify_filters"] = [u'Custom Tags, that contains:tag']
61 kwargs["command_timeout_seconds"] = 9999
62
63 # delete the object in case it already exists
64 try:
65     handler.delete(**delete_kwargs)
66 except Exception as e:
67     print e
68
69 # call the handler with the create_package method, passing in kwargs for arguments
70 response = handler.create_package(**kwargs)
71
72
73 print ""
74 print "Type of response: ", type(response)
75
76 print ""
77 print "print of response:"
78 print response
79
80 print ""
81 print "print the object returned in JSON format:"
82 print response.to_json(response)
83
84 # delete the object, we are done with it now
85 try:
86     handler.delete(**delete_kwargs)
87 except Exception as e:
88     print e
```

### Output from Python Code

```
1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3279
2 No results found searching for PackageSpec, name: 'package49'!!
3 2015-03-26 11:49:18,923 INFO handler: New package 'package49' created with ID 6285, command: 'pa
4
5 Type of response: <class 'taniumpy.object_types.package_spec.PackageSpec'>
6
7 print of response:
8 PackageSpec, name: 'package49'
9
10 print the object returned in JSON format:
11 {
12     "_type": "package_spec",
13     "available_time": "1900-01-01T00:00:00",
14     "command": "package49 $1 $2 $3 $4 $5 $6 $7 $8",
15     "command_timeout": 9999,
16     "creation_time": "2015-03-26T15:49:19",
17     "deleted_flag": 0,
18     "display_name": "package49 API test",
19     "expire_seconds": 1500,
20     "files": {
21         "_type": "package_files",
22         "file": [
23             {
```

```

24         "_type": "file",
25         "bytes_downloaded": 0,
26         "bytes_total": 0,
27         "cache_status": "UNCACHED",
28         "download_seconds": 3600,
29         "id": 6379,
30         "name": "testing.vbs",
31         "size": 0,
32         "source": "https://content.tanium.com/files/initialcontent/bundles/2014-10-01_11-32-15-7844/",
33         "status": 0
34     }
35 ]
36 },
37 "hidden_flag": 0,
38 "id": 6285,
39 "last_modified_by": "Tanium User",
40 "last_update": "2015-03-26T15:49:19",
41 "modification_time": "2015-03-26T15:49:19",
42 "name": "package49",
43 "parameter_definition": "{\"parameterType\": \"com.tanium.components.parameters::ParametersArray\"",
44 "source_id": 0,
45 "verify_group_id": 19222
46 }
47 2015-03-26 11:49:18,945 INFO          handler: Deleted 'PackageSpec, id: 6285'

```

## Create group

Create a group called All Windows Computers API Test

## Example Python Code

```

1  import os
2  import sys
3  sys.dont_write_bytecode = True
4
5  # Determine our script name, script dir
6  my_file = os.path.abspath(sys.argv[0])
7  my_dir = os.path.dirname(my_file)
8
9  # determine the pytan lib dir and add it to the path
10 parent_dir = os.path.dirname(my_dir)
11 pytan_root_dir = os.path.dirname(parent_dir)
12 lib_dir = os.path.join(pytan_root_dir, 'lib')
13 path_adds = [lib_dir]
14
15 for aa in path_adds:
16     if aa not in sys.path:
17         sys.path.append(aa)
18
19
20 # connection info for Tanium Server
21 USERNAME = "Tanium User"
22 PASSWORD = "T@n!um"
23 HOST = "172.16.31.128"
24 PORT = "444"

```

```
25
26 # Logging controls
27 LOGLEVEL = 2
28 DEBUGFORMAT = False
29
30 import tempfile
31
32 import pytan
33 handler = pytan.Handler(
34     username=USERNAME,
35     password=PASSWORD,
36     host=HOST,
37     port=PORT,
38     loglevel=LOGLEVEL,
39     debugformat=DEBUGFORMAT,
40 )
41
42 print handler
43
44 # setup the arguments for the delete method (to remove the package in case it exists)
45 delete_kwargs = {}
46 delete_kwargs["objtype"] = 'group'
47 delete_kwargs["name"] = 'All Windows Computers API Test'
48
49
50 # setup the arguments for the handler method
51 kwargs = {}
52 kwargs["groupname"] = u'All Windows Computers API Test'
53 kwargs["filters"] = [u'Operating System, that contains:Windows']
54 kwargs["filter_options"] = [u'and']
55
56 # delete the object in case it already exists
57 try:
58     handler.delete(**delete_kwargs)
59 except Exception as e:
60     print e
61
62 # call the handler with the create_group method, passing in kwargs for arguments
63 response = handler.create_group(**kwargs)
64
65
66 print ""
67 print "Type of response: ", type(response)
68
69 print ""
70 print "print of response:"
71 print response
72
73 print ""
74 print "print the object returned in JSON format:"
75 print response.to_json(response)
76
77 # delete the object, we are done with it now
78 try:
79     handler.delete(**delete_kwargs)
80 except Exception as e:
81     print e
```

## Output from Python Code

```

1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3279
2 No results found searching for Group, name: 'All Windows Computers API Test'!!
3 2015-03-26 11:49:19,021 INFO      handler: New group 'All Windows Computers API Test' created with ID
4
5 Type of response: <class 'taniumpy.object_types.group.Group'>
6
7 print of response:
8 Group, name: 'All Windows Computers API Test'
9
10 print the object returned in JSON format:
11 {
12     "_type": "group",
13     "and_flag": 1,
14     "deleted_flag": 0,
15     "filters": {
16         "_type": "filters",
17         "filter": [
18             {
19                 "_type": "filter",
20                 "all_times_flag": 0,
21                 "all_values_flag": 0,
22                 "delimiter_index": 0,
23                 "ignore_case_flag": 1,
24                 "max_age_seconds": 0,
25                 "not_flag": 0,
26                 "operator": "RegexMatch",
27                 "sensor": {
28                     "_type": "sensor",
29                     "hash": 45421433
30                 },
31                 "substring_flag": 0,
32                 "substring_length": 0,
33                 "substring_start": 0,
34                 "utf8_flag": 0,
35                 "value": ".*Windows.*",
36                 "value_type": "String"
37             }
38         ]
39     },
40     "id": 19223,
41     "name": "All Windows Computers API Test",
42     "not_flag": 0,
43     "sub_groups": {
44         "_type": "groups",
45         "group": []
46     },
47     "text": " Operating System contains \"Windows\\",
48     "type": 0
49 }
50 2015-03-26 11:49:19,044 INFO      handler: Deleted 'Group, id: 19223'

```

## Create whitelisted url

Create a whitelisted url

## Example Python Code

```
1  import os
2  import sys
3  sys.dont_write_bytecode = True
4
5  # Determine our script name, script dir
6  my_file = os.path.abspath(sys.argv[0])
7  my_dir = os.path.dirname(my_file)
8
9  # determine the pytan lib dir and add it to the path
10 parent_dir = os.path.dirname(my_dir)
11 pytan_root_dir = os.path.dirname(parent_dir)
12 lib_dir = os.path.join(pytan_root_dir, 'lib')
13 path_adds = [lib_dir]
14
15 for aa in path_adds:
16     if aa not in sys.path:
17         sys.path.append(aa)
18
19
20 # connection info for Tanium Server
21 USERNAME = "Tanium User"
22 PASSWORD = "T@n!um"
23 HOST = "172.16.31.128"
24 PORT = "444"
25
26 # Logging conrols
27 LOGLEVEL = 2
28 DEBUGFORMAT = False
29
30 import tempfile
31
32 import pytan
33 handler = pytan.Handler(
34     username=USERNAME,
35     password=PASSWORD,
36     host=HOST,
37     port=PORT,
38     loglevel=LOGLEVEL,
39     debugformat=DEBUGFORMAT,
40 )
41
42 print handler
43
44 # setup the arguments for the delete method (to remove the package in case it exists)
45 delete_kwargs = {}
46 delete_kwargs["objtype"] = 'whitelisted_url'
47 delete_kwargs["url_regex"] = 'regex:http://test.com/.API_Test.*URL'
48
49
50 # setup the arguments for the handler method
51 kwargs = {}
52 kwargs["url"] = u'http://test.com/.API_Test.*URL'
53 kwargs["regex"] = True
54 kwargs["properties"] = [[u'property1', u'value1']]
55 kwargs["download_seconds"] = 3600
56
57 # delete the object in case it already exists
```

```

58 try:
59     handler.delete(**delete_kwargs)
60 except Exception as e:
61     print e
62
63 # call the handler with the create_whitelisted_url method, passing in kwargs for arguments
64 response = handler.create_whitelisted_url(**kwargs)
65
66
67 print ""
68 print "Type of response: ", type(response)
69
70 print ""
71 print "print of response:"
72 print response
73
74 print ""
75 print "print the object returned in JSON format:"
76 print response.to_json(response)
77
78 # delete the object, we are done with it now
79 try:
80     handler.delete(**delete_kwargs)
81 except Exception as e:
82     print e

```

### Output from Python Code

```

1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3279
2 No results found searching for whitelisted_url with {'url_regex': 'regex:http://test.com/.API_Test.*URL'}
3 2015-03-26 11:49:19,095 INFO handler: New Whitelisted URL 'regex:http://test.com/.API_Test.*URL'
4
5 Type of response: <class 'taniumpy.object_types.white_listed_url.WhiteListedUrl'>
6
7 print of response:
8 WhiteListedUrl, id: 1027
9
10 print the object returned in JSON format:
11 {
12     "_type": "white_listed_url",
13     "download_seconds": 3600,
14     "id": 1027,
15     "metadata": {
16         "_type": "metadata",
17         "item": [
18             {
19                 "_type": "item",
20                 "admin_flag": 0,
21                 "name": "TConsole.WhitelistedURL.property1",
22                 "value": "value1"
23             }
24         ]
25     },
26     "url_regex": "regex:http://test.com/.API_Test.*URL"
27 }
28 2015-03-26 11:49:19,119 INFO handler: Deleted 'WhiteListedUrl, id: 1027'

```

## pytan API Invalid Create Object Examples

### Invalid create sensor

Create a sensor (Unsupported!)

#### Example Python Code

```
1  import os
2  import sys
3  sys.dont_write_bytecode = True
4
5  # Determine our script name, script dir
6  my_file = os.path.abspath(sys.argv[0])
7  my_dir = os.path.dirname(my_file)
8
9  # determine the pytan lib dir and add it to the path
10 parent_dir = os.path.dirname(my_dir)
11 pytan_root_dir = os.path.dirname(parent_dir)
12 lib_dir = os.path.join(pytan_root_dir, 'lib')
13 path_adds = [lib_dir]
14
15 for aa in path_adds:
16     if aa not in sys.path:
17         sys.path.append(aa)
18
19
20 # connection info for Tanium Server
21 USERNAME = "Tanium User"
22 PASSWORD = "T@n!um"
23 HOST = "172.16.31.128"
24 PORT = "444"
25
26 # Logging conrols
27 LOGLEVEL = 2
28 DEBUGFORMAT = False
29
30 import tempfile
31
32 import pytan
33 handler = pytan.Handler(
34     username=USERNAME,
35     password=PASSWORD,
36     host=HOST,
37     port=PORT,
38     loglevel=LOGLEVEL,
39     debugformat=DEBUGFORMAT,
40 )
41
42 print handler
43
44 # setup the arguments for the handler method
45 kwargs = {}
46
47
48 # call the handler with the create_sensor method, passing in kwargs for arguments
49 # this should throw an exception: pytan.utils.HandlerError
```



```

50 import traceback
51 try:
52     handler.create_sensor(**kwargs)
53 except Exception as e:
54     traceback.print_exc(file=sys.stdout)

```

### Output from Python Code

```

1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3279
2 Traceback (most recent call last):
3   File "<string>", line 53, in <module>
4   File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 537, in create_sensor
5     raise HandlerError(m)
6 HandlerError: Sensor creation not supported via PyTan as of yet, too complex
7 Use create_sensor_from_json() instead!

```

## pytan API Valid Create Object From JSON Examples

### Create package from json

Export a package object to a JSON file, adding ‘API TEST’ to the name of the package before exporting the JSON file and deleting any pre-existing package with the same (new) name, then create a new package object from the exported JSON file

### Example Python Code

```

1 import os
2 import sys
3 sys.dont_write_bytecode = True
4
5 # Determine our script name, script dir
6 my_file = os.path.abspath(sys.argv[0])
7 my_dir = os.path.dirname(my_file)
8
9 # determine the pytan lib dir and add it to the path
10 parent_dir = os.path.dirname(my_dir)
11 pytan_root_dir = os.path.dirname(parent_dir)
12 lib_dir = os.path.join(pytan_root_dir, 'lib')
13 path_adds = [lib_dir]
14
15 for aa in path_adds:
16     if aa not in sys.path:
17         sys.path.append(aa)
18
19
20 # connection info for Tanium Server
21 USERNAME = "Tanium User"
22 PASSWORD = "T@n!um"
23 HOST = "172.16.31.128"
24 PORT = "444"
25
26 # Logging conrols
27 LOGLEVEL = 2

```

```
28 DEBUGFORMAT = False
29
30 import tempfile
31
32 import pytan
33 handler = pytan.Handler(
34     username=USERNAME,
35     password=PASSWORD,
36     host=HOST,
37     port=PORT,
38     loglevel=LOGLEVEL,
39     debugformat=DEBUGFORMAT,
40 )
41
42 print handler
43
44 # set the attribute name and value we want to add to the original object (if any)
45 attr_name = "name"
46 attr_add = " API TEST"
47
48 # delete object before creating it?
49 delete = True
50
51 # setup the arguments for getting an object to export as json file
52 get_kwargs = {}
53 get_kwargs["objtype"] = u'package'
54 get_kwargs["id"] = 31
55
56
57 # get objects to use as an export to JSON file
58 orig_objs = handler.get(**get_kwargs)
59
60 # if attr_name and attr_add exists, modify the orig_objs to add attr_add to the attribute
61 # attr_name
62 if attr_name:
63     for x in orig_objs:
64         new_attr = getattr(x, attr_name)
65         new_attr += attr_add
66         setattr(x, attr_name, new_attr)
67         if delete:
68             # delete the object in case it already exists
69             del_kwargs = {}
70             del_kwargs[attr_name] = new_attr
71             del_kwargs['objtype'] = u'package'
72             try:
73                 handler.delete(**del_kwargs)
74             except Exception as e:
75                 print e
76
77 # export orig_objs to a json file
78 json_file, results = handler.export_to_report_file(
79     obj=orig_objs,
80     export_format='json',
81     report_dir=tempfile.gettempdir(),
82 )
83
84 # create the object from the exported JSON file
85 create_kwargs = {'objtype': u'package', 'json_file': json_file}
```

```

86 response = handler.create_from_json(**create_kwargs)
87
88
89 print ""
90 print "Type of response: ", type(response)
91
92 print ""
93 print "print of response:"
94 print response
95
96 print ""
97 print "print the object returned in JSON format:"
98 print response.to_json(response)

```

### Output from Python Code

```

1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3279
2 2015-03-26 11:49:19,177 INFO handler: Deleted 'PackageSpec, id: 6283'
3 2015-03-26 11:49:19,178 INFO handler: Report file '/var/folders/dk/vjrlr_c53yx6k6gyp2bbt_c40000g
4 2015-03-26 11:49:19,268 INFO handler: New PackageSpec, name: 'Custom Tagging - Add Tags API TEST
5
6 Type of response: <class 'taniumpy.object_types.package_spec_list.PackageSpecList'>
7
8 print of response:
9 PackageSpecList, len: 1
10
11 print the object returned in JSON format:
12 {
13     "_type": "package_specs",
14     "package_spec": [
15         {
16             "_type": "package_spec",
17             "available_time": "1900-01-01T00:00:00",
18             "command": "cmd /c cscript //T:60 add-tags.vbs \"%$1\"",
19             "command_timeout": 60,
20             "creation_time": "2015-03-26T15:49:19",
21             "deleted_flag": 0,
22             "display_name": "Custom Tagging - Add Tags",
23             "expire_seconds": 660,
24             "hidden_flag": 0,
25             "id": 6286,
26             "last_modified_by": "Tanium User",
27             "last_update": "2015-03-26T15:49:19",
28             "metadata": {
29                 "_type": "metadata",
30                 "item": [
31                     {
32                         "_type": "item",
33                         "admin_flag": 0,
34                         "name": "defined",
35                         "value": "Tanium"
36                     },
37                     {
38                         "_type": "item",
39                         "admin_flag": 0,
40                         "name": "category",
41                         "value": "Tanium"

```

```
42         }
43     ]
44 },
45 "modification_time": "2015-03-26T15:49:19",
46 "name": "Custom Tagging - Add Tags API TEST",
47 "parameter_definition": "{\\"parameters\\":[{\\"label\\":\\"Add tags (space-delimited)\\",\\"maxChars\\":100}],\\"source_id\\":0,\\"verify_group_id\\":0}",
48 "source_id": 0,
49 "verify_group_id": 0
50 }
51 ]
52 }
```

### Create user from json

Export a user object to a JSON file, adding ‘ API TEST’ to the name of the user before exporting the JSON file and deleting any pre-existing user with the same (new) name, then create a new user object from the exported JSON file

### Example Python Code

```
1  import os
2  import sys
3  sys.dont_write_bytecode = True
4
5  # Determine our script name, script dir
6  my_file = os.path.abspath(sys.argv[0])
7  my_dir = os.path.dirname(my_file)
8
9  # determine the pytan lib dir and add it to the path
10 parent_dir = os.path.dirname(my_dir)
11 pytan_root_dir = os.path.dirname(parent_dir)
12 lib_dir = os.path.join(pytan_root_dir, 'lib')
13 path_adds = [lib_dir]
14
15 for aa in path_adds:
16     if aa not in sys.path:
17         sys.path.append(aa)
18
19
20 # connection info for Tanium Server
21 USERNAME = "Tanium User"
22 PASSWORD = "T@n!um"
23 HOST = "172.16.31.128"
24 PORT = "444"
25
26 # Logging conrols
27 LOGLEVEL = 2
28 DEBUGFORMAT = False
29
30 import tempfile
31
32 import pytan
33 handler = pytan.Handler(
34     username=USERNAME,
35     password=PASSWORD,
36     host=HOST,
```

```

37     port=PORT,
38     loglevel=LOGLEVEL,
39     debugformat=DEBUGFORMAT,
40 )
41
42 print handler
43
44 # set the attribute name and value we want to add to the original object (if any)
45 attr_name = "name"
46 attr_add = " API TEST"
47
48 # delete object before creating it?
49 delete = True
50
51 # setup the arguments for getting an object to export as json file
52 get_kwargs = {}
53 get_kwargs["objtype"] = u'user'
54 get_kwargs["id"] = 1
55
56
57 # get objects to use as an export to JSON file
58 orig_objs = handler.get(**get_kwargs)
59
60 # if attr_name and attr_add exists, modify the orig_objs to add attr_add to the attribute
61 # attr_name
62 if attr_name:
63     for x in orig_objs:
64         new_attr = getattr(x, attr_name)
65         new_attr += attr_add
66         setattr(x, attr_name, new_attr)
67         if delete:
68             # delete the object in case it already exists
69             del_kwargs = {}
70             del_kwargs[attr_name] = new_attr
71             del_kwargs['objtype'] = u'user'
72             try:
73                 handler.delete(**del_kwargs)
74             except Exception as e:
75                 print e
76
77 # export orig_objs to a json file
78 json_file, results = handler.export_to_report_file(
79     obj=orig_objs,
80     export_format='json',
81     report_dir=tempfile.gettempdir(),
82 )
83
84 # create the object from the exported JSON file
85 create_kwargs = {'objtype': u'user', 'json_file': json_file}
86 response = handler.create_from_json(**create_kwargs)
87
88
89 print ""
90 print "Type of response: ", type(response)
91
92 print ""
93 print "print of response:"
94 print response

```

```
95
96 print ""
97 print "print the object returned in JSON format:"
98 print response.to_json(response)
```

### Output from Python Code

```
1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3279
2 2015-03-26 11:49:19,323 INFO handler: Deleted "User, name: 'Jim Olsen API TEST'"
3 2015-03-26 11:49:19,324 INFO handler: Report file '/var/folders/dk/vjrlr_c53yx6k6gzzp2bbt_c40000g
4 2015-03-26 11:49:19,351 INFO handler: New User, name: 'Jim Olsen API TEST' (ID: 1016) created su
5
6 Type of response: <class 'taniumpy.object_types.user_list.UserList'>
7
8 print of response:
9 UserList, len: 1
10
11 print the object returned in JSON format:
12 {
13     "_type": "users",
14     "user": [
15         {
16             "_type": "user",
17             "deleted_flag": 0,
18             "group_id": 0,
19             "id": 1016,
20             "last_login": "2001-01-01T00:00:00",
21             "metadata": {
22                 "_type": "metadata",
23                 "item": [
24                     {
25                         "_type": "item",
26                         "admin_flag": 0,
27                         "name": "TConsole.User.Preference.FilterClientsPeriod",
28                         "value": "43200"
29                     }
30                 ]
31             },
32             "name": "Jim Olsen API TEST",
33             "permissions": {
34                 "_type": "permissions",
35                 "permission": "admin"
36             },
37             "roles": {
38                 "_type": "roles",
39                 "role": [
40                     {
41                         "_type": "role",
42                         "description": "Administrators can perform all functions in the system, including creati
43                         "id": 1,
44                         "name": "Administrator",
45                         "permissions": {
46                             "_type": "permissions",
47                             "permission": "admin"
48                         }
49                     }
50                 ]
51             }
52         }
53     ]
54 }
```

```

51     }
52     }
53 ]
54 }

```

### Create saved question from json

Export a saved question object to a JSON file, adding ' API TEST' to the name of the saved question before exporting the JSON file and deleting any pre-existing saved question with the same (new) name, then create a new saved question object from the exported JSON file

### Example Python Code

```

1  import os
2  import sys
3  sys.dont_write_bytecode = True
4
5  # Determine our script name, script dir
6  my_file = os.path.abspath(sys.argv[0])
7  my_dir = os.path.dirname(my_file)
8
9  # determine the pytan lib dir and add it to the path
10 parent_dir = os.path.dirname(my_dir)
11 pytan_root_dir = os.path.dirname(parent_dir)
12 lib_dir = os.path.join(pytan_root_dir, 'lib')
13 path_adds = [lib_dir]
14
15 for aa in path_adds:
16     if aa not in sys.path:
17         sys.path.append(aa)
18
19
20 # connection info for Tanium Server
21 USERNAME = "Tanium User"
22 PASSWORD = "T@n!um"
23 HOST = "172.16.31.128"
24 PORT = "444"
25
26 # Logging conrols
27 LOGLEVEL = 2
28 DEBUGFORMAT = False
29
30 import tempfile
31
32 import pytan
33 handler = pytan.Handler(
34     username=USERNAME,
35     password=PASSWORD,
36     host=HOST,
37     port=PORT,
38     loglevel=LOGLEVEL,
39     debugformat=DEBUGFORMAT,
40 )
41
42 print handler

```

```
43
44 # set the attribute name and value we want to add to the original object (if any)
45 attr_name = "name"
46 attr_add = " API TEST"
47
48 # delete object before creating it?
49 delete = True
50
51 # setup the arguments for getting an object to export as json file
52 get_kwargs = {}
53 get_kwargs["objtype"] = u'saved_question'
54 get_kwargs["id"] = 1
55
56
57 # get objects to use as an export to JSON file
58 orig_objs = handler.get(**get_kwargs)
59
60 # if attr_name and attr_add exists, modify the orig_objs to add attr_add to the attribute
61 # attr_name
62 if attr_name:
63     for x in orig_objs:
64         new_attr = getattr(x, attr_name)
65         new_attr += attr_add
66         setattr(x, attr_name, new_attr)
67         if delete:
68             # delete the object in case it already exists
69             del_kwargs = {}
70             del_kwargs[attr_name] = new_attr
71             del_kwargs['objtype'] = u'saved_question'
72             try:
73                 handler.delete(**del_kwargs)
74             except Exception as e:
75                 print e
76
77 # export orig_objs to a json file
78 json_file, results = handler.export_to_report_file(
79     obj=orig_objs,
80     export_format='json',
81     report_dir=tempfile.gettempdir(),
82 )
83
84 # create the object from the exported JSON file
85 create_kwargs = {'objtype': u'saved_question', 'json_file': json_file}
86 response = handler.create_from_json(**create_kwargs)
87
88
89 print ""
90 print "Type of response: ", type(response)
91
92 print ""
93 print "print of response:"
94 print response
95
96 print ""
97 print "print the object returned in JSON format:"
98 print response.to_json(response)
```



## Output from Python Code

```

1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3279
2 2015-03-26 11:49:19,419 INFO handler: Deleted 'SavedQuestion, id: 11657'
3 2015-03-26 11:49:19,421 INFO handler: Report file '/var/folders/dk/vjrlr_c53yx6k6gzzp2bbt_c40000g
4 2015-03-26 11:49:19,457 INFO handler: New SavedQuestion, name: 'Run Unmanaged Asset Scan on All
5
6 Type of response: <class 'taniumpy.object_types.saved_question_list.SavedQuestionList'>
7
8 print of response:
9 SavedQuestionList, len: 1
10
11 print the object returned in JSON format:
12 {
13     "_type": "saved_questions",
14     "saved_question": [
15         {
16             "_type": "saved_question",
17             "action_tracking_flag": 0,
18             "archive_enabled_flag": 0,
19             "archive_owner": {
20                 "_type": "user"
21             },
22             "expire_seconds": 600,
23             "hidden_flag": 0,
24             "id": 11658,
25             "issue_seconds": 120,
26             "issue_seconds_never_flag": 0,
27             "keep_seconds": 0,
28             "mod_time": "2000-01-01T00:00:00",
29             "most_recent_question_id": 32605,
30             "name": "Run Unmanaged Asset Scan on All Machines API TEST",
31             "packages": {
32                 "_type": "package_specs",
33                 "package_spec": []
34             },
35             "public_flag": 1,
36             "query_text": "Get Is Windows from all machines",
37             "question": {
38                 "_type": "question",
39                 "action_tracking_flag": 0,
40                 "expiration": "2015-03-26T15:16:00",
41                 "expire_seconds": 0,
42                 "force_computer_id_flag": 0,
43                 "hidden_flag": 0,
44                 "id": 32605,
45                 "management_rights_group": {
46                     "_type": "group",
47                     "id": 0
48                 },
49                 "query_text": "Get Is Windows from all machines",
50                 "saved_question": {
51                     "_type": "saved_question",
52                     "id": 11658
53                 },
54                 "selects": {
55                     "_type": "selects",
56                     "select": [
57                     {

```

```

58     "_type": "select",
59     "filter": {
60         "_type": "filter",
61         "all_times_flag": 0,
62         "all_values_flag": 0,
63         "delimiter_index": 0,
64         "end_time": "2001-01-01T00:00:00",
65         "ignore_case_flag": 1,
66         "max_age_seconds": 0,
67         "not_flag": 0,
68         "operator": "Less",
69         "start_time": "2001-01-01T00:00:00",
70         "substring_flag": 0,
71         "substring_length": 0,
72         "substring_start": 0,
73         "utf8_flag": 0,
74         "value_type": "String"
75     },
76     "sensor": {
77         "_type": "sensor",
78         "category": "Operating System",
79         "creation_time": "2015-03-03T19:03:34",
80         "delimiter": ",",
81         "description": "Returns whether the machine runs Windows.  True if so, False if not.",
82         "exclude_from_parse_flag": 0,
83         "hash": 2721439124,
84         "hidden_flag": 0,
85         "id": 35,
86         "ignore_case_flag": 1,
87         "last_modified_by": "Jim Olsen",
88         "max_age_seconds": 86400,
89         "metadata": {
90             "_type": "metadata",
91             "item": [
92                 {
93                     "_type": "item",
94                     "admin_flag": 0,
95                     "name": "defined",
96                     "value": "Tanium"
97                 }
98             ]
99         },
100         "modification_time": "2015-03-03T19:03:34",
101         "name": "Is Windows",
102         "queries": {
103             "_type": "queries",
104             "query": [
105                 {
106                     "_type": "query",
107                     "platform": "Windows",
108                     "script": "&#039;=====\n&#039; Is Windows\n",
109                     "script_type": "VBScript"
110                 },
111                 {
112                     "_type": "query",
113                     "platform": "Linux",
114                     "script": "#!/bin/bash\nnecho False\n",
115                     "script_type": "UnixShell"

```

```

116         },
117         {
118             "_type": "query",
119             "platform": "Mac",
120             "script": "#!/bin/bash\necho False\n",
121             "script_type": "UnixShell"
122         }
123     ],
124     },
125     "source_id": 0,
126     "string_count": 3,
127     "value_type": "String"
128 }
129 }
130 ]
131 },
132 "skip_lock_flag": 0,
133 "user": {
134     "_type": "user",
135     "id": 1,
136     "name": "Jim Olsen"
137 },
138 },
139 "row_count_flag": 0,
140 "sort_column": 0,
141 "user": {
142     "_type": "user",
143     "id": 2,
144     "name": "Tanium User"
145 },
146 }
147 ]
148 }

```

### Create action from json

Export an action object to a JSON file, then create a new action object from the exported JSON file. Actions can not be deleted, so do not delete it. This will, in effect, ‘re-deploy’ an action.

### Example Python Code

```

1  import os
2  import sys
3  sys.dont_write_bytecode = True
4
5  # Determine our script name, script dir
6  my_file = os.path.abspath(sys.argv[0])
7  my_dir = os.path.dirname(my_file)
8
9  # determine the pytan lib dir and add it to the path
10 parent_dir = os.path.dirname(my_dir)
11 pytan_root_dir = os.path.dirname(parent_dir)
12 lib_dir = os.path.join(pytan_root_dir, 'lib')
13 path_adds = [lib_dir]
14

```

```
15 for aa in path_adds:
16     if aa not in sys.path:
17         sys.path.append(aa)
18
19
20 # connection info for Tanium Server
21 USERNAME = "Tanium User"
22 PASSWORD = "T@n!um"
23 HOST = "172.16.31.128"
24 PORT = "444"
25
26 # Logging conrols
27 LOGLEVEL = 2
28 DEBUGFORMAT = False
29
30 import tempfile
31
32 import pytan
33 handler = pytan.Handler(
34     username=USERNAME,
35     password=PASSWORD,
36     host=HOST,
37     port=PORT,
38     loglevel=LOGLEVEL,
39     debugformat=DEBUGFORMAT,
40 )
41
42 print handler
43
44 # set the attribute name and value we want to add to the original object (if any)
45 attr_name = ""
46 attr_add = ""
47
48 # delete object before creating it?
49 delete = False
50
51 # setup the arguments for getting an object to export as json file
52 get_kwargs = {}
53 get_kwargs["objtype"] = u'action'
54 get_kwargs["id"] = 1
55
56
57 # get objects to use as an export to JSON file
58 orig_objs = handler.get(**get_kwargs)
59
60 # if attr_name and attr_add exists, modify the orig_objs to add attr_add to the attribute
61 # attr_name
62 if attr_name:
63     for x in orig_objs:
64         new_attr = getattr(x, attr_name)
65         new_attr += attr_add
66         setattr(x, attr_name, new_attr)
67         if delete:
68             # delete the object in case it already exists
69             del_kwargs = {}
70             del_kwargs[attr_name] = new_attr
71             del_kwargs['objtype'] = u'action'
72             try:
```

```

73         handler.delete(**del_kwargs)
74     except Exception as e:
75         print e
76
77     # export orig_objs to a json file
78     json_file, results = handler.export_to_report_file(
79         obj=orig_objs,
80         export_format='json',
81         report_dir=tempfile.gettempdir(),
82     )
83
84     # create the object from the exported JSON file
85     create_kwargs = {'objtype': u'action', 'json_file': json_file}
86     response = handler.create_from_json(**create_kwargs)
87
88
89     print ""
90     print "Type of response: ", type(response)
91
92     print ""
93     print "print of response:"
94     print response
95
96     print ""
97     print "print the object returned in JSON format:"
98     print response.to_json(response)

```

### Output from Python Code

```

1  Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3279
2  2015-03-26 11:49:19,488 INFO      handler: Report file '/var/folders/dk/vjrlr_c53yx6k6gyp2bbt_c40000g
3  2015-03-26 11:49:19,676 INFO      handler: New Action, name: 'Unmanaged Asset Tracking - Run Scan' (I
4
5  Type of response:  <class 'taniumpy.object_types.action_list.ActionList'>
6
7  print of response:
8  ActionList, len: 1
9
10 print the object returned in JSON format:
11 {
12     "_type": "actions",
13     "action": [
14         {
15             "_type": "action",
16             "action_group": {
17                 "_type": "group",
18                 "id": 0,
19                 "name": "Default"
20             },
21             "comment": "Scans for unmanaged assets on the network.",
22             "creation_time": "2015-03-26T15:49:19",
23             "distribute_seconds": 600,
24             "expiration_time": "2015-03-03T19:55:56",
25             "expire_seconds": 3000,
26             "history_saved_question": {
27                 "_type": "saved_question",
28                 "id": 11652

```

```
29     },
30     "id": 21080,
31     "name": "Unmanaged Asset Tracking - Run Scan",
32     "package_spec": {
33         "_type": "package_spec",
34         "command": "cmd /c start /B cscript //T:3600 ..\\..\\Tools\\run-ua-scan.vbs /RANDOM_WAIT_TIM
35         "id": 6,
36         "name": "Run Unmanaged Asset Scanner"
37     },
38     "saved_action": {
39         "_type": "saved_action",
40         "id": 14804
41     },
42     "skip_lock_flag": 0,
43     "start_time": "2015-03-03T19:05:56",
44     "status": "Expired",
45     "stopped_flag": 0,
46     "target_group": {
47         "_type": "group",
48         "id": 64,
49         "name": "Default"
50     },
51     "user": {
52         "_type": "user",
53         "group_id": 0,
54         "id": 2,
55         "last_login": "2015-03-26T08:12:04",
56         "name": "Tanium User"
57     }
58 }
59 ]
60 }
```

### Create sensor from json

Export a sensor object to a JSON file, adding ‘ API TEST’ to the name of the sensor before exporting the JSON file and deleting any pre-existing sensor with the same (new) name, then create a new sensor object from the exported JSON file

### Example Python Code

```
1 import os
2 import sys
3 sys.dont_write_bytecode = True
4
5 # Determine our script name, script dir
6 my_file = os.path.abspath(sys.argv[0])
7 my_dir = os.path.dirname(my_file)
8
9 # determine the pytan lib dir and add it to the path
10 parent_dir = os.path.dirname(my_dir)
11 pytan_root_dir = os.path.dirname(parent_dir)
12 lib_dir = os.path.join(pytan_root_dir, 'lib')
13 path_adds = [lib_dir]
14
```

```

15 for aa in path_adds:
16     if aa not in sys.path:
17         sys.path.append(aa)
18
19
20 # connection info for Tanium Server
21 USERNAME = "Tanium User"
22 PASSWORD = "T@n!um"
23 HOST = "172.16.31.128"
24 PORT = "444"
25
26 # Logging conrols
27 LOGLEVEL = 2
28 DEBUGFORMAT = False
29
30 import tempfile
31
32 import pytan
33 handler = pytan.Handler(
34     username=USERNAME,
35     password=PASSWORD,
36     host=HOST,
37     port=PORT,
38     loglevel=LOGLEVEL,
39     debugformat=DEBUGFORMAT,
40 )
41
42 print handler
43
44 # set the attribute name and value we want to add to the original object (if any)
45 attr_name = "name"
46 attr_add = " API TEST"
47
48 # delete object before creating it?
49 delete = True
50
51 # setup the arguments for getting an object to export as json file
52 get_kwargs = {}
53 get_kwargs["objtype"] = u'sensor'
54 get_kwargs["id"] = 381
55
56
57 # get objects to use as an export to JSON file
58 orig_objs = handler.get(**get_kwargs)
59
60 # if attr_name and attr_add exists, modify the orig_objs to add attr_add to the attribute
61 # attr_name
62 if attr_name:
63     for x in orig_objs:
64         new_attr = getattr(x, attr_name)
65         new_attr += attr_add
66         setattr(x, attr_name, new_attr)
67         if delete:
68             # delete the object in case it already exists
69             del_kwargs = {}
70             del_kwargs[attr_name] = new_attr
71             del_kwargs['objtype'] = u'sensor'
72             try:

```

```
73         handler.delete(**del_kwargs)
74     except Exception as e:
75         print e
76
77     # export orig_objs to a json file
78     json_file, results = handler.export_to_report_file(
79         obj=orig_objs,
80         export_format='json',
81         report_dir=tempfile.gettempdir(),
82     )
83
84     # create the object from the exported JSON file
85     create_kwargs = {'objtype': u'sensor', 'json_file': json_file}
86     response = handler.create_from_json(**create_kwargs)
87
88
89     print ""
90     print "Type of response: ", type(response)
91
92     print ""
93     print "print of response:"
94     print response
95
96     print ""
97     print "print the object returned in JSON format:"
98     print response.to_json(response)
```

### Output from Python Code

```
1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3279
2 2015-03-26 11:49:19,750 INFO      handler: Deleted 'Sensor, id: 1828'
3 2015-03-26 11:49:19,751 INFO      handler: Report file '/var/folders/dk/vjrlr_c53yx6k6gyp2bbt_c40000g
4 2015-03-26 11:49:19,821 INFO      handler: New Sensor, name: 'Folder Name Search with RegEx Match API
5
6 Type of response:  <class 'taniumpy.object_types.sensor_list.SensorList'>
7
8 print of response:
9 SensorList, len: 1
10
11 print the object returned in JSON format:
12 {
13     "_type": "sensors",
14     "sensor": [
15         {
16             "_type": "sensor",
17             "category": "File System",
18             "creation_time": "2015-03-26T15:49:19",
19             "delimiter": ",",
20             "description": "Finds the specified folder and provides the full path if the folder exists on
21             "exclude_from_parse_flag": 1,
22             "hash": 839342978,
23             "hidden_flag": 0,
24             "id": 1830,
25             "ignore_case_flag": 1,
26             "last_modified_by": "Tanium User",
27             "max_age_seconds": 600,
28             "metadata": {
```



```

29     "_type": "metadata",
30     "item": [
31         {
32             "_type": "item",
33             "admin_flag": 0,
34             "name": "defined",
35             "value": "McAfee"
36         }
37     ]
38 },
39 "modification_time": "2015-03-26T15:49:19",
40 "name": "Folder Name Search with RegEx Match API TEST",
41 "parameter_definition": "{\\"parameters\\":[{\\"label\\":\\"Search for Folder Name\\",\\"maxChars\\":0
42 "queries": {
43     "_type": "queries",
44     "query": [
45         {
46             "_type": "query",
47             "platform": "Windows",
48             "script": "&#039;=====&#039; Folder Name Se
49             "script_type": "VBScript"
50         },
51         {
52             "_type": "query",
53             "platform": "Linux",
54             "script": "#!/bin/bash\n#||dirname|||regexp|||casesensitive|||global||hecho Windows
55             "script_type": "UnixShell"
56         },
57         {
58             "_type": "query",
59             "platform": "Mac",
60             "script": "#!/bin/bash\n#||dirname|||regexp|||casesensitive|||global||hecho Windows
61             "script_type": "UnixShell"
62         }
63     ]
64 },
65 "source_id": 0,
66 "string_count": 0,
67 "value_type": "String"
68 }
69 ]
70 }

```

### Create question from json

Export a question object to a JSON file, then create a new question object from the exported JSON file. Questions can not be deleted, so do not delete it. This will, in effect, ‘re-ask’ a question.

### Example Python Code

```

1 import os
2 import sys
3 sys.dont_write_bytecode = True
4
5 # Determine our script name, script dir

```

```
6 my_file = os.path.abspath(sys.argv[0])
7 my_dir = os.path.dirname(my_file)
8
9 # determine the pytan lib dir and add it to the path
10 parent_dir = os.path.dirname(my_dir)
11 pytan_root_dir = os.path.dirname(parent_dir)
12 lib_dir = os.path.join(pytan_root_dir, 'lib')
13 path_adds = [lib_dir]
14
15 for aa in path_adds:
16     if aa not in sys.path:
17         sys.path.append(aa)
18
19
20 # connection info for Tanium Server
21 USERNAME = "Tanium User"
22 PASSWORD = "T@n!um"
23 HOST = "172.16.31.128"
24 PORT = "444"
25
26 # Logging conrols
27 LOGLEVEL = 2
28 DEBUGFORMAT = False
29
30 import tempfile
31
32 import pytan
33 handler = pytan.Handler(
34     username=USERNAME,
35     password=PASSWORD,
36     host=HOST,
37     port=PORT,
38     loglevel=LOGLEVEL,
39     debugformat=DEBUGFORMAT,
40 )
41
42 print handler
43
44 # set the attribute name and value we want to add to the original object (if any)
45 attr_name = ""
46 attr_add = ""
47
48 # delete object before creating it?
49 delete = False
50
51 # setup the arguments for getting an object to export as json file
52 get_kwargs = {}
53 get_kwargs["objtype"] = u'question'
54 get_kwargs["id"] = 1
55
56
57 # get objects to use as an export to JSON file
58 orig_objs = handler.get(**get_kwargs)
59
60 # if attr_name and attr_add exists, modify the orig_objs to add attr_add to the attribute
61 # attr_name
62 if attr_name:
63     for x in orig_objs:
```

```

64         new_attr = getattr(x, attr_name)
65         new_attr += attr_add
66         setattr(x, attr_name, new_attr)
67         if delete:
68             # delete the object in case it already exists
69             del_kwargs = {}
70             del_kwargs[attr_name] = new_attr
71             del_kwargs['objtype'] = u'question'
72             try:
73                 handler.delete(**del_kwargs)
74             except Exception as e:
75                 print e
76
77         # export orig_objs to a json file
78         json_file, results = handler.export_to_report_file(
79             obj=orig_objs,
80             export_format='json',
81             report_dir=tempfile.gettempdir(),
82         )
83
84         # create the object from the exported JSON file
85         create_kwargs = {'objtype': u'question', 'json_file': json_file}
86         response = handler.create_from_json(**create_kwargs)
87
88
89         print ""
90         print "Type of response: ", type(response)
91
92         print ""
93         print "print of response:"
94         print response
95
96         print ""
97         print "print the object returned in JSON format:"
98         print response.to_json(response)

```

### Output from Python Code

```

1  Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3279
2  2015-03-26 11:49:19,898 INFO      handler: Report file '/var/folders/dk/vjrlr_c53yx6k6gzp2bbt_c40000g
3  2015-03-26 11:49:19,928 INFO      handler: New Question, id: 32660 (ID: 32660) created successfully!
4
5  Type of response:  <class 'taniumpy.object_types.question_list.QuestionList'>
6
7  print of response:
8  QuestionList, len: 1
9
10 print the object returned in JSON format:
11 {
12     "_type": "questions",
13     "question": [
14         {
15             "_type": "question",
16             "action_tracking_flag": 0,
17             "context_group": {
18                 "_type": "group",
19                 "id": 0

```

```
20     },
21     "expiration": "2015-03-26T15:59:20",
22     "expire_seconds": 0,
23     "force_computer_id_flag": 0,
24     "hidden_flag": 0,
25     "id": 32660,
26     "management_rights_group": {
27         "_type": "group",
28         "id": 0
29     },
30     "query_text": "Get number of machines",
31     "saved_question": {
32         "_type": "saved_question",
33         "id": 0
34     },
35     "selects": {
36         "_type": "selects",
37         "select": []
38     },
39     "skip_lock_flag": 0,
40     "user": {
41         "_type": "user",
42         "id": 2,
43         "name": "Tanium User"
44     }
45 }
46 ]
47 }
```

### Create whitelisted url from json

Export a whitelisted url object to a JSON file, adding 'test1' to the url\_regex of the whitelisted url before exporting the JSON file and deleting any pre-existing whitelisted url with the same (new) name, then create a new whitelisted url object from the exported JSON file

### Example Python Code

```
1  import os
2  import sys
3  sys.dont_write_bytecode = True
4
5  # Determine our script name, script dir
6  my_file = os.path.abspath(sys.argv[0])
7  my_dir = os.path.dirname(my_file)
8
9  # determine the pytan lib dir and add it to the path
10 parent_dir = os.path.dirname(my_dir)
11 pytan_root_dir = os.path.dirname(parent_dir)
12 lib_dir = os.path.join(pytan_root_dir, 'lib')
13 path_adds = [lib_dir]
14
15 for aa in path_adds:
16     if aa not in sys.path:
17         sys.path.append(aa)
18
```

```

19
20 # connection info for Tanium Server
21 USERNAME = "Tanium User"
22 PASSWORD = "T@n!um"
23 HOST = "172.16.31.128"
24 PORT = "444"
25
26 # Logging conrols
27 LOGLEVEL = 2
28 DEBUGFORMAT = False
29
30 import tempfile
31
32 import pytan
33 handler = pytan.Handler(
34     username=USERNAME,
35     password=PASSWORD,
36     host=HOST,
37     port=PORT,
38     loglevel=LOGLEVEL,
39     debugformat=DEBUGFORMAT,
40 )
41
42 print handler
43
44 # set the attribute name and value we want to add to the original object (if any)
45 attr_name = "url_regex"
46 attr_add = " API TEST"
47
48 # delete object before creating it?
49 delete = True
50
51 # setup the arguments for getting an object to export as json file
52 get_kwargs = {}
53 get_kwargs["objtype"] = u'whitelisted_url'
54 get_kwargs["url_regex"] = u'test1'
55
56
57 # get objects to use as an export to JSON file
58 orig_objs = handler.get(**get_kwargs)
59
60 # if attr_name and attr_add exists, modify the orig_objs to add attr_add to the attribute
61 # attr_name
62 if attr_name:
63     for x in orig_objs:
64         new_attr = getattr(x, attr_name)
65         new_attr += attr_add
66         setattr(x, attr_name, new_attr)
67         if delete:
68             # delete the object in case it already exists
69             del_kwargs = {}
70             del_kwargs[attr_name] = new_attr
71             del_kwargs['objtype'] = u'whitelisted_url'
72             try:
73                 handler.delete(**del_kwargs)
74             except Exception as e:
75                 print e
76

```

```
77 # export orig_objs to a json file
78 json_file, results = handler.export_to_report_file(
79     obj=orig_objs,
80     export_format='json',
81     report_dir=tempfile.gettempdir(),
82 )
83
84 # create the object from the exported JSON file
85 create_kwargs = {'objtype': u'whitelisted_url', 'json_file': json_file}
86 response = handler.create_from_json(**create_kwargs)
87
88
89 print ""
90 print "Type of response: ", type(response)
91
92 print ""
93 print "print of response:"
94 print response
95
96 print ""
97 print "print the object returned in JSON format:"
98 print response.to_json(response)
```

### Output from Python Code

```
1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3279
2 2015-03-26 11:49:19,986 INFO handler: Deleted 'WhiteListedUrl, id: 1026'
3 2015-03-26 11:49:19,986 INFO handler: Report file '/var/folders/dk/vjrlr_c53yx6k6gzzp2bbt_c40000g
4 2015-03-26 11:49:20,006 INFO handler: New WhiteListedUrl, id: 1028 (ID: 1028) created successful
5
6 Type of response: <class 'taniumpy.object_types.white_listed_url_list.WhiteListedUrlList'>
7
8 print of response:
9 WhiteListedUrlList, len: 1
10
11 print the object returned in JSON format:
12 {
13     "_type": "white_listed_urls",
14     "white_listed_url": [
15         {
16             "_type": "white_listed_url",
17             "download_seconds": 86400,
18             "id": 1028,
19             "url_regex": "test1 API TEST"
20         }
21     ]
22 }
```

### Create group from json

Export a group object to a JSON file, adding ‘ API TEST’ to the name of the group before exporting the JSON file and deleting any pre-existing group with the same (new) name, then create a new group object from the exported JSON file

## Example Python Code

```

1  import os
2  import sys
3  sys.dont_write_bytecode = True
4
5  # Determine our script name, script dir
6  my_file = os.path.abspath(sys.argv[0])
7  my_dir = os.path.dirname(my_file)
8
9  # determine the pytan lib dir and add it to the path
10 parent_dir = os.path.dirname(my_dir)
11 pytan_root_dir = os.path.dirname(parent_dir)
12 lib_dir = os.path.join(pytan_root_dir, 'lib')
13 path_adds = [lib_dir]
14
15 for aa in path_adds:
16     if aa not in sys.path:
17         sys.path.append(aa)
18
19
20 # connection info for Tanium Server
21 USERNAME = "Tanium User"
22 PASSWORD = "T@n!um"
23 HOST = "172.16.31.128"
24 PORT = "444"
25
26 # Logging conrols
27 LOGLEVEL = 2
28 DEBUGFORMAT = False
29
30 import tempfile
31
32 import pytan
33 handler = pytan.Handler(
34     username=USERNAME,
35     password=PASSWORD,
36     host=HOST,
37     port=PORT,
38     loglevel=LOGLEVEL,
39     debugformat=DEBUGFORMAT,
40 )
41
42 print handler
43
44 # set the attribute name and value we want to add to the original object (if any)
45 attr_name = "name"
46 attr_add = " API TEST"
47
48 # delete object before creating it?
49 delete = True
50
51 # setup the arguments for getting an object to export as json file
52 get_kwargs = {}
53 get_kwargs["objtype"] = u'group'
54 get_kwargs["name"] = u'All Computers'
55
56
57 # get objects to use as an export to JSON file

```

```
58 orig_objs = handler.get(**get_kwargs)
59
60 # if attr_name and attr_add exists, modify the orig_objs to add attr_add to the attribute
61 # attr_name
62 if attr_name:
63     for x in orig_objs:
64         new_attr = getattr(x, attr_name)
65         new_attr += attr_add
66         setattr(x, attr_name, new_attr)
67         if delete:
68             # delete the object in case it already exists
69             del_kwargs = {}
70             del_kwargs[attr_name] = new_attr
71             del_kwargs['objtype'] = u'group'
72             try:
73                 handler.delete(**del_kwargs)
74             except Exception as e:
75                 print e
76
77 # export orig_objs to a json file
78 json_file, results = handler.export_to_report_file(
79     obj=orig_objs,
80     export_format='json',
81     report_dir=tempfile.gettempdir(),
82 )
83
84 # create the object from the exported JSON file
85 create_kwargs = {'objtype': u'group', 'json_file': json_file}
86 response = handler.create_from_json(**create_kwargs)
87
88
89 print ""
90 print "Type of response: ", type(response)
91
92 print ""
93 print "print of response:"
94 print response
95
96 print ""
97 print "print the object returned in JSON format:"
98 print response.to_json(response)
```

### Output from Python Code

```
1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3279
2 2015-03-26 11:49:20,053 INFO handler: Deleted 'Group, id: 19172'
3 2015-03-26 11:49:20,054 INFO handler: Report file '/var/folders/dk/vjrlr_c53yx6k6gzzp2bbt_c40000g
4 2015-03-26 11:49:20,103 INFO handler: New Group, name: 'All Computers API TEST' (ID: 19224) crea
5
6 Type of response: <class 'taniumpy.object_types.group_list.GroupList'>
7
8 print of response:
9 GroupList, len: 1
10
11 print the object returned in JSON format:
12 {
13     "_type": "groups",
```



```

14     "group": [
15         {
16             "_type": "group",
17             "and_flag": 0,
18             "deleted_flag": 0,
19             "filters": {
20                 "_type": "filters",
21                 "filter": []
22             },
23             "id": 19224,
24             "name": "All Computers API TEST",
25             "not_flag": 0,
26             "sub_groups": {
27                 "_type": "groups",
28                 "group": []
29             },
30             "type": 0
31         }
32     ]
33 }

```

## pytan API Invalid Create Object From JSON Examples

### Invalid create saved action from json

Create a saved action from json (not supported!)

### Example Python Code

```

1  import os
2  import sys
3  sys.dont_write_bytecode = True
4
5  # Determine our script name, script dir
6  my_file = os.path.abspath(sys.argv[0])
7  my_dir = os.path.dirname(my_file)
8
9  # determine the pytan lib dir and add it to the path
10 parent_dir = os.path.dirname(my_dir)
11 pytan_root_dir = os.path.dirname(parent_dir)
12 lib_dir = os.path.join(pytan_root_dir, 'lib')
13 path_adds = [lib_dir]
14
15 for aa in path_adds:
16     if aa not in sys.path:
17         sys.path.append(aa)
18
19
20 # connection info for Tanium Server
21 USERNAME = "Tanium User"
22 PASSWORD = "T@n!um"
23 HOST = "172.16.31.128"
24 PORT = "444"
25
26 # Logging conrols

```

```
27 LOGLEVEL = 2
28 DEBUGFORMAT = False
29
30 import tempfile
31
32 import pytan
33 handler = pytan.Handler(
34     username=USERNAME,
35     password=PASSWORD,
36     host=HOST,
37     port=PORT,
38     loglevel=LOGLEVEL,
39     debugformat=DEBUGFORMAT,
40 )
41
42 print handler
43
44 # setup the arguments for getting an object to export as json file
45 get_kwargs = {}
46 get_kwargs["objtype"] = u'saved_action'
47 get_kwargs["name"] = u'Distribute Tanium Standard Utilities'
48
49 # get objects to use as an export to JSON file
50 orig_objs = handler.get(**get_kwargs)
51
52 # export orig_objs to a json file
53 json_file, results = handler.export_to_report_file(
54     obj=orig_objs,
55     export_format='json',
56     report_dir=tempfile.gettempdir(),
57 )
58
59 # call the handler with the create_from_json method, passing in kwargs for arguments
60 # this should throw an exception: pytan.utils.HandlerError
61 import traceback
62
63 # create the object from the exported JSON file
64 create_kwargs = {'objtype': u'saved_action', 'json_file': json_file}
65 try:
66     response = handler.create_from_json(**create_kwargs)
67 except Exception as e:
68     traceback.print_exc(file=sys.stdout)
```

### Output from Python Code

```
1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3279
2 2015-03-26 11:49:20,140 INFO      handler: Report file '/var/folders/dk/vjrlr_c53yx6k6gzp2bbt_c40000g
3 Traceback (most recent call last):
4   File "<string>", line 67, in <module>
5   File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 484, in create_from_json
6     raise HandlerError(m(objtype, json_createable))
7 HandlerError: saved_action is not a json createable object! Supported objects: user, whitelisted_url
```

## Invalid create client from json

Create a client from json (not supported!)

### Example Python Code

```

1  import os
2  import sys
3  sys.dont_write_bytecode = True
4
5  # Determine our script name, script dir
6  my_file = os.path.abspath(sys.argv[0])
7  my_dir = os.path.dirname(my_file)
8
9  # determine the pytan lib dir and add it to the path
10 parent_dir = os.path.dirname(my_dir)
11 pytan_root_dir = os.path.dirname(parent_dir)
12 lib_dir = os.path.join(pytan_root_dir, 'lib')
13 path_adds = [lib_dir]
14
15 for aa in path_adds:
16     if aa not in sys.path:
17         sys.path.append(aa)
18
19
20 # connection info for Tanium Server
21 USERNAME = "Tanium User"
22 PASSWORD = "T@n!um"
23 HOST = "172.16.31.128"
24 PORT = "444"
25
26 # Logging conrols
27 LOGLEVEL = 2
28 DEBUGFORMAT = False
29
30 import tempfile
31
32 import pytan
33 handler = pytan.Handler(
34     username=USERNAME,
35     password=PASSWORD,
36     host=HOST,
37     port=PORT,
38     loglevel=LOGLEVEL,
39     debugformat=DEBUGFORMAT,
40 )
41
42 print handler
43
44 # setup the arguments for getting an object to export as json file
45 get_kwargs = {}
46 get_kwargs["objtype"] = u'client'
47 get_kwargs["status"] = u'Leader'
48
49 # get objects to use as an export to JSON file
50 orig_objs = handler.get(**get_kwargs)
51
52 # export orig_objs to a json file

```

```
53 | json_file, results = handler.export_to_report_file(
54 |     obj=orig_objs,
55 |     export_format='json',
56 |     report_dir=tempfile.gettempdir(),
57 | )
58 |
59 | # call the handler with the create_from_json method, passing in kwargs for arguments
60 | # this should throw an exception: pytan.utils.HandlerError
61 | import traceback
62 |
63 | # create the object from the exported JSON file
64 | create_kwargs = {'objtype': u'client', 'json_file': json_file}
65 | try:
66 |     response = handler.create_from_json(**create_kwargs)
67 | except Exception as e:
68 |     traceback.print_exc(file=sys.stdout)
```

### Output from Python Code

```
1 | Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3279
2 | 2015-03-26 11:49:20,176 INFO      handler: Report file '/var/folders/dk/vjr1r_c53yx6k6gyp2bbt_c40000g
3 | Traceback (most recent call last):
4 |   File "<string>", line 67, in <module>
5 |   File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 484, in create_from_json
6 |     raise HandlerError(m(objtype, json_createable))
7 | HandlerError: client is not a json createable object! Supported objects: user, whitelisted_url, save
```

### Invalid create userrole from json

Create a user role from json (not supported!)

### Example Python Code

```
1 | import os
2 | import sys
3 | sys.dont_write_bytecode = True
4 |
5 | # Determine our script name, script dir
6 | my_file = os.path.abspath(sys.argv[0])
7 | my_dir = os.path.dirname(my_file)
8 |
9 | # determine the pytan lib dir and add it to the path
10 | parent_dir = os.path.dirname(my_dir)
11 | pytan_root_dir = os.path.dirname(parent_dir)
12 | lib_dir = os.path.join(pytan_root_dir, 'lib')
13 | path_adds = [lib_dir]
14 |
15 | for aa in path_adds:
16 |     if aa not in sys.path:
17 |         sys.path.append(aa)
18 |
19 |
20 | # connection info for Tanium Server
21 | USERNAME = "Tanium User"
```

```

22 PASSWORD = "T@n!um"
23 HOST = "172.16.31.128"
24 PORT = "444"
25
26 # Logging conrols
27 LOGLEVEL = 2
28 DEBUGFORMAT = False
29
30 import tempfile
31
32 import pytan
33 handler = pytan.Handler(
34     username=USERNAME,
35     password=PASSWORD,
36     host=HOST,
37     port=PORT,
38     loglevel=LOGLEVEL,
39     debugformat=DEBUGFORMAT,
40 )
41
42 print handler
43
44 # setup the arguments for getting an object to export as json file
45 get_kwargs = {}
46 get_kwargs["objtype"] = u'userrole'
47 get_kwargs["name"] = u'Administrator'
48
49 # get objects to use as an export to JSON file
50 orig_objs = handler.get(**get_kwargs)
51
52 # export orig_objs to a json file
53 json_file, results = handler.export_to_report_file(
54     obj=orig_objs,
55     export_format='json',
56     report_dir=tempfile.gettempdir(),
57 )
58
59 # call the handler with the create_from_json method, passing in kwargs for arguments
60 # this should throw an exception: pytan.utils.HandlerError
61 import traceback
62
63 # create the object from the exported JSON file
64 create_kwargs = {'objtype': u'userrole', 'json_file': json_file}
65 try:
66     response = handler.create_from_json(**create_kwargs)
67 except Exception as e:
68     traceback.print_exc(file=sys.stdout)

```

### Output from Python Code

```

1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3279
2 2015-03-26 11:49:20,204 INFO handler: Report file '/var/folders/dk/vjr1r_c53yx6k6gzzp2bbt_c40000g
3 Traceback (most recent call last):
4   File "<string>", line 67, in <module>
5   File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 484, in create_from_json
6     raise HandlerError(m(objtype, json_createable))

```

```
7 | HandlerError: userrole is not a json createable object! Supported objects: user, whitelisted_url, sa
```

### Invalid create setting from json

Create a setting from json (not supported!)

### Example Python Code

```
1 | import os
2 | import sys
3 | sys.dont_write_bytecode = True
4 |
5 | # Determine our script name, script dir
6 | my_file = os.path.abspath(sys.argv[0])
7 | my_dir = os.path.dirname(my_file)
8 |
9 | # determine the pytan lib dir and add it to the path
10 | parent_dir = os.path.dirname(my_dir)
11 | pytan_root_dir = os.path.dirname(parent_dir)
12 | lib_dir = os.path.join(pytan_root_dir, 'lib')
13 | path_adds = [lib_dir]
14 |
15 | for aa in path_adds:
16 |     if aa not in sys.path:
17 |         sys.path.append(aa)
18 |
19 |
20 | # connection info for Tanium Server
21 | USERNAME = "Tanium User"
22 | PASSWORD = "T@n!um"
23 | HOST = "172.16.31.128"
24 | PORT = "444"
25 |
26 | # Logging conrols
27 | LOGLEVEL = 2
28 | DEBUGFORMAT = False
29 |
30 | import tempfile
31 |
32 | import pytan
33 | handler = pytan.Handler(
34 |     username=USERNAME,
35 |     password=PASSWORD,
36 |     host=HOST,
37 |     port=PORT,
38 |     loglevel=LOGLEVEL,
39 |     debugformat=DEBUGFORMAT,
40 | )
41 |
42 | print handler
43 |
44 | # setup the arguments for getting an object to export as json file
45 | get_kwargs = {}
46 | get_kwargs["objtype"] = u'setting'
47 | get_kwargs["id"] = 1
```

```

48 # get objects to use as an export to JSON file
49 orig_objs = handler.get(**get_kwargs)
50
51 # export orig_objs to a json file
52 json_file, results = handler.export_to_report_file(
53     obj=orig_objs,
54     export_format='json',
55     report_dir=tempfile.gettempdir(),
56 )
57
58 # call the handler with the create_from_json method, passing in kwargs for arguments
59 # this should throw an exception: pytan.utils.HandlerError
60 import traceback
61
62 # create the object from the exported JSON file
63 create_kwargs = {'objtype': u'setting', 'json_file': json_file}
64 try:
65     response = handler.create_from_json(**create_kwargs)
66 except Exception as e:
67     traceback.print_exc(file=sys.stdout)
68

```

### Output from Python Code

```

1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3279
2 2015-03-26 11:49:20,231 INFO handler: Report file '/var/folders/dk/vjrlr_c53yx6k6gkz2p2bbt_c40000g
3 Traceback (most recent call last):
4   File "<string>", line 67, in <module>
5   File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 484, in create_from_json
6     raise HandlerError(m(objtype, json_createable))
7 HandlerError: setting is not a json createable object! Supported objects: user, whitelisted_url, sav

```

## pytan API Valid Export ResultSet Examples

### Export resultset csv default options

Export a ResultSet from asking a question as CSV with the default options

### Example Python Code

```

1 import os
2 import sys
3 sys.dont_write_bytecode = True
4
5 # Determine our script name, script dir
6 my_file = os.path.abspath(sys.argv[0])
7 my_dir = os.path.dirname(my_file)
8
9 # determine the pytan lib dir and add it to the path
10 parent_dir = os.path.dirname(my_dir)
11 pytan_root_dir = os.path.dirname(parent_dir)
12 lib_dir = os.path.join(pytan_root_dir, 'lib')
13 path_adds = [lib_dir]
14

```

```
15 for aa in path_adds:
16     if aa not in sys.path:
17         sys.path.append(aa)
18
19
20 # connection info for Tanium Server
21 USERNAME = "Tanium User"
22 PASSWORD = "T@n!um"
23 HOST = "172.16.31.128"
24 PORT = "444"
25
26 # Logging conrols
27 LOGLEVEL = 2
28 DEBUGFORMAT = False
29
30 import tempfile
31
32 import pytan
33 handler = pytan.Handler(
34     username=USERNAME,
35     password=PASSWORD,
36     host=HOST,
37     port=PORT,
38     loglevel=LOGLEVEL,
39     debugformat=DEBUGFORMAT,
40 )
41
42 print handler
43
44 # setup the export_obj kwargs for later
45 export_kwargs = {}
46 export_kwargs["export_format"] = u'csv'
47
48 # ask the question that will provide the resultset that we want to use
49 ask_kwargs = {
50     'qtype': 'manual_human',
51     'sensors': [
52         "Computer Name", "IP Route Details", "IP Address",
53         'Folder Name Search with RegEx Match{dirname=Program Files,regex=.*Shared.*}',
54     ],
55 }
56 response = handler.ask(**ask_kwargs)
57
58 # export the object to a string
59 # (we could just as easily export to a file using export_to_report_file)
60 export_kwargs['obj'] = response['question_results']
61 export_str = handler.export_obj(**export_kwargs)
62
63
64 print ""
65 print "print the export_str returned from export_obj():"
66 if len(out.splitlines()) > 15:
67     out = out.splitlines()[0:15]
68     out.append('..trimmed for brevity..')
69     out = '\n'.join(out)
70
71 print out
```



## Output from Python Code

```

1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3279
2 2015-03-26 11:49:20,488 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detai
3 2015-03-26 11:49:25,521 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detai
4 2015-03-26 11:49:30,553 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detai
5 2015-03-26 11:49:35,587 INFO      question_progress: Results 50% (Get Computer Name and IP Route Deta
6 2015-03-26 11:49:40,619 INFO      question_progress: Results 50% (Get Computer Name and IP Route Deta
7 2015-03-26 11:49:45,650 INFO      question_progress: Results 100% (Get Computer Name and IP Route Deta
8
9 print the export_str returned from export_obj():
10 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3279
11 2015-03-26 11:49:20,231 INFO      handler: Report file '/var/folders/dk/vjrlr_c53yx6k6g2p2bbt_c40000g
12 Traceback (most recent call last):
13   File "<string>", line 67, in <module>
14   File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 484, in create_from_json
15     raise HandlerError(m(objtype, json_createable))
16 HandlerError: setting is not a json createable object! Supported objects: user, whitelisted_url, sav

```

## Export resultset csv expand false

Export a ResultSet from asking a question as CSV with false for expand\_grouped\_columns

## Example Python Code

```

1 import os
2 import sys
3 sys.dont_write_bytecode = True
4
5 # Determine our script name, script dir
6 my_file = os.path.abspath(sys.argv[0])
7 my_dir = os.path.dirname(my_file)
8
9 # determine the pytan lib dir and add it to the path
10 parent_dir = os.path.dirname(my_dir)
11 pytan_root_dir = os.path.dirname(parent_dir)
12 lib_dir = os.path.join(pytan_root_dir, 'lib')
13 path_adds = [lib_dir]
14
15 for aa in path_adds:
16     if aa not in sys.path:
17         sys.path.append(aa)
18
19
20 # connection info for Tanium Server
21 USERNAME = "Tanium User"
22 PASSWORD = "T@n!um"
23 HOST = "172.16.31.128"
24 PORT = "444"
25
26 # Logging conrols
27 LOGLEVEL = 2
28 DEBUGFORMAT = False
29
30 import tempfile
31

```

```

32 import pytan
33 handler = pytan.Handler(
34     username=USERNAME,
35     password=PASSWORD,
36     host=HOST,
37     port=PORT,
38     loglevel=LOGLEVEL,
39     debugformat=DEBUGFORMAT,
40 )
41
42 print handler
43
44 # setup the export_obj kwargs for later
45 export_kwargs = {}
46 export_kwargs["export_format"] = u'csv'
47 export_kwargs["expand_grouped_columns"] = False
48
49 # ask the question that will provide the resultset that we want to use
50 ask_kwargs = {
51     'qtype': 'manual_human',
52     'sensors': [
53         "Computer Name", "IP Route Details", "IP Address",
54         'Folder Name Search with RegEx Match{dirname=Program Files,regex=.*Shared.*}',
55     ],
56 }
57 response = handler.ask(**ask_kwargs)
58
59 # export the object to a string
60 # (we could just as easily export to a file using export_to_report_file)
61 export_kwargs['obj'] = response['question_results']
62 export_str = handler.export_obj(**export_kwargs)
63
64
65 print ""
66 print "print the export_str returned from export_obj():"
67 if len(out.splitlines()) > 15:
68     out = out.splitlines()[0:15]
69     out.append('..trimmed for brevity..')
70     out = '\n'.join(out)
71
72 print out

```

### Output from Python Code

```

1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3279
2 2015-03-26 11:49:45,896 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detail
3 2015-03-26 11:49:50,933 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detail
4 2015-03-26 11:49:55,968 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detail
5 2015-03-26 11:50:00,998 INFO      question_progress: Results 100% (Get Computer Name and IP Route Deta
6
7 print the export_str returned from export_obj():
8 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3279
9 2015-03-26 11:49:20,488 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detail
10 2015-03-26 11:49:25,521 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detail
11 2015-03-26 11:49:30,553 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detail
12 2015-03-26 11:49:35,587 INFO      question_progress: Results 50% (Get Computer Name and IP Route Deta
13 2015-03-26 11:49:40,619 INFO      question_progress: Results 50% (Get Computer Name and IP Route Deta

```

```

14 2015-03-26 11:49:45,650 INFO      question_progress: Results 100% (Get Computer Name and IP Route Det
15
16 print the export_str returned from export_obj():
17 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3279
18 2015-03-26 11:49:20,231 INFO      handler: Report file '/var/folders/dk/vjr1r_c53yx6k6gzp2bbt_c40000q
19 Traceback (most recent call last):
20   File "<string>", line 67, in <module>
21   File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 484, in create_from_json
22     raise HandlerError(m(objtype, json_createable))
23 ..trimmed for brevity..

```

### Export resultset csv expand true

Export a ResultSet from asking a question as CSV with true for expand\_grouped\_columns

### Example Python Code

```

1  import os
2  import sys
3  sys.dont_write_bytecode = True
4
5  # Determine our script name, script dir
6  my_file = os.path.abspath(sys.argv[0])
7  my_dir = os.path.dirname(my_file)
8
9  # determine the pytan lib dir and add it to the path
10 parent_dir = os.path.dirname(my_dir)
11 pytan_root_dir = os.path.dirname(parent_dir)
12 lib_dir = os.path.join(pytan_root_dir, 'lib')
13 path_adds = [lib_dir]
14
15 for aa in path_adds:
16     if aa not in sys.path:
17         sys.path.append(aa)
18
19
20 # connection info for Tanium Server
21 USERNAME = "Tanium User"
22 PASSWORD = "T@n!um"
23 HOST = "172.16.31.128"
24 PORT = "444"
25
26 # Logging conrols
27 LOGLEVEL = 2
28 DEBUGFORMAT = False
29
30 import tempfile
31
32 import pytan
33 handler = pytan.Handler(
34     username=USERNAME,
35     password=PASSWORD,
36     host=HOST,
37     port=PORT,
38     loglevel=LOGLEVEL,

```

```

39     debugformat=DEBUGFORMAT,
40 )
41
42 print handler
43
44 # setup the export_obj kwargs for later
45 export_kwargs = {}
46 export_kwargs["export_format"] = u'csv'
47 export_kwargs["expand_grouped_columns"] = True
48
49 # ask the question that will provide the resultset that we want to use
50 ask_kwargs = {
51     'qtype': 'manual_human',
52     'sensors': [
53         "Computer Name", "IP Route Details", "IP Address",
54         'Folder Name Search with RegEx Match{dirname=Program Files,regex=.*Shared.*}',
55     ],
56 }
57 response = handler.ask(**ask_kwargs)
58
59 # export the object to a string
60 # (we could just as easily export to a file using export_to_report_file)
61 export_kwargs['obj'] = response['question_results']
62 export_str = handler.export_obj(**export_kwargs)
63
64
65 print ""
66 print "print the export_str returned from export_obj():"
67 if len(out.splitlines()) > 15:
68     out = out.splitlines()[0:15]
69     out.append('..trimmed for brevity..')
70     out = '\n'.join(out)
71
72 print out

```

## Output from Python Code

```

1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3279
2 2015-03-26 11:50:01,247 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
3 2015-03-26 11:50:06,286 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
4 2015-03-26 11:50:11,321 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
5 2015-03-26 11:50:16,357 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
6 2015-03-26 11:50:21,386 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
7 2015-03-26 11:50:26,420 INFO question_progress: Results 50% (Get Computer Name and IP Route Deta
8 2015-03-26 11:50:31,453 INFO question_progress: Results 100% (Get Computer Name and IP Route Det
9
10 print the export_str returned from export_obj():
11 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3279
12 2015-03-26 11:49:45,896 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
13 2015-03-26 11:49:50,933 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
14 2015-03-26 11:49:55,968 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
15 2015-03-26 11:50:00,998 INFO question_progress: Results 100% (Get Computer Name and IP Route Det
16
17 print the export_str returned from export_obj():
18 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3279
19 2015-03-26 11:49:20,488 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
20 2015-03-26 11:49:25,521 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail

```

```

21 2015-03-26 11:49:30,553 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detai
22 2015-03-26 11:49:35,587 INFO      question_progress: Results 50% (Get Computer Name and IP Route Deta
23 2015-03-26 11:49:40,619 INFO      question_progress: Results 50% (Get Computer Name and IP Route Deta
24 2015-03-26 11:49:45,650 INFO      question_progress: Results 100% (Get Computer Name and IP Route Deta
25
26 ..trimmed for brevity..

```

### Export resultset csv all options

Export a ResultSet from asking a question as CSV with true for header\_add\_sensor, true for header\_add\_type, true for header\_sort, and true for expand\_grouped\_columns

### Example Python Code

```

1  import os
2  import sys
3  sys.dont_write_bytecode = True
4
5  # Determine our script name, script dir
6  my_file = os.path.abspath(sys.argv[0])
7  my_dir = os.path.dirname(my_file)
8
9  # determine the pytan lib dir and add it to the path
10 parent_dir = os.path.dirname(my_dir)
11 pytan_root_dir = os.path.dirname(parent_dir)
12 lib_dir = os.path.join(pytan_root_dir, 'lib')
13 path_adds = [lib_dir]
14
15 for aa in path_adds:
16     if aa not in sys.path:
17         sys.path.append(aa)
18
19
20 # connection info for Tanium Server
21 USERNAME = "Tanium User"
22 PASSWORD = "T@n!um"
23 HOST = "172.16.31.128"
24 PORT = "444"
25
26 # Logging conrols
27 LOGLEVEL = 2
28 DEBUGFORMAT = False
29
30 import tempfile
31
32 import pytan
33 handler = pytan.Handler(
34     username=USERNAME,
35     password=PASSWORD,
36     host=HOST,
37     port=PORT,
38     loglevel=LOGLEVEL,
39     debugformat=DEBUGFORMAT,
40 )
41

```

```

42 print handler
43
44 # setup the export_obj kwargs for later
45 export_kwargs = {}
46 export_kwargs["header_sort"] = True
47 export_kwargs["export_format"] = u'csv'
48 export_kwargs["header_add_type"] = True
49 export_kwargs["expand_grouped_columns"] = True
50 export_kwargs["header_add_sensor"] = True
51
52 # ask the question that will provide the resultset that we want to use
53 ask_kwargs = {
54     'qtype': 'manual_human',
55     'sensors': [
56         "Computer Name", "IP Route Details", "IP Address",
57         'Folder Name Search with RegEx Match{dirname=Program Files,regex=.*Shared.*}',
58     ],
59 }
60 response = handler.ask(**ask_kwargs)
61
62 # export the object to a string
63 # (we could just as easily export to a file using export_to_report_file)
64 export_kwargs['obj'] = response['question_results']
65 export_str = handler.export_obj(**export_kwargs)
66
67
68 print ""
69 print "print the export_str returned from export_obj():"
70 if len(out.splitlines()) > 15:
71     out = out.splitlines()[0:15]
72     out.append('..trimmed for brevity..')
73     out = '\n'.join(out)
74
75 print out

```

## Output from Python Code

```

1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3279
2 2015-03-26 11:50:31,786 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
3 2015-03-26 11:50:36,818 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
4 2015-03-26 11:50:41,852 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
5 2015-03-26 11:50:46,900 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
6 2015-03-26 11:50:51,932 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
7 2015-03-26 11:50:56,967 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
8 2015-03-26 11:51:01,991 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
9 2015-03-26 11:51:07,023 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
10 2015-03-26 11:51:12,051 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
11 2015-03-26 11:51:17,078 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
12 2015-03-26 11:51:22,110 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
13 2015-03-26 11:51:27,141 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
14 2015-03-26 11:51:32,176 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
15 2015-03-26 11:51:37,210 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
16 2015-03-26 11:51:42,245 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
17 2015-03-26 11:51:47,275 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
18 2015-03-26 11:51:52,304 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
19 2015-03-26 11:51:57,331 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
20 2015-03-26 11:52:02,368 INFO question_progress: Results 100% (Get Computer Name and IP Route Det

```

```

21
22 print the export_str returned from export_obj():
23 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3279
24 2015-03-26 11:50:01,247 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detail
25 2015-03-26 11:50:06,286 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detail
26 2015-03-26 11:50:11,321 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detail
27 2015-03-26 11:50:16,357 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detail
28 2015-03-26 11:50:21,386 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detail
29 2015-03-26 11:50:26,420 INFO      question_progress: Results 50% (Get Computer Name and IP Route Detail
30 2015-03-26 11:50:31,453 INFO      question_progress: Results 100% (Get Computer Name and IP Route Detail
31
32 print the export_str returned from export_obj():
33 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3279
34 2015-03-26 11:49:45,896 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detail
35 2015-03-26 11:49:50,933 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detail
36 2015-03-26 11:49:55,968 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detail
37 2015-03-26 11:50:00,998 INFO      question_progress: Results 100% (Get Computer Name and IP Route Detail
38 ..trimmed for brevity..

```

## Export resultset json

Export a ResultSet from asking a question as JSON with the default options

## Example Python Code

```

1  import os
2  import sys
3  sys.dont_write_bytecode = True
4
5  # Determine our script name, script dir
6  my_file = os.path.abspath(sys.argv[0])
7  my_dir = os.path.dirname(my_file)
8
9  # determine the pytan lib dir and add it to the path
10 parent_dir = os.path.dirname(my_dir)
11 pytan_root_dir = os.path.dirname(parent_dir)
12 lib_dir = os.path.join(pytan_root_dir, 'lib')
13 path_adds = [lib_dir]
14
15 for aa in path_adds:
16     if aa not in sys.path:
17         sys.path.append(aa)
18
19
20 # connection info for Tanium Server
21 USERNAME = "Tanium User"
22 PASSWORD = "T@n!um"
23 HOST = "172.16.31.128"
24 PORT = "444"
25
26 # Logging conrols
27 LOGLEVEL = 2
28 DEBUGFORMAT = False
29
30 import tempfile

```

```

31
32 import pytan
33 handler = pytan.Handler(
34     username=USERNAME,
35     password=PASSWORD,
36     host=HOST,
37     port=PORT,
38     loglevel=LOGLEVEL,
39     debugformat=DEBUGFORMAT,
40 )
41
42 print handler
43
44 # setup the export_obj kwargs for later
45 export_kwargs = {}
46 export_kwargs["export_format"] = u'json'
47
48 # ask the question that will provide the resultset that we want to use
49 ask_kwargs = {
50     'qtype': 'manual_human',
51     'sensors': [
52         "Computer Name", "IP Route Details", "IP Address",
53         'Folder Name Search with RegEx Match{dirname=Program Files,regex=.*Shared.*}',
54     ],
55 }
56 response = handler.ask(**ask_kwargs)
57
58 # export the object to a string
59 # (we could just as easily export to a file using export_to_report_file)
60 export_kwargs['obj'] = response['question_results']
61 export_str = handler.export_obj(**export_kwargs)
62
63
64 print ""
65 print "print the export_str returned from export_obj():"
66 if len(out.splitlines()) > 15:
67     out = out.splitlines()[0:15]
68     out.append('..trimmed for brevity..')
69     out = '\n'.join(out)
70
71 print out

```

### Output from Python Code

```

1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3279
2 2015-03-26 11:52:02,884 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detail
3 2015-03-26 11:52:07,914 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detail
4 2015-03-26 11:52:12,944 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detail
5 2015-03-26 11:52:17,971 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detail
6 2015-03-26 11:52:23,004 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detail
7 2015-03-26 11:52:28,038 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detail
8 2015-03-26 11:52:33,071 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detail
9 2015-03-26 11:52:38,103 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detail
10 2015-03-26 11:52:43,142 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detail
11 2015-03-26 11:52:48,167 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detail
12 2015-03-26 11:52:53,194 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detail
13 2015-03-26 11:52:58,227 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detail

```



```

14 2015-03-26 11:53:03,258 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detail
15 2015-03-26 11:53:08,294 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detail
16 2015-03-26 11:53:13,322 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detail
17 2015-03-26 11:53:18,352 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detail
18 2015-03-26 11:53:23,384 INFO      question_progress: Results 100% (Get Computer Name and IP Route Detail
19
20 print the export_str returned from export_obj():
21 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3279
22 2015-03-26 11:50:31,786 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detail
23 2015-03-26 11:50:36,818 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detail
24 2015-03-26 11:50:41,852 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detail
25 2015-03-26 11:50:46,900 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detail
26 2015-03-26 11:50:51,932 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detail
27 2015-03-26 11:50:56,967 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detail
28 2015-03-26 11:51:01,991 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detail
29 2015-03-26 11:51:07,023 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detail
30 2015-03-26 11:51:12,051 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detail
31 2015-03-26 11:51:17,078 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detail
32 2015-03-26 11:51:22,110 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detail
33 2015-03-26 11:51:27,141 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detail
34 2015-03-26 11:51:32,176 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detail
35 2015-03-26 11:51:37,210 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detail
36 ..trimmed for brevity..

```

### Export resultset csv sort empty

Export a ResultSet from asking a question as CSV with an empty list for header\_sort

### Example Python Code

```

1  import os
2  import sys
3  sys.dont_write_bytecode = True
4
5  # Determine our script name, script dir
6  my_file = os.path.abspath(sys.argv[0])
7  my_dir = os.path.dirname(my_file)
8
9  # determine the pytan lib dir and add it to the path
10 parent_dir = os.path.dirname(my_dir)
11 pytan_root_dir = os.path.dirname(parent_dir)
12 lib_dir = os.path.join(pytan_root_dir, 'lib')
13 path_adds = [lib_dir]
14
15 for aa in path_adds:
16     if aa not in sys.path:
17         sys.path.append(aa)
18
19
20 # connection info for Tanium Server
21 USERNAME = "Tanium User"
22 PASSWORD = "T@n!um"
23 HOST = "172.16.31.128"
24 PORT = "444"
25

```

```
26 # Logging conrols
27 LOGLEVEL = 2
28 DEBUGFORMAT = False
29
30 import tempfile
31
32 import pytan
33 handler = pytan.Handler(
34     username=USERNAME,
35     password=PASSWORD,
36     host=HOST,
37     port=PORT,
38     loglevel=LOGLEVEL,
39     debugformat=DEBUGFORMAT,
40 )
41
42 print handler
43
44 # setup the export_obj kwargs for later
45 export_kwargs = {}
46 export_kwargs["export_format"] = u'csv'
47 export_kwargs["header_sort"] = []
48
49 # ask the question that will provide the resultset that we want to use
50 ask_kwargs = {
51     'qtype': 'manual_human',
52     'sensors': [
53         "Computer Name", "IP Route Details", "IP Address",
54         'Folder Name Search with RegEx Match{dirname=Program Files,regex=.*Shared.*}',
55     ],
56 }
57 response = handler.ask(**ask_kwargs)
58
59 # export the object to a string
60 # (we could just as easily export to a file using export_to_report_file)
61 export_kwargs['obj'] = response['question_results']
62 export_str = handler.export_obj(**export_kwargs)
63
64
65 print ""
66 print "print the export_str returned from export_obj():"
67 if len(out.splitlines()) > 15:
68     out = out.splitlines()[0:15]
69     out.append('..trimmed for brevity..')
70     out = '\n'.join(out)
71
72 print out
```

### Output from Python Code

```
1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3279
2 2015-03-26 11:53:23,634 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detai
3 2015-03-26 11:53:28,666 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detai
4 2015-03-26 11:53:33,702 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detai
5 2015-03-26 11:53:38,738 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detai
6 2015-03-26 11:53:43,777 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detai
7 2015-03-26 11:53:48,810 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detai
```

```

8 2015-03-26 11:53:53,840 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
9 2015-03-26 11:53:58,865 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
10 2015-03-26 11:54:03,896 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
11 2015-03-26 11:54:08,930 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
12 2015-03-26 11:54:13,966 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
13 2015-03-26 11:54:18,999 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
14 2015-03-26 11:54:24,033 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
15 2015-03-26 11:54:29,065 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
16 2015-03-26 11:54:34,092 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
17 2015-03-26 11:54:39,120 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
18 2015-03-26 11:54:44,149 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
19 2015-03-26 11:54:49,182 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
20 2015-03-26 11:54:54,211 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
21 2015-03-26 11:54:59,248 INFO question_progress: Results 100% (Get Computer Name and IP Route Detail
22
23 print the export_str returned from export_obj():
24 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3279
25 2015-03-26 11:52:02,884 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
26 2015-03-26 11:52:07,914 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
27 2015-03-26 11:52:12,944 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
28 2015-03-26 11:52:17,971 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
29 2015-03-26 11:52:23,004 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
30 2015-03-26 11:52:28,038 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
31 2015-03-26 11:52:33,071 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
32 2015-03-26 11:52:38,103 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
33 2015-03-26 11:52:43,142 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
34 2015-03-26 11:52:48,167 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
35 2015-03-26 11:52:53,194 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
36 2015-03-26 11:52:58,227 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
37 2015-03-26 11:53:03,258 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
38 2015-03-26 11:53:08,294 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
39 ..trimmed for brevity..

```

### Export resultset csv sort true

Export a ResultSet from asking a question as CSV with true for header\_sort

### Example Python Code

```

1 import os
2 import sys
3 sys.dont_write_bytecode = True
4
5 # Determine our script name, script dir
6 my_file = os.path.abspath(sys.argv[0])
7 my_dir = os.path.dirname(my_file)
8
9 # determine the pytan lib dir and add it to the path
10 parent_dir = os.path.dirname(my_dir)
11 pytan_root_dir = os.path.dirname(parent_dir)
12 lib_dir = os.path.join(pytan_root_dir, 'lib')
13 path_adds = [lib_dir]
14
15 for aa in path_adds:
16     if aa not in sys.path:

```

```
17         sys.path.append(aa)
18
19
20     # connection info for Tanium Server
21     USERNAME = "Tanium User"
22     PASSWORD = "T@n!um"
23     HOST = "172.16.31.128"
24     PORT = "444"
25
26     # Logging conrols
27     LOGLEVEL = 2
28     DEBUGFORMAT = False
29
30     import tempfile
31
32     import pytan
33     handler = pytan.Handler(
34         username=USERNAME,
35         password=PASSWORD,
36         host=HOST,
37         port=PORT,
38         loglevel=LOGLEVEL,
39         debugformat=DEBUGFORMAT,
40     )
41
42     print handler
43
44     # setup the export_obj kwargs for later
45     export_kwargs = {}
46     export_kwargs["export_format"] = u'csv'
47     export_kwargs["header_sort"] = True
48
49     # ask the question that will provide the resultset that we want to use
50     ask_kwargs = {
51         'qtype': 'manual_human',
52         'sensors': [
53             "Computer Name", "IP Route Details", "IP Address",
54             'Folder Name Search with RegEx Match{dirname=Program Files,regex=.*Shared.*}',
55         ],
56     }
57     response = handler.ask(**ask_kwargs)
58
59     # export the object to a string
60     # (we could just as easily export to a file using export_to_report_file)
61     export_kwargs['obj'] = response['question_results']
62     export_str = handler.export_obj(**export_kwargs)
63
64
65     print ""
66     print "print the export_str returned from export_obj():"
67     if len(out.splitlines()) > 15:
68         out = out.splitlines()[0:15]
69         out.append('..trimmed for brevity..')
70         out = '\n'.join(out)
71
72     print out
```

## Output from Python Code

```

1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3279
2 2015-03-26 11:54:59,504 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detai
3 2015-03-26 11:55:04,536 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detai
4 2015-03-26 11:55:09,566 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detai
5 2015-03-26 11:55:14,596 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detai
6 2015-03-26 11:55:19,631 INFO      question_progress: Results 50% (Get Computer Name and IP Route Deta
7 2015-03-26 11:55:24,663 INFO      question_progress: Results 100% (Get Computer Name and IP Route Deta
8
9 print the export_str returned from export_obj():
10 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3279
11 2015-03-26 11:53:23,634 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detai
12 2015-03-26 11:53:28,666 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detai
13 2015-03-26 11:53:33,702 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detai
14 2015-03-26 11:53:38,738 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detai
15 2015-03-26 11:53:43,777 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detai
16 2015-03-26 11:53:48,810 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detai
17 2015-03-26 11:53:53,840 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detai
18 2015-03-26 11:53:58,865 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detai
19 2015-03-26 11:54:03,896 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detai
20 2015-03-26 11:54:08,930 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detai
21 2015-03-26 11:54:13,966 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detai
22 2015-03-26 11:54:18,999 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detai
23 2015-03-26 11:54:24,033 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detai
24 2015-03-26 11:54:29,065 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detai
25 ..trimmed for brevity..

```

## Export resultset csv sort false

Export a ResultSet from asking a question as CSV with false for header\_sort

## Example Python Code

```

1 import os
2 import sys
3 sys.dont_write_bytecode = True
4
5 # Determine our script name, script dir
6 my_file = os.path.abspath(sys.argv[0])
7 my_dir = os.path.dirname(my_file)
8
9 # determine the pytan lib dir and add it to the path
10 parent_dir = os.path.dirname(my_dir)
11 pytan_root_dir = os.path.dirname(parent_dir)
12 lib_dir = os.path.join(pytan_root_dir, 'lib')
13 path_adds = [lib_dir]
14
15 for aa in path_adds:
16     if aa not in sys.path:
17         sys.path.append(aa)
18
19
20 # connection info for Tanium Server
21 USERNAME = "Tanium User"
22 PASSWORD = "T@n!um"

```

```
23 HOST = "172.16.31.128"
24 PORT = "444"
25
26 # Logging conrols
27 LOGLEVEL = 2
28 DEBUGFORMAT = False
29
30 import tempfile
31
32 import pytan
33 handler = pytan.Handler(
34     username=USERNAME,
35     password=PASSWORD,
36     host=HOST,
37     port=PORT,
38     loglevel=LOGLEVEL,
39     debugformat=DEBUGFORMAT,
40 )
41
42 print handler
43
44 # setup the export_obj kwargs for later
45 export_kwargs = {}
46 export_kwargs["export_format"] = u'csv'
47 export_kwargs["header_sort"] = False
48
49 # ask the question that will provide the resultset that we want to use
50 ask_kwargs = {
51     'qtype': 'manual_human',
52     'sensors': [
53         "Computer Name", "IP Route Details", "IP Address",
54         'Folder Name Search with RegEx Match{dirname=Program Files,regex=.*Shared.*}',
55     ],
56 }
57 response = handler.ask(**ask_kwargs)
58
59 # export the object to a string
60 # (we could just as easily export to a file using export_to_report_file)
61 export_kwargs['obj'] = response['question_results']
62 export_str = handler.export_obj(**export_kwargs)
63
64
65 print ""
66 print "print the export_str returned from export_obj():"
67 if len(out.splitlines()) > 15:
68     out = out.splitlines()[0:15]
69     out.append('..trimmed for brevity..')
70     out = '\n'.join(out)
71
72 print out
```

### Output from Python Code

```
1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3279
2 2015-03-26 11:55:24,887 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detai
3 2015-03-26 11:55:29,914 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detai
4 2015-03-26 11:55:34,951 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detai
```

```

5 2015-03-26 11:55:39,977 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detail
6 2015-03-26 11:55:45,015 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detail
7 2015-03-26 11:55:50,052 INFO      question_progress: Results 50% (Get Computer Name and IP Route Detail
8 2015-03-26 11:55:55,081 INFO      question_progress: Results 100% (Get Computer Name and IP Route Detail
9
10 print the export_str returned from export_obj():
11 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3279
12 2015-03-26 11:54:59,504 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detail
13 2015-03-26 11:55:04,536 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detail
14 2015-03-26 11:55:09,566 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detail
15 2015-03-26 11:55:14,596 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detail
16 2015-03-26 11:55:19,631 INFO      question_progress: Results 50% (Get Computer Name and IP Route Detail
17 2015-03-26 11:55:24,663 INFO      question_progress: Results 100% (Get Computer Name and IP Route Detail
18
19 print the export_str returned from export_obj():
20 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3279
21 2015-03-26 11:53:23,634 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detail
22 2015-03-26 11:53:28,666 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detail
23 2015-03-26 11:53:33,702 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detail
24 2015-03-26 11:53:38,738 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detail
25 2015-03-26 11:53:43,777 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detail
26 ..trimmed for brevity..

```

### Export resultset csv sort list

Export a ResultSet from asking a question as CSV with Computer Name and IP Address for the header\_sort

### Example Python Code

```

1 import os
2 import sys
3 sys.dont_write_bytecode = True
4
5 # Determine our script name, script dir
6 my_file = os.path.abspath(sys.argv[0])
7 my_dir = os.path.dirname(my_file)
8
9 # determine the pytan lib dir and add it to the path
10 parent_dir = os.path.dirname(my_dir)
11 pytan_root_dir = os.path.dirname(parent_dir)
12 lib_dir = os.path.join(pytan_root_dir, 'lib')
13 path_adds = [lib_dir]
14
15 for aa in path_adds:
16     if aa not in sys.path:
17         sys.path.append(aa)
18
19
20 # connection info for Tanium Server
21 USERNAME = "Tanium User"
22 PASSWORD = "T@n!um"
23 HOST = "172.16.31.128"
24 PORT = "444"
25
26 # Logging conrols

```

```
27 LOGLEVEL = 2
28 DEBUGFORMAT = False
29
30 import tempfile
31
32 import pytan
33 handler = pytan.Handler(
34     username=USERNAME,
35     password=PASSWORD,
36     host=HOST,
37     port=PORT,
38     loglevel=LOGLEVEL,
39     debugformat=DEBUGFORMAT,
40 )
41
42 print handler
43
44 # setup the export_obj kwargs for later
45 export_kwargs = {}
46 export_kwargs["export_format"] = u'csv'
47 export_kwargs["header_sort"] = [u'Computer Name', u'IP Address']
48
49 # ask the question that will provide the resultset that we want to use
50 ask_kwargs = {
51     'qtype': 'manual_human',
52     'sensors': [
53         "Computer Name", "IP Route Details", "IP Address",
54         'Folder Name Search with RegEx Match{dirname=Program Files,regex=.*Shared.*}',
55     ],
56 }
57 response = handler.ask(**ask_kwargs)
58
59 # export the object to a string
60 # (we could just as easily export to a file using export_to_report_file)
61 export_kwargs['obj'] = response['question_results']
62 export_str = handler.export_obj(**export_kwargs)
63
64
65 print ""
66 print "print the export_str returned from export_obj():"
67 if len(out.splitlines()) > 15:
68     out = out.splitlines()[0:15]
69     out.append('..trimmed for brevity..')
70     out = '\n'.join(out)
71
72 print out
```

### Output from Python Code

```
1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3279
2 2015-03-26 11:55:55,299 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detai
3 2015-03-26 11:56:00,328 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detai
4 2015-03-26 11:56:05,362 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detai
5 2015-03-26 11:56:10,394 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detai
6 2015-03-26 11:56:15,425 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detai
7 2015-03-26 11:56:20,462 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detai
8 2015-03-26 11:56:25,497 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detai
```



```

9 2015-03-26 11:56:30,527 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
10 2015-03-26 11:56:35,567 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
11 2015-03-26 11:56:40,599 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
12 2015-03-26 11:56:45,629 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
13 2015-03-26 11:56:50,659 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
14 2015-03-26 11:56:55,688 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
15 2015-03-26 11:57:00,716 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
16 2015-03-26 11:57:05,746 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
17 2015-03-26 11:57:10,772 INFO question_progress: Results 100% (Get Computer Name and IP Route Detail
18
19 print the export_str returned from export_obj():
20 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3279
21 2015-03-26 11:55:24,887 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
22 2015-03-26 11:55:29,914 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
23 2015-03-26 11:55:34,951 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
24 2015-03-26 11:55:39,977 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
25 2015-03-26 11:55:45,015 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
26 2015-03-26 11:55:50,052 INFO question_progress: Results 50% (Get Computer Name and IP Route Detail
27 2015-03-26 11:55:55,081 INFO question_progress: Results 100% (Get Computer Name and IP Route Detail
28
29 print the export_str returned from export_obj():
30 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3279
31 2015-03-26 11:54:59,504 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
32 2015-03-26 11:55:04,536 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
33 2015-03-26 11:55:09,566 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
34 2015-03-26 11:55:14,596 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
35 ..trimmed for brevity..

```

### Export resultset csv type false

Export a ResultSet from asking a question as CSV with false for header\_add\_type

### Example Python Code

```

1  import os
2  import sys
3  sys.dont_write_bytecode = True
4
5  # Determine our script name, script dir
6  my_file = os.path.abspath(sys.argv[0])
7  my_dir = os.path.dirname(my_file)
8
9  # determine the pytan lib dir and add it to the path
10 parent_dir = os.path.dirname(my_dir)
11 pytan_root_dir = os.path.dirname(parent_dir)
12 lib_dir = os.path.join(pytan_root_dir, 'lib')
13 path_adds = [lib_dir]
14
15 for aa in path_adds:
16     if aa not in sys.path:
17         sys.path.append(aa)
18
19
20 # connection info for Tanium Server
21 USERNAME = "Tanium User"

```

```
22 PASSWORD = "T@n!um"
23 HOST = "172.16.31.128"
24 PORT = "444"
25
26 # Logging conrols
27 LOGLEVEL = 2
28 DEBUGFORMAT = False
29
30 import tempfile
31
32 import pytan
33 handler = pytan.Handler(
34     username=USERNAME,
35     password=PASSWORD,
36     host=HOST,
37     port=PORT,
38     loglevel=LOGLEVEL,
39     debugformat=DEBUGFORMAT,
40 )
41
42 print handler
43
44 # setup the export_obj kwargs for later
45 export_kwargs = {}
46 export_kwargs["export_format"] = u'csv'
47 export_kwargs["header_add_type"] = False
48
49 # ask the question that will provide the resultset that we want to use
50 ask_kwargs = {
51     'qtype': 'manual_human',
52     'sensors': [
53         "Computer Name", "IP Route Details", "IP Address",
54         'Folder Name Search with RegEx Match{dirname=Program Files,regex=.*Shared.*}',
55     ],
56 }
57 response = handler.ask(**ask_kwargs)
58
59 # export the object to a string
60 # (we could just as easily export to a file using export_to_report_file)
61 export_kwargs['obj'] = response['question_results']
62 export_str = handler.export_obj(**export_kwargs)
63
64
65 print ""
66 print "print the export_str returned from export_obj():"
67 if len(out.splitlines()) > 15:
68     out = out.splitlines()[0:15]
69     out.append('..trimmed for brevity..')
70     out = '\n'.join(out)
71
72 print out
```

### Output from Python Code

```
1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3279
2 2015-03-26 11:57:10,970 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detai
3 2015-03-26 11:57:16,002 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detai
```

```

4 2015-03-26 11:57:21,030 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
5 2015-03-26 11:57:26,054 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
6 2015-03-26 11:57:31,086 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
7 2015-03-26 11:57:36,115 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
8 2015-03-26 11:57:41,139 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
9 2015-03-26 11:57:46,170 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
10 2015-03-26 11:57:51,195 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
11 2015-03-26 11:57:56,223 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
12 2015-03-26 11:58:01,251 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
13 2015-03-26 11:58:06,277 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
14 2015-03-26 11:58:11,308 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
15 2015-03-26 11:58:16,332 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
16 2015-03-26 11:58:21,361 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
17 2015-03-26 11:58:26,392 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
18 2015-03-26 11:58:31,421 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
19 2015-03-26 11:58:36,449 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
20 2015-03-26 11:58:41,481 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
21 2015-03-26 11:58:46,520 INFO question_progress: Results 100% (Get Computer Name and IP Route Detail
22
23 print the export_str returned from export_obj():
24 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3279
25 2015-03-26 11:55:55,299 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
26 2015-03-26 11:56:00,328 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
27 2015-03-26 11:56:05,362 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
28 2015-03-26 11:56:10,394 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
29 2015-03-26 11:56:15,425 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
30 2015-03-26 11:56:20,462 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
31 2015-03-26 11:56:25,497 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
32 2015-03-26 11:56:30,527 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
33 2015-03-26 11:56:35,567 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
34 2015-03-26 11:56:40,599 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
35 2015-03-26 11:56:45,629 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
36 2015-03-26 11:56:50,659 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
37 2015-03-26 11:56:55,688 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
38 2015-03-26 11:57:00,716 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
39 ..trimmed for brevity..

```

### Export resultset csv type true

Export a ResultSet from asking a question as CSV with true for header\_add\_type

### Example Python Code

```

1 import os
2 import sys
3 sys.dont_write_bytecode = True
4
5 # Determine our script name, script dir
6 my_file = os.path.abspath(sys.argv[0])
7 my_dir = os.path.dirname(my_file)
8
9 # determine the pytan lib dir and add it to the path
10 parent_dir = os.path.dirname(my_dir)
11 pytan_root_dir = os.path.dirname(parent_dir)
12 lib_dir = os.path.join(pytan_root_dir, 'lib')

```

```
13 path_adds = [lib_dir]
14
15 for aa in path_adds:
16     if aa not in sys.path:
17         sys.path.append(aa)
18
19
20 # connection info for Tanium Server
21 USERNAME = "Tanium User"
22 PASSWORD = "T@n!um"
23 HOST = "172.16.31.128"
24 PORT = "444"
25
26 # Logging conrols
27 LOGLEVEL = 2
28 DEBUGFORMAT = False
29
30 import tempfile
31
32 import pytan
33 handler = pytan.Handler(
34     username=USERNAME,
35     password=PASSWORD,
36     host=HOST,
37     port=PORT,
38     loglevel=LOGLEVEL,
39     debugformat=DEBUGFORMAT,
40 )
41
42 print handler
43
44 # setup the export_obj kwargs for later
45 export_kwargs = {}
46 export_kwargs["export_format"] = u'csv'
47 export_kwargs["header_add_type"] = True
48
49 # ask the question that will provide the resultset that we want to use
50 ask_kwargs = {
51     'qtype': 'manual_human',
52     'sensors': [
53         "Computer Name", "IP Route Details", "IP Address",
54         'Folder Name Search with RegEx Match{dirname=Program Files,regex=.*Shared.*}',
55     ],
56 }
57 response = handler.ask(**ask_kwargs)
58
59 # export the object to a string
60 # (we could just as easily export to a file using export_to_report_file)
61 export_kwargs['obj'] = response['question_results']
62 export_str = handler.export_obj(**export_kwargs)
63
64
65 print ""
66 print "print the export_str returned from export_obj():"
67 if len(out.splitlines()) > 15:
68     out = out.splitlines()[0:15]
69     out.append('..trimmed for brevity..')
70     out = '\n'.join(out)
```

```
print out
```

### Output from Python Code

```

1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3279
2 2015-03-26 11:58:46,754 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detail
3 2015-03-26 11:58:51,786 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detail
4 2015-03-26 11:58:56,820 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detail
5 2015-03-26 11:59:01,853 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detail
6 2015-03-26 11:59:06,888 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detail
7 2015-03-26 11:59:11,919 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detail
8 2015-03-26 11:59:16,950 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detail
9 2015-03-26 11:59:21,981 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detail
10 2015-03-26 11:59:27,011 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detail
11 2015-03-26 11:59:32,052 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detail
12 2015-03-26 11:59:37,088 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detail
13 2015-03-26 11:59:42,140 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detail
14 2015-03-26 11:59:47,170 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detail
15 2015-03-26 11:59:52,208 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detail
16 2015-03-26 11:59:57,242 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detail
17 2015-03-26 12:00:02,270 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detail
18 2015-03-26 12:00:07,297 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detail
19 2015-03-26 12:00:12,335 INFO      question_progress: Results 50% (Get Computer Name and IP Route Detail
20 2015-03-26 12:00:17,367 INFO      question_progress: Results 50% (Get Computer Name and IP Route Detail
21 2015-03-26 12:00:22,394 INFO      question_progress: Results 50% (Get Computer Name and IP Route Detail
22 2015-03-26 12:00:27,432 INFO      question_progress: Results 100% (Get Computer Name and IP Route Detail
23
24 print the export_str returned from export_obj():
25 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3279
26 2015-03-26 11:57:10,970 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detail
27 2015-03-26 11:57:16,002 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detail
28 2015-03-26 11:57:21,030 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detail
29 2015-03-26 11:57:26,054 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detail
30 2015-03-26 11:57:31,086 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detail
31 2015-03-26 11:57:36,115 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detail
32 2015-03-26 11:57:41,139 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detail
33 2015-03-26 11:57:46,170 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detail
34 2015-03-26 11:57:51,195 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detail
35 2015-03-26 11:57:56,223 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detail
36 2015-03-26 11:58:01,251 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detail
37 2015-03-26 11:58:06,277 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detail
38 2015-03-26 11:58:11,308 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detail
39 2015-03-26 11:58:16,332 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detail
40 ..trimmed for brevity..

```

### Export resultset csv sensor false

Export a ResultSet from asking a question as CSV with false for header\_add\_sensor

### Example Python Code

```

1 import os
2 import sys

```

```
3 sys.dont_write_bytecode = True
4
5 # Determine our script name, script dir
6 my_file = os.path.abspath(sys.argv[0])
7 my_dir = os.path.dirname(my_file)
8
9 # determine the pytan lib dir and add it to the path
10 parent_dir = os.path.dirname(my_dir)
11 pytan_root_dir = os.path.dirname(parent_dir)
12 lib_dir = os.path.join(pytan_root_dir, 'lib')
13 path_adds = [lib_dir]
14
15 for aa in path_adds:
16     if aa not in sys.path:
17         sys.path.append(aa)
18
19
20 # connection info for Tanium Server
21 USERNAME = "Tanium User"
22 PASSWORD = "T@n!um"
23 HOST = "172.16.31.128"
24 PORT = "444"
25
26 # Logging conrols
27 LOGLEVEL = 2
28 DEBUGFORMAT = False
29
30 import tempfile
31
32 import pytan
33 handler = pytan.Handler(
34     username=USERNAME,
35     password=PASSWORD,
36     host=HOST,
37     port=PORT,
38     loglevel=LOGLEVEL,
39     debugformat=DEBUGFORMAT,
40 )
41
42 print handler
43
44 # setup the export_obj kwargs for later
45 export_kwargs = {}
46 export_kwargs["export_format"] = u'csv'
47 export_kwargs["header_add_sensor"] = False
48
49 # ask the question that will provide the resultset that we want to use
50 ask_kwargs = {
51     'qtype': 'manual_human',
52     'sensors': [
53         "Computer Name", "IP Route Details", "IP Address",
54         'Folder Name Search with RegEx Match{dirname=Program Files,regex=.*Shared.*}',
55     ],
56 }
57 response = handler.ask(**ask_kwargs)
58
59 # export the object to a string
60 # (we could just as easily export to a file using export_to_report_file)
```

```

61 export_kwargs['obj'] = response['question_results']
62 export_str = handler.export_obj(**export_kwargs)
63
64
65 print ""
66 print "print the export_str returned from export_obj():"
67 if len(out.splitlines()) > 15:
68     out = out.splitlines()[0:15]
69     out.append('..trimmed for brevity..')
70     out = '\n'.join(out)
71
72 print out

```

### Output from Python Code

```

1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3279
2 2015-03-26 12:00:27,763 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail)
3 2015-03-26 12:00:32,795 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail)
4 2015-03-26 12:00:37,824 INFO question_progress: Results 50% (Get Computer Name and IP Route Detail)
5 2015-03-26 12:00:42,862 INFO question_progress: Results 50% (Get Computer Name and IP Route Detail)
6 2015-03-26 12:00:47,900 INFO question_progress: Results 50% (Get Computer Name and IP Route Detail)
7 2015-03-26 12:00:52,931 INFO question_progress: Results 100% (Get Computer Name and IP Route Detail)
8
9 print the export_str returned from export_obj():
10 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3279
11 2015-03-26 11:58:46,754 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail)
12 2015-03-26 11:58:51,786 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail)
13 2015-03-26 11:58:56,820 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail)
14 2015-03-26 11:59:01,853 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail)
15 2015-03-26 11:59:06,888 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail)
16 2015-03-26 11:59:11,919 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail)
17 2015-03-26 11:59:16,950 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail)
18 2015-03-26 11:59:21,981 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail)
19 2015-03-26 11:59:27,011 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail)
20 2015-03-26 11:59:32,052 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail)
21 2015-03-26 11:59:37,088 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail)
22 2015-03-26 11:59:42,140 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail)
23 2015-03-26 11:59:47,170 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail)
24 2015-03-26 11:59:52,208 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail)
25 ..trimmed for brevity..

```

### Export resultset csv sensor true

Export a ResultSet from asking a question as CSV with true for header\_add\_sensor

### Example Python Code

```

1 import os
2 import sys
3 sys.dont_write_bytecode = True
4
5 # Determine our script name, script dir
6 my_file = os.path.abspath(sys.argv[0])
7 my_dir = os.path.dirname(my_file)

```

```
8
9  # determine the pytan lib dir and add it to the path
10 parent_dir = os.path.dirname(my_dir)
11 pytan_root_dir = os.path.dirname(parent_dir)
12 lib_dir = os.path.join(pytan_root_dir, 'lib')
13 path_adds = [lib_dir]
14
15 for aa in path_adds:
16     if aa not in sys.path:
17         sys.path.append(aa)
18
19
20 # connection info for Tanium Server
21 USERNAME = "Tanium User"
22 PASSWORD = "T@n!um"
23 HOST = "172.16.31.128"
24 PORT = "444"
25
26 # Logging conrols
27 LOGLEVEL = 2
28 DEBUGFORMAT = False
29
30 import tempfile
31
32 import pytan
33 handler = pytan.Handler(
34     username=USERNAME,
35     password=PASSWORD,
36     host=HOST,
37     port=PORT,
38     loglevel=LOGLEVEL,
39     debugformat=DEBUGFORMAT,
40 )
41
42 print handler
43
44 # setup the export_obj kwargs for later
45 export_kwargs = {}
46 export_kwargs["export_format"] = u'csv'
47 export_kwargs["header_add_sensor"] = True
48
49 # ask the question that will provide the resultset that we want to use
50 ask_kwargs = {
51     'qtype': 'manual_human',
52     'sensors': [
53         "Computer Name", "IP Route Details", "IP Address",
54         'Folder Name Search with RegEx Match{dirname=Program Files,regex=.*Shared.*}',
55     ],
56 }
57 response = handler.ask(**ask_kwargs)
58
59 # export the object to a string
60 # (we could just as easily export to a file using export_to_report_file)
61 export_kwargs['obj'] = response['question_results']
62 export_str = handler.export_obj(**export_kwargs)
63
64
65 print ""
```



```

66 print "print the export_str returned from export_obj():"
67 if len(out.splitlines()) > 15:
68     out = out.splitlines()[0:15]
69     out.append('..trimmed for brevity..')
70     out = '\n'.join(out)
71
72 print out

```

### Output from Python Code

```

1  Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3279
2  2015-03-26 12:00:53,180 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detai
3  2015-03-26 12:00:58,210 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detai
4  2015-03-26 12:01:03,247 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detai
5  2015-03-26 12:01:08,280 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detai
6  2015-03-26 12:01:13,313 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detai
7  2015-03-26 12:01:18,346 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detai
8  2015-03-26 12:01:23,379 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detai
9  2015-03-26 12:01:28,408 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detai
10 2015-03-26 12:01:33,441 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detai
11 2015-03-26 12:01:38,481 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detai
12 2015-03-26 12:01:43,510 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detai
13 2015-03-26 12:01:48,536 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detai
14 2015-03-26 12:01:53,566 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detai
15 2015-03-26 12:01:58,593 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detai
16 2015-03-26 12:02:03,619 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detai
17 2015-03-26 12:02:08,650 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detai
18 2015-03-26 12:02:13,682 INFO      question_progress: Results 50% (Get Computer Name and IP Route Deta
19 2015-03-26 12:02:18,713 INFO      question_progress: Results 50% (Get Computer Name and IP Route Deta
20 2015-03-26 12:02:23,740 INFO      question_progress: Results 50% (Get Computer Name and IP Route Deta
21 2015-03-26 12:02:28,771 INFO      question_progress: Results 50% (Get Computer Name and IP Route Deta
22 2015-03-26 12:02:33,801 INFO      question_progress: Results 50% (Get Computer Name and IP Route Deta
23 2015-03-26 12:02:38,826 INFO      question_progress: Results 100% (Get Computer Name and IP Route Det
24
25 print the export_str returned from export_obj():
26 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3279
27 2015-03-26 12:00:27,763 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detai
28 2015-03-26 12:00:32,795 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detai
29 2015-03-26 12:00:37,824 INFO      question_progress: Results 50% (Get Computer Name and IP Route Deta
30 2015-03-26 12:00:42,862 INFO      question_progress: Results 50% (Get Computer Name and IP Route Deta
31 2015-03-26 12:00:47,900 INFO      question_progress: Results 50% (Get Computer Name and IP Route Deta
32 2015-03-26 12:00:52,931 INFO      question_progress: Results 100% (Get Computer Name and IP Route Det
33
34 print the export_str returned from export_obj():
35 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3279
36 2015-03-26 11:58:46,754 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detai
37 2015-03-26 11:58:51,786 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detai
38 2015-03-26 11:58:56,820 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detai
39 2015-03-26 11:59:01,853 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detai
40 2015-03-26 11:59:06,888 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detai
41 ..trimmed for brevity..

```

## pytan API Invalid Export ResultSet Examples

### Invalid export resultset csv bad sort sub type

Export a ResultSet from asking a question using a bad header\_sort

#### Example Python Code

```
1  import os
2  import sys
3  sys.dont_write_bytecode = True
4
5  # Determine our script name, script dir
6  my_file = os.path.abspath(sys.argv[0])
7  my_dir = os.path.dirname(my_file)
8
9  # determine the pytan lib dir and add it to the path
10 parent_dir = os.path.dirname(my_dir)
11 pytan_root_dir = os.path.dirname(parent_dir)
12 lib_dir = os.path.join(pytan_root_dir, 'lib')
13 path_adds = [lib_dir]
14
15 for aa in path_adds:
16     if aa not in sys.path:
17         sys.path.append(aa)
18
19
20 # connection info for Tanium Server
21 USERNAME = "Tanium User"
22 PASSWORD = "T@n!um"
23 HOST = "172.16.31.128"
24 PORT = "444"
25
26 # Logging conrols
27 LOGLEVEL = 2
28 DEBUGFORMAT = False
29
30 import tempfile
31
32 import pytan
33 handler = pytan.Handler(
34     username=USERNAME,
35     password=PASSWORD,
36     host=HOST,
37     port=PORT,
38     loglevel=LOGLEVEL,
39     debugformat=DEBUGFORMAT,
40 )
41
42 print handler
43
44 # setup the export_obj kwargs for later
45 export_kwargs = {}
46 export_kwargs["export_format"] = u'csv'
47 export_kwargs["header_sort"] = [[]]
48
49 # ask the question that will provide the resultset that we want to use
```

```

50 ask_kwargs = {
51     'qtype': 'manual_human',
52     'sensors': [
53         "Computer Name"
54     ],
55 }
56 response = handler.ask(**ask_kwargs)
57 export_kwargs['obj'] = response['question_results']
58
59 # export the object to a string
60 # this should throw an exception: pytan.utils.HandlerError
61 import traceback
62
63 try:
64     handler.export_obj(**export_kwargs)
65 except Exception as e:
66     traceback.print_exc(file=sys.stdout)

```

### Output from Python Code

```

1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3279
2 2015-03-26 12:02:39,033 INFO      question_progress: Results 0% (Get Computer Name from all machines)
3 2015-03-26 12:02:44,049 INFO      question_progress: Results 100% (Get Computer Name from all machines)
4 Traceback (most recent call last):
5   File "<string>", line 65, in <module>
6   File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 1435, in export_obj
7     utils.check_dictkey(**check_args)
8   File "/Users/jolsen/gh/pytan/lib/pytan/utils.py", line 2531, in check_dictkey
9     raise HandlerError(err(key, valid_list_types, list_types))
10 HandlerError: 'header_sort' must be a list of [<type 'str'>, <type 'unicode'>], you supplied [<type

```

### Invalid export resultset csv bad sort type

Export a ResultSet from asking a question using a bad header\_sort

### Example Python Code

```

1 import os
2 import sys
3 sys.dont_write_bytecode = True
4
5 # Determine our script name, script dir
6 my_file = os.path.abspath(sys.argv[0])
7 my_dir = os.path.dirname(my_file)
8
9 # determine the pytan lib dir and add it to the path
10 parent_dir = os.path.dirname(my_dir)
11 pytan_root_dir = os.path.dirname(parent_dir)
12 lib_dir = os.path.join(pytan_root_dir, 'lib')
13 path_adds = [lib_dir]
14
15 for aa in path_adds:
16     if aa not in sys.path:
17         sys.path.append(aa)

```

```
18
19
20 # connection info for Tanium Server
21 USERNAME = "Tanium User"
22 PASSWORD = "T@n!um"
23 HOST = "172.16.31.128"
24 PORT = "444"
25
26 # Logging conrols
27 LOGLEVEL = 2
28 DEBUGFORMAT = False
29
30 import tempfile
31
32 import pytan
33 handler = pytan.Handler(
34     username=USERNAME,
35     password=PASSWORD,
36     host=HOST,
37     port=PORT,
38     loglevel=LOGLEVEL,
39     debugformat=DEBUGFORMAT,
40 )
41
42 print handler
43
44 # setup the export_obj kwargs for later
45 export_kwargs = {}
46 export_kwargs["export_format"] = u'csv'
47 export_kwargs["header_sort"] = u'bad'
48
49 # ask the question that will provide the resultset that we want to use
50 ask_kwargs = {
51     'qtype': 'manual_human',
52     'sensors': [
53         "Computer Name"
54     ],
55 }
56 response = handler.ask(**ask_kwargs)
57 export_kwargs['obj'] = response['question_results']
58
59 # export the object to a string
60 # this should throw an exception: pytan.utils.HandlerError
61 import traceback
62
63 try:
64     handler.export_obj(**export_kwargs)
65 except Exception as e:
66     traceback.print_exc(file=sys.stdout)
```

### Output from Python Code

```
1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3279
2 2015-03-26 12:02:44,126 INFO      question_progress: Results 0% (Get Computer Name from all machines)
3 2015-03-26 12:02:49,143 INFO      question_progress: Results 0% (Get Computer Name from all machines)
4 2015-03-26 12:02:54,155 INFO      question_progress: Results 100% (Get Computer Name from all machine
5 Traceback (most recent call last):
```

```

6   File "<string>", line 65, in <module>
7   File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 1435, in export_obj
8       utils.check_dictkey(**check_args)
9   File "/Users/jolsen/gh/pytan/lib/pytan/utils.py", line 2524, in check_dictkey
10      raise HandlerError(err(key, valid_types, k_type))
11 HandlerError: 'header_sort' must be one of [<type 'bool'>, <type 'list'>, <type 'tuple'>], you supplied

```

### Invalid export resultset csv bad expand type

Export a ResultSet from asking a question using a bad expand\_grouped\_columns

### Example Python Code

```

1  import os
2  import sys
3  sys.dont_write_bytecode = True
4
5  # Determine our script name, script dir
6  my_file = os.path.abspath(sys.argv[0])
7  my_dir = os.path.dirname(my_file)
8
9  # determine the pytan lib dir and add it to the path
10 parent_dir = os.path.dirname(my_dir)
11 pytan_root_dir = os.path.dirname(parent_dir)
12 lib_dir = os.path.join(pytan_root_dir, 'lib')
13 path_adds = [lib_dir]
14
15 for aa in path_adds:
16     if aa not in sys.path:
17         sys.path.append(aa)
18
19
20 # connection info for Tanium Server
21 USERNAME = "Tanium User"
22 PASSWORD = "T@n!um"
23 HOST = "172.16.31.128"
24 PORT = "444"
25
26 # Logging conrols
27 LOGLEVEL = 2
28 DEBUGFORMAT = False
29
30 import tempfile
31
32 import pytan
33 handler = pytan.Handler(
34     username=USERNAME,
35     password=PASSWORD,
36     host=HOST,
37     port=PORT,
38     loglevel=LOGLEVEL,
39     debugformat=DEBUGFORMAT,
40 )
41
42 print handler

```

```
43
44 # setup the export_obj kwargs for later
45 export_kwargs = {}
46 export_kwargs["export_format"] = u'csv'
47 export_kwargs["expand_grouped_columns"] = u'bad'
48
49 # ask the question that will provide the resultset that we want to use
50 ask_kwargs = {
51     'qtype': 'manual_human',
52     'sensors': [
53         "Computer Name"
54     ],
55 }
56 response = handler.ask(**ask_kwargs)
57 export_kwargs['obj'] = response['question_results']
58
59 # export the object to a string
60 # this should throw an exception: pytan.utils.HandlerError
61 import traceback
62
63 try:
64     handler.export_obj(**export_kwargs)
65 except Exception as e:
66     traceback.print_exc(file=sys.stdout)
```

### Output from Python Code

```
1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3279
2 2015-03-26 12:02:54,290 INFO      question_progress: Results 0% (Get Computer Name from all machines)
3 2015-03-26 12:02:59,304 INFO      question_progress: Results 0% (Get Computer Name from all machines)
4 2015-03-26 12:03:04,317 INFO      question_progress: Results 100% (Get Computer Name from all machines)
5 Traceback (most recent call last):
6   File "<string>", line 65, in <module>
7   File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 1435, in export_obj
8     utils.check_dictkey(**check_args)
9   File "/Users/jolsen/gh/pytan/lib/pytan/utils.py", line 2524, in check_dictkey
10     raise HandlerError(err(key, valid_types, k_type))
11 HandlerError: 'expand_grouped_columns' must be one of [<type 'bool'>], you supplied <type 'unicode'>
```

### Invalid export resultset csv bad sensors sub type

Export a ResultSet from asking a question using a bad sensors

### Example Python Code

```
1 import os
2 import sys
3 sys.dont_write_bytecode = True
4
5 # Determine our script name, script dir
6 my_file = os.path.abspath(sys.argv[0])
7 my_dir = os.path.dirname(my_file)
8
9 # determine the pytan lib dir and add it to the path
```

```

10 parent_dir = os.path.dirname(my_dir)
11 pytan_root_dir = os.path.dirname(parent_dir)
12 lib_dir = os.path.join(pytan_root_dir, 'lib')
13 path_adds = [lib_dir]
14
15 for aa in path_adds:
16     if aa not in sys.path:
17         sys.path.append(aa)
18
19
20 # connection info for Tanium Server
21 USERNAME = "Tanium User"
22 PASSWORD = "T@n!um"
23 HOST = "172.16.31.128"
24 PORT = "444"
25
26 # Logging conrols
27 LOGLEVEL = 2
28 DEBUGFORMAT = False
29
30 import tempfile
31
32 import pytan
33 handler = pytan.Handler(
34     username=USERNAME,
35     password=PASSWORD,
36     host=HOST,
37     port=PORT,
38     loglevel=LOGLEVEL,
39     debugformat=DEBUGFORMAT,
40 )
41
42 print handler
43
44 # setup the export_obj kwargs for later
45 export_kwargs = {}
46 export_kwargs["export_format"] = u'csv'
47 export_kwargs["sensors"] = [[]]
48 export_kwargs["header_add_sensor"] = True
49
50 # ask the question that will provide the resultset that we want to use
51 ask_kwargs = {
52     'qtype': 'manual_human',
53     'sensors': [
54         "Computer Name"
55     ],
56 }
57 response = handler.ask(**ask_kwargs)
58 export_kwargs['obj'] = response['question_results']
59
60 # export the object to a string
61 # this should throw an exception: pytan.utils.HandlerError
62 import traceback
63
64 try:
65     handler.export_obj(**export_kwargs)
66 except Exception as e:

```

```
67     traceback.print_exc(file=sys.stdout)
```

### Output from Python Code

```
1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3279
2 2015-03-26 12:03:04,396 INFO      question_progress: Results 0% (Get Computer Name from all machines)
3 2015-03-26 12:03:09,409 INFO      question_progress: Results 0% (Get Computer Name from all machines)
4 2015-03-26 12:03:14,423 INFO      question_progress: Results 100% (Get Computer Name from all machines)
5 Traceback (most recent call last):
6   File "<string>", line 66, in <module>
7   File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 1435, in export_obj
8     utils.check_dictkey(**check_args)
9   File "/Users/jolsen/gh/pytan/lib/pytan/utils.py", line 2531, in check_dictkey
10     raise HandlerError(err(key, valid_list_types, list_types))
11 HandlerError: 'sensors' must be a list of [<class 'taniumpy.object_types.sensor.Sensor'>], you supplied
```

### Invalid export resultset bad format

Export a ResultSet from asking a question using a bad export\_format

### Example Python Code

```
1 import os
2 import sys
3 sys.dont_write_bytecode = True
4
5 # Determine our script name, script dir
6 my_file = os.path.abspath(sys.argv[0])
7 my_dir = os.path.dirname(my_file)
8
9 # determine the pytan lib dir and add it to the path
10 parent_dir = os.path.dirname(my_dir)
11 pytan_root_dir = os.path.dirname(parent_dir)
12 lib_dir = os.path.join(pytan_root_dir, 'lib')
13 path_adds = [lib_dir]
14
15 for aa in path_adds:
16     if aa not in sys.path:
17         sys.path.append(aa)
18
19
20 # connection info for Tanium Server
21 USERNAME = "Tanium User"
22 PASSWORD = "T@n!um"
23 HOST = "172.16.31.128"
24 PORT = "444"
25
26 # Logging conrols
27 LOGLEVEL = 2
28 DEBUGFORMAT = False
29
30 import tempfile
31
32 import pytan
```



```

33 handler = pytan.Handler(
34     username=USERNAME,
35     password=PASSWORD,
36     host=HOST,
37     port=PORT,
38     loglevel=LOGLEVEL,
39     debugformat=DEBUGFORMAT,
40 )
41
42 print handler
43
44 # setup the export_obj kwargs for later
45 export_kwargs = {}
46 export_kwargs["export_format"] = u'bad'
47
48 # ask the question that will provide the resultset that we want to use
49 ask_kwargs = {
50     'qtype': 'manual_human',
51     'sensors': [
52         "Computer Name"
53     ],
54 }
55 response = handler.ask(**ask_kwargs)
56 export_kwargs['obj'] = response['question_results']
57
58 # export the object to a string
59 # this should throw an exception: pytan.utils.HandlerError
60 import traceback
61
62 try:
63     handler.export_obj(**export_kwargs)
64 except Exception as e:
65     traceback.print_exc(file=sys.stdout)

```

## Output from Python Code

```

1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3279
2 2015-03-26 12:03:14,502 INFO      question_progress: Results 0% (Get Computer Name from all machines)
3 2015-03-26 12:03:19,517 INFO      question_progress: Results 50% (Get Computer Name from all machines)
4 2015-03-26 12:03:24,533 INFO      question_progress: Results 50% (Get Computer Name from all machines)
5 2015-03-26 12:03:29,545 INFO      question_progress: Results 100% (Get Computer Name from all machines)
6 Traceback (most recent call last):
7   File "<string>", line 64, in <module>
8   File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 1429, in export_obj
9     raise HandlerError(err)
10 HandlerError: u'bad' not a supported export format for ResultSet, must be one of: json, csv

```

## pytan API Valid Export BaseType Examples

### Export basetype csv default options

Export a BaseType from getting objects as CSV with the default options

### Example Python Code

```
1 import os
2 import sys
3 sys.dont_write_bytecode = True
4
5 # Determine our script name, script dir
6 my_file = os.path.abspath(sys.argv[0])
7 my_dir = os.path.dirname(my_file)
8
9 # determine the pytan lib dir and add it to the path
10 parent_dir = os.path.dirname(my_dir)
11 pytan_root_dir = os.path.dirname(parent_dir)
12 lib_dir = os.path.join(pytan_root_dir, 'lib')
13 path_adds = [lib_dir]
14
15 for aa in path_adds:
16     if aa not in sys.path:
17         sys.path.append(aa)
18
19
20 # connection info for Tanium Server
21 USERNAME = "Tanium User"
22 PASSWORD = "T@n!um"
23 HOST = "172.16.31.128"
24 PORT = "444"
25
26 # Logging conrols
27 LOGLEVEL = 2
28 DEBUGFORMAT = False
29
30 import tempfile
31
32 import pytan
33 handler = pytan.Handler(
34     username=USERNAME,
35     password=PASSWORD,
36     host=HOST,
37     port=PORT,
38     loglevel=LOGLEVEL,
39     debugformat=DEBUGFORMAT,
40 )
41
42 print handler
43
44 # setup the export_obj kwargs for later
45 export_kwargs = {}
46 export_kwargs["export_format"] = u'csv'
47
48 # get the objects that will provide the basetype that we want to use
49 get_kwargs = {
50     'name': [
51         "Computer Name", "IP Route Details", "IP Address",
52         "Folder Name Search with RegEx Match",
53     ],
54     'objtype': 'sensor',
55 }
56 response = handler.get(**get_kwargs)
57
58 # export the object to a string
```

```

59 # (we could just as easily export to a file using export_to_report_file)
60 export_kwargs['obj'] = response
61 export_str = handler.export_obj(**export_kwargs)
62
63
64 print ""
65 print "print the export_str returned from export_obj():"
66
67 out = export_str
68 if len(out.splitlines()) > 15:
69     out = out.splitlines()[0:15]
70     out.append('..trimmed for brevity..')
71     out = '\n'.join(out)
72
73 print out

```

### Output from Python Code

```

1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3279
2
3 print the export_str returned from export_obj():
4 category,creation_time,delimiter,description,exclude_from_parse_flag,hash,hidden_flag,id,ignore_case
5 Reserved,,, "The assigned name of the client machine.
6 Example: workstation-1.company.com",0,3409330187,0,3,1,,86400,,,,,Computer Name,,Windows,select CSNa
7 Network,2015-03-03T19:03:36,|,"Returns IPv4 network routes, filtered to exclude noise. With Flags, M
8 Example: 172.16.0.0|192.168.1.1|255.255.0.0|UG|100|eth0",1,435227963,0,737,1,Jim Olsen,60,0,defined
9 Set objWMIService = GetObject(&quot;winmgmts:&quot;; _
10     &amp; &quot;{impersonationLevel=impersonate}!\\&quot;; &amp; strComputer &amp; &quot;;\root\cimv2&
11
12 Set collip = objWMIService.ExecQuery(&quot;select * from win32_networkadapterconfiguration where IPE
13 dim ipaddrs()
14 ipcount = 0
15 for each ipItem in collip
16     for each ipaddr in ipItem.IPAddress
17         ipcount = ipcount + 1
18     next
19 ..trimmed for brevity..

```

### Export basetype json type false

Export a BaseType from getting objects as JSON with false for include\_type

### Example Python Code

```

1 import os
2 import sys
3 sys.dont_write_bytecode = True
4
5 # Determine our script name, script dir
6 my_file = os.path.abspath(sys.argv[0])
7 my_dir = os.path.dirname(my_file)
8
9 # determine the pytan lib dir and add it to the path
10 parent_dir = os.path.dirname(my_dir)

```

```
11 pytan_root_dir = os.path.dirname(parent_dir)
12 lib_dir = os.path.join(pytan_root_dir, 'lib')
13 path_adds = [lib_dir]
14
15 for aa in path_adds:
16     if aa not in sys.path:
17         sys.path.append(aa)
18
19
20 # connection info for Tanium Server
21 USERNAME = "Tanium User"
22 PASSWORD = "T@n!um"
23 HOST = "172.16.31.128"
24 PORT = "444"
25
26 # Logging conrols
27 LOGLEVEL = 2
28 DEBUGFORMAT = False
29
30 import tempfile
31
32 import pytan
33 handler = pytan.Handler(
34     username=USERNAME,
35     password=PASSWORD,
36     host=HOST,
37     port=PORT,
38     loglevel=LOGLEVEL,
39     debugformat=DEBUGFORMAT,
40 )
41
42 print handler
43
44 # setup the export_obj kwargs for later
45 export_kwargs = {}
46 export_kwargs["export_format"] = u'json'
47 export_kwargs["include_type"] = False
48
49 # get the objects that will provide the basetype that we want to use
50 get_kwargs = {
51     'name': [
52         "Computer Name", "IP Route Details", "IP Address",
53         'Folder Name Search with RegEx Match',
54     ],
55     'objtype': 'sensor',
56 }
57 response = handler.get(**get_kwargs)
58
59 # export the object to a string
60 # (we could just as easily export to a file using export_to_report_file)
61 export_kwargs['obj'] = response
62 export_str = handler.export_obj(**export_kwargs)
63
64
65 print ""
66 print "print the export_str returned from export_obj():"
67
68 out = export_str
```

```

69 if len(out.splitlines()) > 15:
70     out = out.splitlines()[0:15]
71     out.append('..trimmed for brevity..')
72     out = '\n'.join(out)
73
74 print out

```

### Output from Python Code

```

1  Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3279
2
3  print the export_str returned from export_obj():
4  {
5      "sensor": [
6          {
7              "category": "Reserved",
8              "description": "The assigned name of the client machine.\nExample: workstation-1.company.com",
9              "exclude_from_parse_flag": 0,
10             "hash": 3409330187,
11             "hidden_flag": 0,
12             "id": 3,
13             "ignore_case_flag": 1,
14             "max_age_seconds": 86400,
15             "name": "Computer Name",
16             "queries": {
17                 "query": [
18                     {
19 ..trimmed for brevity..

```

### Export basetype json explode false

Export a BaseType from getting objects as JSON with false for explode\_json\_string\_values

### Example Python Code

```

1  import os
2  import sys
3  sys.dont_write_bytecode = True
4
5  # Determine our script name, script dir
6  my_file = os.path.abspath(sys.argv[0])
7  my_dir = os.path.dirname(my_file)
8
9  # determine the pytan lib dir and add it to the path
10 parent_dir = os.path.dirname(my_dir)
11 pytan_root_dir = os.path.dirname(parent_dir)
12 lib_dir = os.path.join(pytan_root_dir, 'lib')
13 path_adds = [lib_dir]
14
15 for aa in path_adds:
16     if aa not in sys.path:
17         sys.path.append(aa)
18
19

```

```
20 # connection info for Tanium Server
21 USERNAME = "Tanium User"
22 PASSWORD = "T@n!um"
23 HOST = "172.16.31.128"
24 PORT = "444"
25
26 # Logging conrols
27 LOGLEVEL = 2
28 DEBUGFORMAT = False
29
30 import tempfile
31
32 import pytan
33 handler = pytan.Handler(
34     username=USERNAME,
35     password=PASSWORD,
36     host=HOST,
37     port=PORT,
38     loglevel=LOGLEVEL,
39     debugformat=DEBUGFORMAT,
40 )
41
42 print handler
43
44 # setup the export_obj kwargs for later
45 export_kwargs = {}
46 export_kwargs["export_format"] = u'json'
47 export_kwargs["explode_json_string_values"] = False
48
49 # get the objects that will provide the basetype that we want to use
50 get_kwargs = {
51     'name': [
52         "Computer Name", "IP Route Details", "IP Address",
53         'Folder Name Search with RegEx Match',
54     ],
55     'objtype': 'sensor',
56 }
57 response = handler.get(**get_kwargs)
58
59 # export the object to a string
60 # (we could just as easily export to a file using export_to_report_file)
61 export_kwargs['obj'] = response
62 export_str = handler.export_obj(**export_kwargs)
63
64
65 print ""
66 print "print the export_str returned from export_obj():"
67
68 out = export_str
69 if len(out.splitlines()) > 15:
70     out = out.splitlines()[0:15]
71     out.append('..trimmed for brevity..')
72     out = '\n'.join(out)
73
74 print out
```

## Output from Python Code

```

1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3279
2
3 print the export_str returned from export_obj():
4 {
5     "_type": "sensors",
6     "sensor": [
7         {
8             "_type": "sensor",
9             "category": "Reserved",
10            "description": "The assigned name of the client machine.\nExample: workstation-1.company.com",
11            "exclude_from_parse_flag": 0,
12            "hash": 3409330187,
13            "hidden_flag": 0,
14            "id": 3,
15            "ignore_case_flag": 1,
16            "max_age_seconds": 86400,
17            "name": "Computer Name",
18            "queries": {
19                ..trimmed for brevity..

```

### Export basetype json explode true

Export a BaseType from getting objects as JSON with true for explode\_json\_string\_values

### Example Python Code

```

1 import os
2 import sys
3 sys.dont_write_bytecode = True
4
5 # Determine our script name, script dir
6 my_file = os.path.abspath(sys.argv[0])
7 my_dir = os.path.dirname(my_file)
8
9 # determine the pytan lib dir and add it to the path
10 parent_dir = os.path.dirname(my_dir)
11 pytan_root_dir = os.path.dirname(parent_dir)
12 lib_dir = os.path.join(pytan_root_dir, 'lib')
13 path_adds = [lib_dir]
14
15 for aa in path_adds:
16     if aa not in sys.path:
17         sys.path.append(aa)
18
19
20 # connection info for Tanium Server
21 USERNAME = "Tanium User"
22 PASSWORD = "T@n!um"
23 HOST = "172.16.31.128"
24 PORT = "444"
25
26 # Logging conrols
27 LOGLEVEL = 2
28 DEBUGFORMAT = False
29

```

```
30 import tempfile
31
32 import pytan
33 handler = pytan.Handler(
34     username=USERNAME,
35     password=PASSWORD,
36     host=HOST,
37     port=PORT,
38     loglevel=LOGLEVEL,
39     debugformat=DEBUGFORMAT,
40 )
41
42 print handler
43
44 # setup the export_obj kwargs for later
45 export_kwargs = {}
46 export_kwargs["export_format"] = u'json'
47 export_kwargs["explode_json_string_values"] = True
48
49 # get the objects that will provide the basetype that we want to use
50 get_kwargs = {
51     'name': [
52         "Computer Name", "IP Route Details", "IP Address",
53         'Folder Name Search with RegEx Match',
54     ],
55     'objtype': 'sensor',
56 }
57 response = handler.get(**get_kwargs)
58
59 # export the object to a string
60 # (we could just as easily export to a file using export_to_report_file)
61 export_kwargs['obj'] = response
62 export_str = handler.export_obj(**export_kwargs)
63
64
65 print ""
66 print "print the export_str returned from export_obj():"
67
68 out = export_str
69 if len(out.splitlines()) > 15:
70     out = out.splitlines()[0:15]
71     out.append('..trimmed for brevity..')
72     out = '\n'.join(out)
73
74 print out
```

### Output from Python Code

```
1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3279
2
3 print the export_str returned from export_obj():
4 {
5     "_type": "sensors",
6     "sensor": [
7         {
8             "_type": "sensor",
9             "category": "Reserved",
```



```

10     "description": "The assigned name of the client machine.\nExample: workstation-1|company.com",
11     "exclude_from_parse_flag": 0,
12     "hash": 3409330187,
13     "hidden_flag": 0,
14     "id": 3,
15     "ignore_case_flag": 1,
16     "max_age_seconds": 86400,
17     "name": "Computer Name",
18     "queries": {
19 ..trimmed for brevity..

```

### Export basetype xml default options

Export a BaseType from getting objects as XML with the default options

### Example Python Code

```

1  import os
2  import sys
3  sys.dont_write_bytecode = True
4
5  # Determine our script name, script dir
6  my_file = os.path.abspath(sys.argv[0])
7  my_dir = os.path.dirname(my_file)
8
9  # determine the pytan lib dir and add it to the path
10 parent_dir = os.path.dirname(my_dir)
11 pytan_root_dir = os.path.dirname(parent_dir)
12 lib_dir = os.path.join(pytan_root_dir, 'lib')
13 path_adds = [lib_dir]
14
15 for aa in path_adds:
16     if aa not in sys.path:
17         sys.path.append(aa)
18
19
20 # connection info for Tanium Server
21 USERNAME = "Tanium User"
22 PASSWORD = "T@n!um"
23 HOST = "172.16.31.128"
24 PORT = "444"
25
26 # Logging conrols
27 LOGLEVEL = 2
28 DEBUGFORMAT = False
29
30 import tempfile
31
32 import pytan
33 handler = pytan.Handler(
34     username=USERNAME,
35     password=PASSWORD,
36     host=HOST,
37     port=PORT,
38     loglevel=LOGLEVEL,

```

```
39     debugformat=DEBUGFORMAT,
40 )
41
42 print handler
43
44 # setup the export_obj kwargs for later
45 export_kwargs = {}
46 export_kwargs["export_format"] = u'xml'
47
48 # get the objects that will provide the basetype that we want to use
49 get_kwargs = {
50     'name': [
51         "Computer Name", "IP Route Details", "IP Address",
52         'Folder Name Search with RegEx Match',
53     ],
54     'objtype': 'sensor',
55 }
56 response = handler.get(**get_kwargs)
57
58 # export the object to a string
59 # (we could just as easily export to a file using export_to_report_file)
60 export_kwargs['obj'] = response
61 export_str = handler.export_obj(**export_kwargs)
62
63
64 print ""
65 print "print the export_str returned from export_obj():"
66
67 out = export_str
68 if len(out.splitlines()) > 15:
69     out = out.splitlines()[0:15]
70     out.append('..trimmed for brevity..')
71     out = '\n'.join(out)
72
73 print out
```

### Output from Python Code

```
1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3279
2
3 print the export_str returned from export_obj():
4 <sensors><cache_info /><sensor><category>Reserved</category><preview_sensor_flag /><hash>3409330187<
5 Example: workstation-1.company.com</description><string_hints /><subcolumns /><metadata /><parameter
6 Example: 172.16.0.0|192.168.1.1|255.255.0.0|UG|100|eth0</description><string_hints /><subcolumns><s
7 Set objWMIService = GetObject(&quot;winmgmts:&quot; _
8     &amp;&amp; &quot;{impersonationLevel=impersonate}!\&quot; &amp;&amp; strComputer &amp;&amp;
9
10 Set collip = objWMIService.ExecQuery(&quot;select * from win32_networkadapterconfiguration where
11 dim ipaddrs()
12 ipcount = 0
13 for each ipItem in collip
14     for each ipaddr in ipItem.IPAddress
15         ipcount = ipcount + 1
16     next
17 next
18 redim ipaddrs(ipcount)
```

```
19 | ..trimmed for brevity..
```

### Export basetype xml minimal false

Export a BaseType from getting objects as XML with false for minimal

### Example Python Code

```
1  import os
2  import sys
3  sys.dont_write_bytecode = True
4
5  # Determine our script name, script dir
6  my_file = os.path.abspath(sys.argv[0])
7  my_dir = os.path.dirname(my_file)
8
9  # determine the pytan lib dir and add it to the path
10 parent_dir = os.path.dirname(my_dir)
11 pytan_root_dir = os.path.dirname(parent_dir)
12 lib_dir = os.path.join(pytan_root_dir, 'lib')
13 path_adds = [lib_dir]
14
15 for aa in path_adds:
16     if aa not in sys.path:
17         sys.path.append(aa)
18
19
20 # connection info for Tanium Server
21 USERNAME = "Tanium User"
22 PASSWORD = "T@n!um"
23 HOST = "172.16.31.128"
24 PORT = "444"
25
26 # Logging conrols
27 LOGLEVEL = 2
28 DEBUGFORMAT = False
29
30 import tempfile
31
32 import pytan
33 handler = pytan.Handler(
34     username=USERNAME,
35     password=PASSWORD,
36     host=HOST,
37     port=PORT,
38     loglevel=LOGLEVEL,
39     debugformat=DEBUGFORMAT,
40 )
41
42 print handler
43
44 # setup the export_obj kwargs for later
45 export_kwargs = {}
46 export_kwargs["export_format"] = u'xml'
47 export_kwargs["minimal"] = False
```

```
48
49 # get the objects that will provide the basetype that we want to use
50 get_kwargs = {
51     'name': [
52         "Computer Name", "IP Route Details", "IP Address",
53         'Folder Name Search with RegEx Match',
54     ],
55     'objtype': 'sensor',
56 }
57 response = handler.get(**get_kwargs)
58
59 # export the object to a string
60 # (we could just as easily export to a file using export_to_report_file)
61 export_kwargs['obj'] = response
62 export_str = handler.export_obj(**export_kwargs)
63
64
65 print ""
66 print "print the export_str returned from export_obj():"
67
68 out = export_str
69 if len(out.splitlines()) > 15:
70     out = out.splitlines()[0:15]
71     out.append('..trimmed for brevity..')
72     out = '\n'.join(out)
73
74 print out
```

### Output from Python Code

```
1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3279
2
3 print the export_str returned from export_obj():
4 <sensors><cache_info /><sensor><category>Reserved</category><preview_sensor_flag /><hash>3409330187<
5 Example: workstation-1.company.com</description><string_hints /><subcolumns /><metadata /><parameter
6 Example: 172.16.0.0|192.168.1.1|255.255.0.0|UG|100|eth0</description><string_hints /><subcolumns><s
7 Set objWMIService = GetObject(&quot;winmgmts:&quot; _
8     &amp; & &quot;{impersonationLevel=impersonate}!\\&quot; & & strComputer & &
9
10 Set collip = objWMIService.ExecQuery(&quot;select * from win32_networkadapterconfiguration where
11 dim ipaddrs()
12 ipcount = 0
13 for each ipItem in collip
14     for each ipaddr in ipItem.IPAddress
15         ipcount = ipcount + 1
16     next
17 next
18 redim ipaddrs(ipcount)
19 ..trimmed for brevity..
```

### Export basetype xml minimal true

Export a BaseType from getting objects as XML with true for minimal

## Example Python Code

```

1  import os
2  import sys
3  sys.dont_write_bytecode = True
4
5  # Determine our script name, script dir
6  my_file = os.path.abspath(sys.argv[0])
7  my_dir = os.path.dirname(my_file)
8
9  # determine the pytan lib dir and add it to the path
10 parent_dir = os.path.dirname(my_dir)
11 pytan_root_dir = os.path.dirname(parent_dir)
12 lib_dir = os.path.join(pytan_root_dir, 'lib')
13 path_adds = [lib_dir]
14
15 for aa in path_adds:
16     if aa not in sys.path:
17         sys.path.append(aa)
18
19
20 # connection info for Tanium Server
21 USERNAME = "Tanium User"
22 PASSWORD = "T@n!um"
23 HOST = "172.16.31.128"
24 PORT = "444"
25
26 # Logging conrols
27 LOGLEVEL = 2
28 DEBUGFORMAT = False
29
30 import tempfile
31
32 import pytan
33 handler = pytan.Handler(
34     username=USERNAME,
35     password=PASSWORD,
36     host=HOST,
37     port=PORT,
38     loglevel=LOGLEVEL,
39     debugformat=DEBUGFORMAT,
40 )
41
42 print handler
43
44 # setup the export_obj kwargs for later
45 export_kwargs = {}
46 export_kwargs["export_format"] = u'xml'
47 export_kwargs["minimal"] = True
48
49 # get the objects that will provide the basetype that we want to use
50 get_kwargs = {
51     'name': [
52         "Computer Name", "IP Route Details", "IP Address",
53         'Folder Name Search with RegEx Match',
54     ],
55     'objtype': 'sensor',
56 }
57 response = handler.get(**get_kwargs)

```

```
58
59 # export the object to a string
60 # (we could just as easily export to a file using export_to_report_file)
61 export_kwargs['obj'] = response
62 export_str = handler.export_obj(**export_kwargs)
63
64
65 print ""
66 print "print the export_str returned from export_obj():"
67
68 out = export_str
69 if len(out.splitlines()) > 15:
70     out = out.splitlines()[0:15]
71     out.append('..trimmed for brevity..')
72     out = '\n'.join(out)
73
74 print out
```

### Output from Python Code

```
1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3279
2
3 print the export_str returned from export_obj():
4 <sensors><sensor><category>Reserved</category><hash>3409330187</hash><name>Computer Name</name><hid
5 Example: workstation-1.company.com</description><queries><query><platform>Windows</platform><script_
6 Example: 172.16.0.0|192.168.1.1|255.255.0.0|UG|100|eth0</description><subcolumns><subcolumn><index>
7 Set objWMIService = GetObject(&quot;winmgmts:&quot;; _
8     &amp;amp; &quot;{impersonationLevel=impersonate}!\&quot;; &amp;amp; strComputer &amp;amp;
9
10 Set collip = objWMIService.ExecQuery(&quot;select * from win32_networkadapterconfiguration where
11 dim ipaddrs()
12 ipcount = 0
13 for each ipItem in collip
14     for each ipaddr in ipItem.IPAddress
15         ipcount = ipcount + 1
16     next
17 next
18 redim ipaddrs(ipcount)
19 ..trimmed for brevity..
```

### Export basetype csv with explode false

Export a BaseType from getting objects as CSV with false for explode\_json\_string\_values

### Example Python Code

```
1 import os
2 import sys
3 sys.dont_write_bytecode = True
4
5 # Determine our script name, script dir
6 my_file = os.path.abspath(sys.argv[0])
7 my_dir = os.path.dirname(my_file)
8
```

```

9  # determine the pytan lib dir and add it to the path
10 parent_dir = os.path.dirname(my_dir)
11 pytan_root_dir = os.path.dirname(parent_dir)
12 lib_dir = os.path.join(pytan_root_dir, 'lib')
13 path_adds = [lib_dir]
14
15 for aa in path_adds:
16     if aa not in sys.path:
17         sys.path.append(aa)
18
19
20 # connection info for Tanium Server
21 USERNAME = "Tanium User"
22 PASSWORD = "T@n!um"
23 HOST = "172.16.31.128"
24 PORT = "444"
25
26 # Logging conrols
27 LOGLEVEL = 2
28 DEBUGFORMAT = False
29
30 import tempfile
31
32 import pytan
33 handler = pytan.Handler(
34     username=USERNAME,
35     password=PASSWORD,
36     host=HOST,
37     port=PORT,
38     loglevel=LOGLEVEL,
39     debugformat=DEBUGFORMAT,
40 )
41
42 print handler
43
44 # setup the export_obj kwargs for later
45 export_kwargs = {}
46 export_kwargs["export_format"] = u'csv'
47 export_kwargs["explode_json_string_values"] = False
48
49 # get the objects that will provide the basetype that we want to use
50 get_kwargs = {
51     'name': [
52         "Computer Name", "IP Route Details", "IP Address",
53         'Folder Name Search with RegEx Match',
54     ],
55     'objtype': 'sensor',
56 }
57 response = handler.get(**get_kwargs)
58
59 # export the object to a string
60 # (we could just as easily export to a file using export_to_report_file)
61 export_kwargs['obj'] = response
62 export_str = handler.export_obj(**export_kwargs)
63
64
65 print ""
66 print "print the export_str returned from export_obj():"

```

```
67
68 out = export_str
69 if len(out.splitlines()) > 15:
70     out = out.splitlines()[0:15]
71     out.append('..trimmed for brevity..')
72     out = '\n'.join(out)
73
74 print out
```

### Output from Python Code

```
1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3279
2
3 print the export_str returned from export_obj():
4 category,creation_time,delimiter,description,exclude_from_parse_flag,hash,hidden_flag,id,ignore_case
5 Reserved,,, "The assigned name of the client machine.
6 Example: workstation-1.company.com",0,3409330187,0,3,1,,86400,,,,,Computer Name,,Windows,select CSNa
7 Network,2015-03-03T19:03:36,|,"Returns IPv4 network routes, filtered to exclude noise. With Flags, M
8 Example: 172.16.0.0|192.168.1.1|255.255.0.0|UG|100|eth0",1,435227963,0,737,1,Jim Olsen,60,0,defined
9 Set objWMIService = GetObject(&quot;winmgmts:&quot;; _
10     &amp; &quot;{impersonationLevel=impersonate}!\\&quot;; &amp; strComputer &amp; &quot;;\root\cimv2&
11
12 Set collip = objWMIService.ExecQuery(&quot;select * from win32_networkadapterconfiguration where IPE
13 dim ipaddrs()
14 ipcount = 0
15 for each ipItem in collip
16     for each ipaddr in ipItem.IPAddress
17         ipcount = ipcount + 1
18     next
19 ..trimmed for brevity..
```

### Export basetype csv with explode true

Export a BaseType from getting objects as CSV with true for explode\_json\_string\_values

### Example Python Code

```
1 import os
2 import sys
3 sys.dont_write_bytecode = True
4
5 # Determine our script name, script dir
6 my_file = os.path.abspath(sys.argv[0])
7 my_dir = os.path.dirname(my_file)
8
9 # determine the pytan lib dir and add it to the path
10 parent_dir = os.path.dirname(my_dir)
11 pytan_root_dir = os.path.dirname(parent_dir)
12 lib_dir = os.path.join(pytan_root_dir, 'lib')
13 path_adds = [lib_dir]
14
15 for aa in path_adds:
16     if aa not in sys.path:
17         sys.path.append(aa)
```



```

18
19
20 # connection info for Tanium Server
21 USERNAME = "Tanium User"
22 PASSWORD = "T@n!um"
23 HOST = "172.16.31.128"
24 PORT = "444"
25
26 # Logging conrols
27 LOGLEVEL = 2
28 DEBUGFORMAT = False
29
30 import tempfile
31
32 import pytan
33 handler = pytan.Handler(
34     username=USERNAME,
35     password=PASSWORD,
36     host=HOST,
37     port=PORT,
38     loglevel=LOGLEVEL,
39     debugformat=DEBUGFORMAT,
40 )
41
42 print handler
43
44 # setup the export_obj kwargs for later
45 export_kwargs = {}
46 export_kwargs["export_format"] = u'csv'
47 export_kwargs["explode_json_string_values"] = True
48
49 # get the objects that will provide the basetype that we want to use
50 get_kwargs = {
51     'name': [
52         "Computer Name", "IP Route Details", "IP Address",
53         "Folder Name Search with RegEx Match",
54     ],
55     'objtype': 'sensor',
56 }
57 response = handler.get(**get_kwargs)
58
59 # export the object to a string
60 # (we could just as easily export to a file using export_to_report_file)
61 export_kwargs['obj'] = response
62 export_str = handler.export_obj(**export_kwargs)
63
64
65 print ""
66 print "print the export_str returned from export_obj():"
67
68 out = export_str
69 if len(out.splitlines()) > 15:
70     out = out.splitlines()[0:15]
71     out.append('..trimmed for brevity..')
72     out = '\n'.join(out)
73
74 print out

```

## Output from Python Code

```

1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3279
2
3 print the export_str returned from export_obj():
4 category,creation_time,delimiter,description,exclude_from_parse_flag,hash,hidden_flag,id,ignore_case
5 Reserved,,, "The assigned name of the client machine.
6 Example: workstation-1.company.com",0,3409330187,0,3,1,,86400,,,,,Computer Name,,,,,,,,,,,,,
7 Network,2015-03-03T19:03:36,|,"Returns IPv4 network routes, filtered to exclude noise. With Flags, M
8 Example: 172.16.0.0|192.168.1.1|255.255.0.0|UG|100|eth0",1,435227963,0,737,1,Jim Olsen,60,0,defined
9 Set objWMIService = GetObject(&quot;winmgmts:&quot; _
10     &amp; &quot;{impersonationLevel=impersonate}!\&quot; &amp; strComputer &amp; &quot;\root\cimv2&
11
12 Set collip = objWMIService.ExecQuery(&quot;select * from win32_networkadapterconfiguration where IPE
13 dim ipaddrs()
14 ipcount = 0
15 for each ipItem in collip
16     for each ipaddr in ipItem.IPAddress
17         ipcount = ipcount + 1
18     next
19 ..trimmed for brevity..

```

## Export basetype csv with sort empty list

Export a BaseType from getting objects as CSV with an empty list for header\_sort

## Example Python Code

```

1 import os
2 import sys
3 sys.dont_write_bytecode = True
4
5 # Determine our script name, script dir
6 my_file = os.path.abspath(sys.argv[0])
7 my_dir = os.path.dirname(my_file)
8
9 # determine the pytan lib dir and add it to the path
10 parent_dir = os.path.dirname(my_dir)
11 pytan_root_dir = os.path.dirname(parent_dir)
12 lib_dir = os.path.join(pytan_root_dir, 'lib')
13 path_adds = [lib_dir]
14
15 for aa in path_adds:
16     if aa not in sys.path:
17         sys.path.append(aa)
18
19
20 # connection info for Tanium Server
21 USERNAME = "Tanium User"
22 PASSWORD = "T@n!um"
23 HOST = "172.16.31.128"
24 PORT = "444"
25
26 # Logging conrols
27 LOGLEVEL = 2
28 DEBUGFORMAT = False

```

```

29
30 import tempfile
31
32 import pytan
33 handler = pytan.Handler(
34     username=USERNAME,
35     password=PASSWORD,
36     host=HOST,
37     port=PORT,
38     loglevel=LOGLEVEL,
39     debugformat=DEBUGFORMAT,
40 )
41
42 print handler
43
44 # setup the export_obj kwargs for later
45 export_kwargs = {}
46 export_kwargs["export_format"] = u'csv'
47 export_kwargs["header_sort"] = []
48
49 # get the objects that will provide the basetype that we want to use
50 get_kwargs = {
51     'name': [
52         "Computer Name", "IP Route Details", "IP Address",
53         'Folder Name Search with RegEx Match',
54     ],
55     'objtype': 'sensor',
56 }
57 response = handler.get(**get_kwargs)
58
59 # export the object to a string
60 # (we could just as easily export to a file using export_to_report_file)
61 export_kwargs['obj'] = response
62 export_str = handler.export_obj(**export_kwargs)
63
64
65 print ""
66 print "print the export_str returned from export_obj():"
67
68 out = export_str
69 if len(out.splitlines()) > 15:
70     out = out.splitlines()[0:15]
71     out.append('..trimmed for brevity..')
72     out = '\n'.join(out)
73
74 print out

```

### Output from Python Code

```

1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3279
2
3 print the export_str returned from export_obj():
4 category,creation_time,delimiter,description,exclude_from_parse_flag,hash,hidden_flag,id,ignore_case
5 Reserved,,, "The assigned name of the client machine.
6 Example: workstation-1.company.com",0,3409330187,0,3,1,,86400,,,,,Computer Name,,Windows,select CSNa
7 Network,2015-03-03T19:03:36,|,"Returns IPv4 network routes, filtered to exclude noise. With Flags, M
8 Example: 172.16.0.0|192.168.1.1|255.255.0.0|UG|100|eth0",1,435227963,0,737,1,Jim Olsen,60,0,defined

```

```
9 Set objWMIService = GetObject(&quot;winmgmts:&quot;; _
10     &amp; &quot;{impersonationLevel=impersonate}!\\&quot;; &amp; strComputer &amp; &quot;;\root\cimv2&
11
12 Set collip = objWMIService.ExecQuery(&quot;select * from win32_networkadapterconfiguration where IP
13 dim ipaddrs()
14 ipcount = 0
15 for each ipItem in collip
16     for each ipaddr in ipItem.IPAddress
17         ipcount = ipcount + 1
18     next
19 ..trimmed for brevity..
```

### Export basetype csv with sort true

Export a BaseType from getting objects as CSV with true for header\_sort

### Example Python Code

```
1 import os
2 import sys
3 sys.dont_write_bytecode = True
4
5 # Determine our script name, script dir
6 my_file = os.path.abspath(sys.argv[0])
7 my_dir = os.path.dirname(my_file)
8
9 # determine the pytan lib dir and add it to the path
10 parent_dir = os.path.dirname(my_dir)
11 pytan_root_dir = os.path.dirname(parent_dir)
12 lib_dir = os.path.join(pytan_root_dir, 'lib')
13 path_adds = [lib_dir]
14
15 for aa in path_adds:
16     if aa not in sys.path:
17         sys.path.append(aa)
18
19
20 # connection info for Tanium Server
21 USERNAME = "Tanium User"
22 PASSWORD = "T@n!um"
23 HOST = "172.16.31.128"
24 PORT = "444"
25
26 # Logging conrols
27 LOGLEVEL = 2
28 DEBUGFORMAT = False
29
30 import tempfile
31
32 import pytan
33 handler = pytan.Handler(
34     username=USERNAME,
35     password=PASSWORD,
36     host=HOST,
37     port=PORT,
```

```

38     loglevel=LOGLEVEL,
39     debugformat=DEBUGFORMAT,
40 )
41
42 print handler
43
44 # setup the export_obj kwargs for later
45 export_kwargs = {}
46 export_kwargs["export_format"] = u'csv'
47 export_kwargs["header_sort"] = True
48
49 # get the objects that will provide the basetype that we want to use
50 get_kwargs = {
51     'name': [
52         "Computer Name", "IP Route Details", "IP Address",
53         'Folder Name Search with RegEx Match',
54     ],
55     'objtype': 'sensor',
56 }
57 response = handler.get(**get_kwargs)
58
59 # export the object to a string
60 # (we could just as easily export to a file using export_to_report_file)
61 export_kwargs['obj'] = response
62 export_str = handler.export_obj(**export_kwargs)
63
64
65 print ""
66 print "print the export_str returned from export_obj():"
67
68 out = export_str
69 if len(out.splitlines()) > 15:
70     out = out.splitlines()[0:15]
71     out.append('..trimmed for brevity..')
72     out = '\n'.join(out)
73
74 print out

```

## Output from Python Code

```

1  Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3279
2
3  print the export_str returned from export_obj():
4  category,creation_time,delimiter,description,exclude_from_parse_flag,hash,hidden_flag,id,ignore_case
5  Reserved,,, "The assigned name of the client machine.
6  Example: workstation-1.company.com",0,3409330187,0,3,1,,86400,,,,Computer Name,,Windows,select CSNa
7  Network,2015-03-03T19:03:36,|,"Returns IPv4 network routes, filtered to exclude noise. With Flags, M
8  Example: 172.16.0.0|192.168.1.1|255.255.0.0|UG|100|eth0",1,435227963,0,737,1,Jim Olsen,60,0,defined
9  Set objWMIService = GetObject(&quot;winmgmts:&quot;; _
10     &amp; &quot;{impersonationLevel=impersonate}!\\&quot;; &amp; strComputer &amp; &quot;;\root\cimv2&
11
12 Set collip = objWMIService.ExecQuery(&quot;select * from win32_networkadapterconfiguration where IP
13 dim ipaddrs()
14 ipcount = 0
15 for each ipItem in collip
16     for each ipaddr in ipItem.IPAddress
17         ipcount = ipcount + 1

```

```
18     next
19     ..trimmed for brevity..
```

### Export basetype csv with sort list

Export a BaseType from getting objects as CSV with name and description for header\_sort

### Example Python Code

```
1  import os
2  import sys
3  sys.dont_write_bytecode = True
4
5  # Determine our script name, script dir
6  my_file = os.path.abspath(sys.argv[0])
7  my_dir = os.path.dirname(my_file)
8
9  # determine the pytan lib dir and add it to the path
10 parent_dir = os.path.dirname(my_dir)
11 pytan_root_dir = os.path.dirname(parent_dir)
12 lib_dir = os.path.join(pytan_root_dir, 'lib')
13 path_adds = [lib_dir]
14
15 for aa in path_adds:
16     if aa not in sys.path:
17         sys.path.append(aa)
18
19
20 # connection info for Tanium Server
21 USERNAME = "Tanium User"
22 PASSWORD = "T@n!um"
23 HOST = "172.16.31.128"
24 PORT = "444"
25
26 # Logging conrols
27 LOGLEVEL = 2
28 DEBUGFORMAT = False
29
30 import tempfile
31
32 import pytan
33 handler = pytan.Handler(
34     username=USERNAME,
35     password=PASSWORD,
36     host=HOST,
37     port=PORT,
38     loglevel=LOGLEVEL,
39     debugformat=DEBUGFORMAT,
40 )
41
42 print handler
43
44 # setup the export_obj kwargs for later
45 export_kwargs = {}
46 export_kwargs["export_format"] = u'csv'
```

```

47 export_kwargs["header_sort"] = [u'name', u'description']
48
49 # get the objects that will provide the basetype that we want to use
50 get_kwargs = {
51     'name': [
52         "Computer Name", "IP Route Details", "IP Address",
53         'Folder Name Search with RegEx Match',
54     ],
55     'objtype': 'sensor',
56 }
57 response = handler.get(**get_kwargs)
58
59 # export the object to a string
60 # (we could just as easily export to a file using export_to_report_file)
61 export_kwargs['obj'] = response
62 export_str = handler.export_obj(**export_kwargs)
63
64
65 print ""
66 print "print the export_str returned from export_obj():"
67
68 out = export_str
69 if len(out.splitlines()) > 15:
70     out = out.splitlines()[0:15]
71     out.append('..trimmed for brevity..')
72     out = '\n'.join(out)
73
74 print out

```

### Output from Python Code

```

1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3279
2
3 print the export_str returned from export_obj():
4 name,description,category,creation_time,delimiter,exclude_from_parse_flag,hash,hidden_flag,id,ignore
5 Computer Name,"The assigned name of the client machine.
6 Example: workstation-1.company.com",Reserved,,,0,3409330187,0,3,1,,86400,,,,,Windows,select CSName
7 IP Route Details,"Returns IPv4 network routes, filtered to exclude noise. With Flags, Metric, Interf
8 Example: 172.16.0.0|192.168.1.1|255.255.0.0|UG|100|eth0",Network,2015-03-03T19:03:36,,1,435227963,
9 Set objWMIService = GetObject(&quot;winmgmts:&quot;; _
10     &amp; &quot;{impersonationLevel=impersonate}!\\&quot;; &amp; strComputer &amp; &quot;;\root\cimv2&
11
12 Set collip = objWMIService.ExecQuery(&quot;select * from win32_networkadapterconfiguration where IPE
13 dim ipaddrs()
14 ipcount = 0
15 for each ipItem in collip
16     for each ipaddr in ipItem.IPAddress
17         ipcount = ipcount + 1
18     next
19 ..trimmed for brevity..

```

### Export basetype json default options

Export a BaseType from getting objects as JSON with the default options

## Example Python Code

```
1  import os
2  import sys
3  sys.dont_write_bytecode = True
4
5  # Determine our script name, script dir
6  my_file = os.path.abspath(sys.argv[0])
7  my_dir = os.path.dirname(my_file)
8
9  # determine the pytan lib dir and add it to the path
10 parent_dir = os.path.dirname(my_dir)
11 pytan_root_dir = os.path.dirname(parent_dir)
12 lib_dir = os.path.join(pytan_root_dir, 'lib')
13 path_adds = [lib_dir]
14
15 for aa in path_adds:
16     if aa not in sys.path:
17         sys.path.append(aa)
18
19
20 # connection info for Tanium Server
21 USERNAME = "Tanium User"
22 PASSWORD = "T@n!um"
23 HOST = "172.16.31.128"
24 PORT = "444"
25
26 # Logging conrols
27 LOGLEVEL = 2
28 DEBUGFORMAT = False
29
30 import tempfile
31
32 import pytan
33 handler = pytan.Handler(
34     username=USERNAME,
35     password=PASSWORD,
36     host=HOST,
37     port=PORT,
38     loglevel=LOGLEVEL,
39     debugformat=DEBUGFORMAT,
40 )
41
42 print handler
43
44 # setup the export_obj kwargs for later
45 export_kwargs = {}
46 export_kwargs["export_format"] = u'json'
47
48 # get the objects that will provide the basetype that we want to use
49 get_kwargs = {
50     'name': [
51         "Computer Name", "IP Route Details", "IP Address",
52         'Folder Name Search with RegEx Match',
53     ],
54     'objtype': 'sensor',
55 }
56 response = handler.get(**get_kwargs)
57
```



```

58 # export the object to a string
59 # (we could just as easily export to a file using export_to_report_file)
60 export_kwargs['obj'] = response
61 export_str = handler.export_obj(**export_kwargs)
62
63
64 print ""
65 print "print the export_str returned from export_obj():"
66
67 out = export_str
68 if len(out.splitlines()) > 15:
69     out = out.splitlines()[0:15]
70     out.append('..trimmed for brevity..')
71     out = '\n'.join(out)
72
73 print out

```

### Output from Python Code

```

1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3279
2
3 print the export_str returned from export_obj():
4 {
5     "_type": "sensors",
6     "sensor": [
7         {
8             "_type": "sensor",
9             "category": "Reserved",
10            "description": "The assigned name of the client machine.\nExample: workstation-1.company.com",
11            "exclude_from_parse_flag": 0,
12            "hash": 3409330187,
13            "hidden_flag": 0,
14            "id": 3,
15            "ignore_case_flag": 1,
16            "max_age_seconds": 86400,
17            "name": "Computer Name",
18            "queries": {
19                ..trimmed for brevity..

```

### Export basetype json type true

Export a BaseType from getting objects as JSON with true for include\_type

### Example Python Code

```

1 import os
2 import sys
3 sys.dont_write_bytecode = True
4
5 # Determine our script name, script dir
6 my_file = os.path.abspath(sys.argv[0])
7 my_dir = os.path.dirname(my_file)
8
9 # determine the pytan lib dir and add it to the path

```

```
10 parent_dir = os.path.dirname(my_dir)
11 pytan_root_dir = os.path.dirname(parent_dir)
12 lib_dir = os.path.join(pytan_root_dir, 'lib')
13 path_adds = [lib_dir]
14
15 for aa in path_adds:
16     if aa not in sys.path:
17         sys.path.append(aa)
18
19
20 # connection info for Tanium Server
21 USERNAME = "Tanium User"
22 PASSWORD = "T@n!um"
23 HOST = "172.16.31.128"
24 PORT = "444"
25
26 # Logging conrols
27 LOGLEVEL = 2
28 DEBUGFORMAT = False
29
30 import tempfile
31
32 import pytan
33 handler = pytan.Handler(
34     username=USERNAME,
35     password=PASSWORD,
36     host=HOST,
37     port=PORT,
38     loglevel=LOGLEVEL,
39     debugformat=DEBUGFORMAT,
40 )
41
42 print handler
43
44 # setup the export_obj kwargs for later
45 export_kwargs = {}
46 export_kwargs["export_format"] = u'json'
47 export_kwargs["include_type"] = True
48
49 # get the objects that will provide the basetype that we want to use
50 get_kwargs = {
51     'name': [
52         "Computer Name", "IP Route Details", "IP Address",
53         'Folder Name Search with RegEx Match',
54     ],
55     'objtype': 'sensor',
56 }
57 response = handler.get(**get_kwargs)
58
59 # export the object to a string
60 # (we could just as easily export to a file using export_to_report_file)
61 export_kwargs['obj'] = response
62 export_str = handler.export_obj(**export_kwargs)
63
64
65 print ""
66 print "print the export_str returned from export_obj():"
67
```

```

68 out = export_str
69 if len(out.splitlines()) > 15:
70     out = out.splitlines()[0:15]
71     out.append('..trimmed for brevity..')
72     out = '\n'.join(out)
73
74 print out

```

### Output from Python Code

```

1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3279
2
3 print the export_str returned from export_obj():
4 {
5     "_type": "sensors",
6     "sensor": [
7         {
8             "_type": "sensor",
9             "category": "Reserved",
10            "description": "The assigned name of the client machine.\nExample: workstation-1.company.com",
11            "exclude_from_parse_flag": 0,
12            "hash": 3409330187,
13            "hidden_flag": 0,
14            "id": 3,
15            "ignore_case_flag": 1,
16            "max_age_seconds": 86400,
17            "name": "Computer Name",
18            "queries": {
19                ..trimmed for brevity..

```

## pytan API Invalid Export BaseType Examples

### Invalid export basetype csv bad explode type

Export a BaseType from getting objects using a bad explode\_json\_string\_values

### Example Python Code

```

1 import os
2 import sys
3 sys.dont_write_bytecode = True
4
5 # Determine our script name, script dir
6 my_file = os.path.abspath(sys.argv[0])
7 my_dir = os.path.dirname(my_file)
8
9 # determine the pytan lib dir and add it to the path
10 parent_dir = os.path.dirname(my_dir)
11 pytan_root_dir = os.path.dirname(parent_dir)
12 lib_dir = os.path.join(pytan_root_dir, 'lib')
13 path_adds = [lib_dir]
14
15 for aa in path_adds:
16     if aa not in sys.path:

```

```
17         sys.path.append(aa)
18
19
20     # connection info for Tanium Server
21     USERNAME = "Tanium User"
22     PASSWORD = "T@n!um"
23     HOST = "172.16.31.128"
24     PORT = "444"
25
26     # Logging conrols
27     LOGLEVEL = 2
28     DEBUGFORMAT = False
29
30     import tempfile
31
32     import pytan
33     handler = pytan.Handler(
34         username=USERNAME,
35         password=PASSWORD,
36         host=HOST,
37         port=PORT,
38         loglevel=LOGLEVEL,
39         debugformat=DEBUGFORMAT,
40     )
41
42     print handler
43
44     # setup the export_obj kwargs for later
45     export_kwargs = {}
46     export_kwargs["export_format"] = u'csv'
47     export_kwargs["explode_json_string_values"] = u'bad'
48
49     # get the objects that will provide the basetype that we want to use
50     get_kwargs = {
51         'name': [
52             "Computer Name", "IP Route Details", "IP Address",
53             'Folder Name Search with RegEx Match',
54         ],
55         'objtype': 'sensor',
56     }
57     response = handler.get(**get_kwargs)
58     export_kwargs['obj'] = response
59
60     # export the object to a string
61     # this should throw an exception: pytan.utils.HandlerError
62     import traceback
63
64     try:
65         handler.export_obj(**export_kwargs)
66     except Exception as e:
67         traceback.print_exc(file=sys.stdout)
```

### Output from Python Code

```
1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3279
2 Traceback (most recent call last):
3   File "<string>", line 66, in <module>
```

```

4 File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 1435, in export_obj
5     utils.check_dictkey(**check_args)
6 File "/Users/jolsen/gh/pytan/lib/pytan/utils.py", line 2524, in check_dictkey
7     raise HandlerError(err(key, valid_types, k_type))
8 HandlerError: 'explode_json_string_values' must be one of [<type 'bool'>], you supplied <type 'unicod

```

### Invalid export basetype csv bad sort sub type

Export a BaseType from getting objects using a bad header\_sort

### Example Python Code

```

1  import os
2  import sys
3  sys.dont_write_bytecode = True
4
5  # Determine our script name, script dir
6  my_file = os.path.abspath(sys.argv[0])
7  my_dir = os.path.dirname(my_file)
8
9  # determine the pytan lib dir and add it to the path
10 parent_dir = os.path.dirname(my_dir)
11 pytan_root_dir = os.path.dirname(parent_dir)
12 lib_dir = os.path.join(pytan_root_dir, 'lib')
13 path_adds = [lib_dir]
14
15 for aa in path_adds:
16     if aa not in sys.path:
17         sys.path.append(aa)
18
19
20 # connection info for Tanium Server
21 USERNAME = "Tanium User"
22 PASSWORD = "T@n!um"
23 HOST = "172.16.31.128"
24 PORT = "444"
25
26 # Logging conrols
27 LOGLEVEL = 2
28 DEBUGFORMAT = False
29
30 import tempfile
31
32 import pytan
33 handler = pytan.Handler(
34     username=USERNAME,
35     password=PASSWORD,
36     host=HOST,
37     port=PORT,
38     loglevel=LOGLEVEL,
39     debugformat=DEBUGFORMAT,
40 )
41
42 print handler
43

```

```
44 # setup the export_obj kwargs for later
45 export_kwargs = {}
46 export_kwargs["export_format"] = u'csv'
47 export_kwargs["header_sort"] = [[]]
48
49 # get the objects that will provide the basetype that we want to use
50 get_kwargs = {
51     'name': [
52         "Computer Name", "IP Route Details", "IP Address",
53         'Folder Name Search with RegEx Match',
54     ],
55     'objtype': 'sensor',
56 }
57 response = handler.get(**get_kwargs)
58 export_kwargs['obj'] = response
59
60 # export the object to a string
61 # this should throw an exception: pytan.utils.HandlerError
62 import traceback
63
64 try:
65     handler.export_obj(**export_kwargs)
66 except Exception as e:
67     traceback.print_exc(file=sys.stdout)
```

### Output from Python Code

```
1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3279
2 Traceback (most recent call last):
3   File "<string>", line 66, in <module>
4   File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 1435, in export_obj
5     utils.check_dictkey(**check_args)
6   File "/Users/jolsen/gh/pytan/lib/pytan/utils.py", line 2531, in check_dictkey
7     raise HandlerError(err(key, valid_list_types, list_types))
8 HandlerError: 'header_sort' must be a list of [<type 'str'>, <type 'unicode'>], you supplied [<type
```

### Invalid export basetype csv bad sort type

Export a BaseType from getting objects using a bad header\_sort

### Example Python Code

```
1 import os
2 import sys
3 sys.dont_write_bytecode = True
4
5 # Determine our script name, script dir
6 my_file = os.path.abspath(sys.argv[0])
7 my_dir = os.path.dirname(my_file)
8
9 # determine the pytan lib dir and add it to the path
10 parent_dir = os.path.dirname(my_dir)
11 pytan_root_dir = os.path.dirname(parent_dir)
12 lib_dir = os.path.join(pytan_root_dir, 'lib')
```

```

13 path_adds = [lib_dir]
14
15 for aa in path_adds:
16     if aa not in sys.path:
17         sys.path.append(aa)
18
19
20 # connection info for Tanium Server
21 USERNAME = "Tanium User"
22 PASSWORD = "T@n!um"
23 HOST = "172.16.31.128"
24 PORT = "444"
25
26 # Logging conrols
27 LOGLEVEL = 2
28 DEBUGFORMAT = False
29
30 import tempfile
31
32 import pytan
33 handler = pytan.Handler(
34     username=USERNAME,
35     password=PASSWORD,
36     host=HOST,
37     port=PORT,
38     loglevel=LOGLEVEL,
39     debugformat=DEBUGFORMAT,
40 )
41
42 print handler
43
44 # setup the export_obj kwargs for later
45 export_kwargs = {}
46 export_kwargs["export_format"] = u'csv'
47 export_kwargs["header_sort"] = u'bad'
48
49 # get the objects that will provide the basetype that we want to use
50 get_kwargs = {
51     'name': [
52         "Computer Name", "IP Route Details", "IP Address",
53         'Folder Name Search with RegEx Match',
54     ],
55     'objtype': 'sensor',
56 }
57 response = handler.get(**get_kwargs)
58 export_kwargs['obj'] = response
59
60 # export the object to a string
61 # this should throw an exception: pytan.utils.HandlerError
62 import traceback
63
64 try:
65     handler.export_obj(**export_kwargs)
66 except Exception as e:
67     traceback.print_exc(file=sys.stdout)

```

## Output from Python Code

```
1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3279
2 Traceback (most recent call last):
3   File "<string>", line 66, in <module>
4   File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 1435, in export_obj
5     utils.check_dictkey(**check_args)
6   File "/Users/jolsen/gh/pytan/lib/pytan/utils.py", line 2524, in check_dictkey
7     raise HandlerError(err(key, valid_types, k_type))
8 HandlerError: 'header_sort' must be one of [<type 'bool'>, <type 'list'>, <type 'tuple'>], you supplied
```

### Invalid export basetype xml bad minimal type

Export a BaseType from getting objects using a bad minimal

### Example Python Code

```
1 import os
2 import sys
3 sys.dont_write_bytecode = True
4
5 # Determine our script name, script dir
6 my_file = os.path.abspath(sys.argv[0])
7 my_dir = os.path.dirname(my_file)
8
9 # determine the pytan lib dir and add it to the path
10 parent_dir = os.path.dirname(my_dir)
11 pytan_root_dir = os.path.dirname(parent_dir)
12 lib_dir = os.path.join(pytan_root_dir, 'lib')
13 path_adds = [lib_dir]
14
15 for aa in path_adds:
16     if aa not in sys.path:
17         sys.path.append(aa)
18
19
20 # connection info for Tanium Server
21 USERNAME = "Tanium User"
22 PASSWORD = "T@n!um"
23 HOST = "172.16.31.128"
24 PORT = "444"
25
26 # Logging conrols
27 LOGLEVEL = 2
28 DEBUGFORMAT = False
29
30 import tempfile
31
32 import pytan
33 handler = pytan.Handler(
34     username=USERNAME,
35     password=PASSWORD,
36     host=HOST,
37     port=PORT,
38     loglevel=LOGLEVEL,
39     debugformat=DEBUGFORMAT,
40 )
```



```

41
42 print handler
43
44 # setup the export_obj kwargs for later
45 export_kwargs = {}
46 export_kwargs["export_format"] = u'xml'
47 export_kwargs["minimal"] = u'bad'
48
49 # get the objects that will provide the basetype that we want to use
50 get_kwargs = {
51     'name': [
52         "Computer Name", "IP Route Details", "IP Address",
53         'Folder Name Search with RegEx Match',
54     ],
55     'objtype': 'sensor',
56 }
57 response = handler.get(**get_kwargs)
58 export_kwargs['obj'] = response
59
60 # export the object to a string
61 # this should throw an exception: pytan.utils.HandlerError
62 import traceback
63
64 try:
65     handler.export_obj(**export_kwargs)
66 except Exception as e:
67     traceback.print_exc(file=sys.stdout)

```

### Output from Python Code

```

1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3279
2 Traceback (most recent call last):
3   File "<string>", line 66, in <module>
4   File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 1435, in export_obj
5     utils.check_dictkey(**check_args)
6   File "/Users/jolsen/gh/pytan/lib/pytan/utils.py", line 2524, in check_dictkey
7     raise HandlerError(err(key, valid_types, k_type))
8 HandlerError: 'minimal' must be one of [<type 'bool'>], you supplied <type 'unicode'>!

```

### Invalid export basetype json bad include type

Export a BaseType from getting objects using a bad include\_type

### Example Python Code

```

1 import os
2 import sys
3 sys.dont_write_bytecode = True
4
5 # Determine our script name, script dir
6 my_file = os.path.abspath(sys.argv[0])
7 my_dir = os.path.dirname(my_file)
8
9 # determine the pytan lib dir and add it to the path

```

```
10 parent_dir = os.path.dirname(my_dir)
11 pytan_root_dir = os.path.dirname(parent_dir)
12 lib_dir = os.path.join(pytan_root_dir, 'lib')
13 path_adds = [lib_dir]
14
15 for aa in path_adds:
16     if aa not in sys.path:
17         sys.path.append(aa)
18
19
20 # connection info for Tanium Server
21 USERNAME = "Tanium User"
22 PASSWORD = "T@n!um"
23 HOST = "172.16.31.128"
24 PORT = "444"
25
26 # Logging conrols
27 LOGLEVEL = 2
28 DEBUGFORMAT = False
29
30 import tempfile
31
32 import pytan
33 handler = pytan.Handler(
34     username=USERNAME,
35     password=PASSWORD,
36     host=HOST,
37     port=PORT,
38     loglevel=LOGLEVEL,
39     debugformat=DEBUGFORMAT,
40 )
41
42 print handler
43
44 # setup the export_obj kwargs for later
45 export_kwargs = {}
46 export_kwargs["export_format"] = u'json'
47 export_kwargs["include_type"] = u'bad'
48
49 # get the objects that will provide the basetype that we want to use
50 get_kwargs = {
51     'name': [
52         "Computer Name", "IP Route Details", "IP Address",
53         'Folder Name Search with RegEx Match',
54     ],
55     'objtype': 'sensor',
56 }
57 response = handler.get(**get_kwargs)
58 export_kwargs['obj'] = response
59
60 # export the object to a string
61 # this should throw an exception: pytan.utils.HandlerError
62 import traceback
63
64 try:
65     handler.export_obj(**export_kwargs)
66 except Exception as e:
```

```
67 traceback.print_exc(file=sys.stdout)
```

### Output from Python Code

```
1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3279
2 Traceback (most recent call last):
3   File "<string>", line 66, in <module>
4   File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 1435, in export_obj
5     utils.check_dictkey(**check_args)
6   File "/Users/jolsen/gh/pytan/lib/pytan/utils.py", line 2524, in check_dictkey
7     raise HandlerError(err(key, valid_types, k_type))
8 HandlerError: 'include_type' must be one of [<type 'bool'>], you supplied <type 'unicode'>!
```

### Invalid export basetype json bad explode type

Export a BaseType from getting objects using a bad explode\_json\_string\_values

### Example Python Code

```
1 import os
2 import sys
3 sys.dont_write_bytecode = True
4
5 # Determine our script name, script dir
6 my_file = os.path.abspath(sys.argv[0])
7 my_dir = os.path.dirname(my_file)
8
9 # determine the pytan lib dir and add it to the path
10 parent_dir = os.path.dirname(my_dir)
11 pytan_root_dir = os.path.dirname(parent_dir)
12 lib_dir = os.path.join(pytan_root_dir, 'lib')
13 path_adds = [lib_dir]
14
15 for aa in path_adds:
16     if aa not in sys.path:
17         sys.path.append(aa)
18
19
20 # connection info for Tanium Server
21 USERNAME = "Tanium User"
22 PASSWORD = "T@n!um"
23 HOST = "172.16.31.128"
24 PORT = "444"
25
26 # Logging conrols
27 LOGLEVEL = 2
28 DEBUGFORMAT = False
29
30 import tempfile
31
32 import pytan
33 handler = pytan.Handler(
34     username=USERNAME,
35     password=PASSWORD,
```

```
36     host=HOST,
37     port=PORT,
38     loglevel=LOGLEVEL,
39     debugformat=DEBUGFORMAT,
40 )
41
42 print handler
43
44 # setup the export_obj kwargs for later
45 export_kwargs = {}
46 export_kwargs["export_format"] = u'json'
47 export_kwargs["explode_json_string_values"] = u'bad'
48
49 # get the objects that will provide the basetype that we want to use
50 get_kwargs = {
51     'name': [
52         "Computer Name", "IP Route Details", "IP Address",
53         'Folder Name Search with RegEx Match',
54     ],
55     'objtype': 'sensor',
56 }
57 response = handler.get(**get_kwargs)
58 export_kwargs['obj'] = response
59
60 # export the object to a string
61 # this should throw an exception: pytan.utils.HandlerError
62 import traceback
63
64 try:
65     handler.export_obj(**export_kwargs)
66 except Exception as e:
67     traceback.print_exc(file=sys.stdout)
```

## Output from Python Code

```
1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3279
2 Traceback (most recent call last):
3   File "<string>", line 66, in <module>
4   File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 1435, in export_obj
5     utils.check_dictkey(**check_args)
6   File "/Users/jolsen/gh/pytan/lib/pytan/utils.py", line 2524, in check_dictkey
7     raise HandlerError(err(key, valid_types, k_type))
8 HandlerError: 'explode_json_string_values' must be one of [<type 'bool'>], you supplied <type 'unicode'>
```

## Invalid export basetype bad format

Export a BaseType from getting objects using a bad export\_format

## Example Python Code

```
1 import os
2 import sys
3 sys.dont_write_bytecode = True
4
```

```

5  # Determine our script name, script dir
6  my_file = os.path.abspath(sys.argv[0])
7  my_dir = os.path.dirname(my_file)
8
9  # determine the pytan lib dir and add it to the path
10 parent_dir = os.path.dirname(my_dir)
11 pytan_root_dir = os.path.dirname(parent_dir)
12 lib_dir = os.path.join(pytan_root_dir, 'lib')
13 path_adds = [lib_dir]
14
15 for aa in path_adds:
16     if aa not in sys.path:
17         sys.path.append(aa)
18
19
20 # connection info for Tanium Server
21 USERNAME = "Tanium User"
22 PASSWORD = "T@n!um"
23 HOST = "172.16.31.128"
24 PORT = "444"
25
26 # Logging conrols
27 LOGLEVEL = 2
28 DEBUGFORMAT = False
29
30 import tempfile
31
32 import pytan
33 handler = pytan.Handler(
34     username=USERNAME,
35     password=PASSWORD,
36     host=HOST,
37     port=PORT,
38     loglevel=LOGLEVEL,
39     debugformat=DEBUGFORMAT,
40 )
41
42 print handler
43
44 # setup the export_obj kwargs for later
45 export_kwargs = {}
46 export_kwargs["export_format"] = u'bad'
47
48 # get the objects that will provide the basetype that we want to use
49 get_kwargs = {
50     'name': [
51         "Computer Name", "IP Route Details", "IP Address",
52         'Folder Name Search with RegEx Match',
53     ],
54     'objtype': 'sensor',
55 }
56 response = handler.get(**get_kwargs)
57 export_kwargs['obj'] = response
58
59 # export the object to a string
60 # this should throw an exception: pytan.utils.HandlerError
61 import traceback
62

```

```
63 try:
64     handler.export_obj(**export_kwargs)
65 except Exception as e:
66     traceback.print_exc(file=sys.stdout)
```

### Output from Python Code

```
1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3279
2 Traceback (most recent call last):
3   File "<string>", line 65, in <module>
4   File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 1429, in export_obj
5     raise HandlerError(err)
6 HandlerError: u'bad' not a supported export format for SensorList, must be one of: xml, json, csv
```

## 1.2.2 pytan.handler module

The main `pytan` module that provides methods for programmatic use.

### Handler Class

**class** `pytan.handler.Handler` (*username, password, host, port='444', loglevel=0, debugformat=False, get\_version=True, \*\*kwargs*)

Bases: `object`

Creates a connection to a Tanium SOAP Server on *host:port*

**Parameters** **username** : str

*username* to connect to *host* with

**password** : str

*password* to connect to *host* with

**host** : str

hostname or ip of Tanium SOAP Server

**port** : int, optional

port of Tanium SOAP Server on *host*

**loglevel** : int, optional

0 should not print anything, 1 and higher will print more

**debugformat** : bool, optional

False use one line logformat, True use two lines

**See also:**

`pytan.constants.LOG_LEVEL_MAPS` maps a given *loglevel* to respective logger names and their logger levels

`pytan.constants.INFO_FORMAT` *debugformat=False*

`pytan.constants.DEBUG_FORMAT` *debugformat=True*

## Notes

- port 444 is the default SOAP port
- port 443 forwards /soap/ URLs to the SOAP port
- Use port 444 if you have direct access to it

### Example: Create a Handler object

Setup a Handler() object:

```
>>> import sys
>>> sys.path.append('/path/to/pytan/')
>>> import pytan
>>> handler = pytan.Handler('username', 'password', 'host')
```

## Handler Methods: Questions and Actions

### Ask a Question

Handler.**ask** (\*\*kwargs)

Ask a type of question and get the results back

**Parameters** **qtype** : str

type of question to ask: saved\_question, manual, or manual\_human

**Returns** **result** : dict, containing:

- *question\_object* : one of the following depending on *qtype*:  
   taniumpy.object\_types.question.Question or  
   taniumpy.object\_types.saved\_question.SavedQuestion
- *question\_results* : taniumpy.object\_types.result\_set.ResultSet

See also:

`pytan.constants.Q_OBJ_MAP` maps qtype to a method in Handler()

### Ask a Saved Question

Handler.**ask\_saved** (\*\*kwargs)

Ask a saved question and get the results back

**Parameters** **id** : int, list of int, optional

id of saved question to ask

**name** : str, list of str

name of saved question

**Returns** **ret** : dict, containing

- *question\_object* : taniumpy.object\_types.saved\_question.SavedQuestion
- *question\_results* : taniumpy.object\_types.result\_set.ResultSet

See also:

`pytan.constants.ASK_KWARGS` list of kwargs that can be passed to `taniumpy.question_asker.QuestionAsker`

## Notes

id or name must be supplied

## Asking a Manual Question

Handler `.ask_manual` (*get\_results=True, \*\*kwargs*)

Ask a manual question using definitions and get the results back

This method requires in-depth knowledge of how filters and options are created in the API, and as such is not meant for human consumption. Use `ask_manual_human()` instead.

**Parameters** `sensor_defs` : str, dict, list of str or dict

sensor definitions

`question_filter_defs` : dict, list of dict, optional

question filter definitions

`question_option_defs` : dict, list of dict, optional

question option definitions

`get_results` : bool, optional

- True: wait for result completion after asking question
- False: just ask the question and return it in *ret*

**Returns** `ret` : dict, containing:

- `question_object` : `taniumpy.object_types.question.Question`
- `question_results` : `taniumpy.object_types.result_set.ResultSet`

See also:

`pytan.constants.FILTER_MAPS` valid filter dictionaries for filters

`pytan.constants.OPTION_MAPS` valid option dictionaries for options

`pytan.constants.ASK_KWARGS` list of kwargs that can be passed to `taniumpy.question_asker.QuestionAsker`

## Examples

```
>>> # example of str for sensor_defs
>>> sensor_defs = 'Sensor1'
```

```
>>> # example of dict for sensor_defs
>>> sensor_defs = {
...     'name': 'Sensor1',
...     'filter': {
```



```

...     'operator': 'RegexMatch',
...     'not_flag': 0,
...     'value': '.*'
... },
...     'params': {'key': 'value'},
...     'options': {'and_flag': 1}
... }

```

```

>>> # example of dict for question_filter_defs
>>> question_filter_defs = {
...     'operator': 'RegexMatch',
...     'not_flag': 0,
...     'value': '.*'
... }

```

`Handler.ask_manual_human(**kwargs)`

Ask a manual question using human strings and get the results back

This method takes a string or list of strings and parses them into their corresponding definitions needed by `ask_manual()`

**Parameters** `sensors` : str, list of str

sensors (columns) to include in question

**question\_filters** : str, list of str, optional

filters that apply to the whole question

**question\_options** : str, list of str, optional

options that apply to the whole question

**get\_results** : bool, optional

- True: wait for result completion after asking question
- False: just ask the question and return it in result

**sensors\_help** : bool, optional

- False: do not print the help string for sensors
- True: print the help string for sensors and exit

**filters\_help** : bool, optional

- False: do not print the help string for filters
- True: print the help string for filters and exit

**options\_help** : bool, optional

- False: do not print the help string for options
- True: print the help string for options and exit

**Returns** `result` : dict, containing:

- `question_object` : `taniumpy.object_types.question.Question`
- `question_results` : `taniumpy.object_types.result_set.ResultSet`

See also:

`pytan.constants.FILTER_MAPS` valid filter dictionaries for filters

`pytan.constants.OPTION_MAPS` valid option dictionaries for options

`pytan.constants.ASK_KWARGS` list of kwargs that can be passed to `taniumpy.question_asker.QuestionAsker`

## Examples

```
>>> # example of str for `sensors`
>>> sensors = 'Sensor1'
```

```
>>> # example of str for `sensors` with params
>>> sensors = 'Sensor1{key:value}'
```

```
>>> # example of str for `sensors` with params and filter
>>> sensors = 'Sensor1{key:value}, that contains:example text'
```

```
>>> # example of str for `sensors` with params and filter and options
>>> sensors = (
...     'Sensor1{key:value}, that contains:example text,'
...     'opt:ignore_case, opt:max_data_age:60'
... )
```

```
>>> # example of str for question_filters
>>> question_filters = 'Sensor2, that contains:example test'
```

```
>>> # example of list of str for question_options
>>> question_options = ['max_data_age:3600', 'and']
```

## Deploy an Action

`Handler.deploy_action(run=False, get_results=True, **kwargs)`

Deploy an action and get the results back

This method requires in-depth knowledge of how filters and options are created in the API, and as such is not meant for human consumption. Use `deploy_action_human()` instead.

**Parameters** `package_def`: dict

definition that describes a package

**action\_filter\_defs**: str, dict, list of str or dict, optional

action filter definitions

**action\_option\_defs**: dict, list of dict, optional

action filter option definitions

**start\_seconds\_from\_now**: int, optional

start action N seconds from now

**expire\_seconds**: int, optional

expire action N seconds from now, will be derived from package if not supplied

**run** : bool, optional

- False: just ask the question that pertains to verify action, export the results to CSV, and raise RunFalse – does not deploy the action
- True: actually deploy the action

**get\_results** : bool, optional

- True: wait for result completion after deploying action
- False: just deploy the action and return the object in *ret*

**Returns** **ret** : dict, containing:

- *action\_object* : `taniumpy.object_types.action.Action`
- *action\_results* : `taniumpy.object_types.result_set.ResultSet`
- *action\_progress\_human* : str, progress map in human form
- *action\_progress\_map* : dict, progress map in dictionary form
- *pre\_action\_question\_results* : `taniumpy.object_types.result_set.ResultSet`

See also:

`pytan.constants.FILTER_MAPS` valid filter dictionaries for filters

`pytan.constants.OPTION_MAPS` valid option dictionaries for options

## Examples

```
>>> # example of dict for `package_def`
>>> package_def = {'name': 'PackageName1', 'params':{'param1': 'value1'}}
```

```
>>> # example of str for `action_filter_defs`
>>> action_filter_defs = 'Sensor1'
```

```
>>> # example of dict for `action_filter_defs`
>>> action_filter_defs = {
...     'name': 'Sensor1',
...     'filter': {
...         'operator': 'RegexMatch',
...         'not_flag': 0,
...         'value': '.*'
...     },
...     'options': {'and_flag': 1}
... }
```

Handler.**deploy\_action\_human** (\*\*kwargs)

Deploy an action and get the results back

This method takes a string or list of strings and parses them into their corresponding definitions needed by `deploy_action()`

**Parameters** **package** : str

each string must describe a package

**action\_filters** : str, list of str, optional

each string must describe a sensor and a filter which limits which computers the action will deploy *package* to

**action\_options** : str, list of str, optional

options to apply to *action\_filters*

**start\_seconds\_from\_now** : int, optional

start action N seconds from now

**expire\_seconds** : int, optional

expire action N seconds from now, will be derived from package if not supplied

**run** : bool, optional

- False: just ask the question that pertains to verify action, export the results to CSV, and raise RunFalse – does not deploy the action
- True: actually deploy the action

**get\_results** : bool, optional

- True: wait for result completion after deploying action
- False: just deploy the action and return the object in *ret*

**package\_help** : bool, optional

- False: do not print the help string for package
- True: print the help string for package and exit

**filters\_help** : bool, optional

- False: do not print the help string for filters
- True: print the help string for filters and exit

**options\_help** : bool, optional

- False: do not print the help string for options
- True: print the help string for options and exit

**Returns** *ret* : dict, containing:

- *action\_object* : `taniumpy.object_types.action.Action`
- *action\_results* : `taniumpy.object_types.result_set.ResultSet`
- *action\_progress\_human* : str, progress map in human form
- *action\_progress\_map* : dict, progress map in dictionary form
- *pre\_action\_question\_results* : `taniumpy.object_types.result_set.ResultSet`

See also:

`pytan.constants.FILTER_MAPS` valid filter dictionaries for filters

`pytan.constants.OPTION_MAPS` valid option dictionaries for options

## Examples

```
>>> # example of str for `package`
>>> package = 'Package1'
```

```
>>> # example of str for `package` with params
>>> package = 'Package1{key:value}'
```

```
>>> # example of str for `action_filters` with params and filter for sensors
>>> action_filters = 'Sensor1{key:value}, that contains:example text'
```

```
>>> # example of list of str for `action_options`
>>> action_options = ['max_data_age:3600', 'and']
```

`Handler.deploy_action_asker(action_id, passed_count=0)`

Checks the results of a deploy action job and waits for completion

**Parameters** `action_id` : int

id of deploy action to get results for and wait on completion

`passed_count` : int, optional

the number of servers that must equate “completed” in order for deploy action to be recognized as completed

**Returns** `ret` : dict, containing:

- `action_object` : `taniumpy.object_types.action.Action`
- `action_results` : `taniumpy.object_types.result_set.ResultSet`
- `action_progress_human` : str, progress map in human form
- `action_progress_map` : dict, progress map in dictionary form

See also:

`pytan.constants.ACTION_RESULT_STATUS` maps the values in *Action Statuses* columns to success/completed/failed/etc

## Stopping an Action

`Handler.stop_action(id, **kwargs)`

Stop an action

**Parameters** `id` : int

id of action to stop

**Returns** `action_stop_obj` : `taniumpy.object_types.action_stop.ActionStop`

The object containing the ID of the action stop job

## Handler Methods: Exporting/Importing Objects

### Import an API Object from JSON

`Handler.create_from_json(objtype, json_file)`

Creates a new object using the SOAP api from a json file

**Parameters** `objtype` : str

Type of object described in `json_file`

`json_file` : str

path to JSON file that describes an API object

**Returns** `ret` : `taniumpy.object_types.base.BaseType`

TaniumPy object added to Tanium SOAP Server

See also:

`pytan.constants.GET_OBJ_MAP` maps `objtype` to supported 'create\_json' types

### Load a Python Object from JSON

`Handler.load_taniumpy_from_json(json_file)`

Opens a json file and parses it into an `taniumpy` object

**Parameters** `json_file` : str

path to JSON file that describes an API object

**Returns** `obj` : `taniumpy.object_types.base.BaseType`

TaniumPy object converted from json file

### Export Object

`Handler.export_obj(obj, export_format, **kwargs)`

Exports a python API object to a given export format

**Parameters** `obj` : `taniumpy.object_types.base.BaseType` or `taniumpy.object_types.result_set.ResultSet`

TaniumPy object to export

`export_format` : str

the number of servers that must equate "completed" in order for deploy action to be recognized as completed

`header_sort` : list of str, bool, optional

- for `export_format` csv and `obj` types `taniumpy.object_types.base.BaseType` or `taniumpy.object_types.result_set.ResultSet`
- True: sort the headers automatically
- False: do not sort the headers at all
- list of str: sort the headers returned by priority based on provided list

`header_add_sensor` : bool, optional

- for `export_format` csv and `obj` type `taniumpy.object_types.result_set.ResultSet`
- False: do not prefix the headers with the associated sensor name for each column

- True: prefix the headers with the associated sensor name for each column

**header\_add\_type** : bool, optional

- for *export\_format* csv and *obj* type `taniumpy.object_types.result_set.ResultSet`
- False: do not postfix the headers with the result type for each column
- True: postfix the headers with the result type for each column

**expand\_grouped\_columns** : bool, optional

- for *export\_format* csv and *obj* type `taniumpy.object_types.result_set.ResultSet`
- False: do not expand multiline row entries into their own rows
- True: expand multiline row entries into their own rows

**explode\_json\_string\_values** : bool, optional

- for *export\_format* json or csv and *obj* type `taniumpy.object_types.base.BaseType`
- False: do not explode JSON strings in object attributes into their own object attributes
- True: explode JSON strings in object attributes into their own object attributes

**minimal** : bool, optional

- for *export\_format* xml and *obj* type `taniumpy.object_types.base.BaseType`
- False: include empty attributes in XML output
- True: do not include empty attributes in XML output

**Returns** **result** : str

the contents of exporting *export\_format*

See also:

`pytan.constants.EXPORT_MAPS` maps the type *obj* to *export\_format* and the optional args supported for each

## Export Object to Report File

Handler.**export\_to\_report\_file** (*obj*, *export\_format*, *\*\*kwargs*)

Exports a python API object to a file

**Parameters** **obj** : `taniumpy.object_types.base.BaseType` or `taniumpy.object_types.result_set.ResultSet`

TaniumPy object to export

**export\_format** : str

the format to export *obj* to, can be one of: csv, xml, json

**header\_sort** : list of str, bool, optional

- for *export\_format* csv and *obj* types `taniumpy.object_types.base.BaseType` or `taniumpy.object_types.result_set.ResultSet`
- True: sort the headers automatically
- False: do not sort the headers at all
- list of str: sort the headers returned by priority based on provided list

**header\_add\_sensor** : bool, optional

- for *export\_format* csv and *obj* type `taniumpy.object_types.result_set.ResultSet`
- False: do not prefix the headers with the associated sensor name for each column
- True: prefix the headers with the associated sensor name for each column

**header\_add\_type** : bool, optional

- for *export\_format* csv and *obj* type `taniumpy.object_types.result_set.ResultSet`
- False: do not postfix the headers with the result type for each column
- True: postfix the headers with the result type for each column

**expand\_grouped\_columns** : bool, optional

- for *export\_format* csv and *obj* type `taniumpy.object_types.result_set.ResultSet`
- False: do not expand multiline row entries into their own rows
- True: expand multiline row entries into their own rows

**explode\_json\_string\_values** : bool, optional

- for *export\_format* json or csv and *obj* type `taniumpy.object_types.base.BaseType`
- False: do not explode JSON strings in object attributes into their own object attributes
- True: explode JSON strings in object attributes into their own object attributes

**minimal** : bool, optional

- for *export\_format* xml and *obj* type `taniumpy.object_types.base.BaseType`
- False: include empty attributes in XML output
- True: do not include empty attributes in XML output

**report\_file**: str, optional

filename to save report as, will be automatically generated if not supplied

**report\_dir**: str, optional

directory to save report in, if not supplied, will be extracted from *report\_file*. if no directory in *report\_file* or *report\_file* not specified, will use current working directory.

**prefix**: str, optional

prefix to add to *report\_file*

**postfix**: str, optional

postfix to add to *report\_file*

**Returns** **report\_path** : str

the full path to the file created with contents of *result*

**result** : str

the str of *export\_format*

---



## Handler Methods: Creating Objects

### Create a Group

`Handler.create_group(groupname, filters=[], filter_options=[], **kwargs)`

Create a group object

**Parameters** `groupname` : str

name of group to create

**filters** : str or list of str, optional

each string must describe a filter

**filter\_options** : str or list of str, optional

each string must describe an option for *filters*

**filters\_help** : bool, optional

- False: do not print the help string for filters
- True: print the help string for filters and exit

**options\_help** : bool, optional

- False: do not print the help string for options
- True: print the help string for options and exit

**Returns** `group_obj` : `taniumpy.object_types.group.Group`

TaniumPy object added to Tanium SOAP Server

See also:

`pytan.constants.FILTER_MAPS` valid filters for filters

`pytan.constants.OPTION_MAPS` valid options for filter\_options

### Create a Package

`Handler.create_package(name, command, display_name='', file_urls=[], command_timeout_seconds=600, expire_seconds=600, parameters_json_file='', verify_filters=[], verify_filter_options=[], verify_expire_seconds=600, **kwargs)`

Create a package object

**Parameters** `name` : str

name of package to create

**command** : str

command to execute

**display\_name** : str, optional

display name of package

**file\_urls** : list of strings, optional

- URL of file to add to package
- can optionally define download\_seconds by using SECONDS::URL
- can optionally define file name by using FILENAME||URL

- can combine optionals by using SECONDS::FILENAME||URL
- FILENAME will be extracted from basename of URL if not provided

**command\_timeout\_seconds** : int, optional

timeout for command execution in seconds

**parameters\_json\_file** : str, optional

path to json file describing parameters for package

**expire\_seconds** : int, optional

timeout for action expiry in seconds

**verify\_filters** : str or list of str, optional

each string must describe a filter to be used to verify the package

**verify\_filter\_options** : str or list of str, optional

each string must describe an option for *verify\_filters*

**verify\_expire\_seconds** : int, optional

timeout for verify action expiry in seconds

**filters\_help** : bool, optional

- False: do not print the help string for filters
- True: print the help string for filters and exit

**options\_help** : bool, optional

- False: do not print the help string for options
- True: print the help string for options and exit

**metadata**: list of list of strs, optional

- each list must be a 2 item list:
- list item 1 property name
- list item 2 property value

**Returns** **package\_obj** : `taniumpy.object_types.package_spec.PackageSpec`

TaniumPy object added to Tanium SOAP Server

**See also:**

`pytan.constants.FILTER_MAPS` valid filters for `verify_filters`

`pytan.constants.OPTION_MAPS` valid options for `verify_filter_options`

### Create a Sensor

`Handler.create_sensor()`

Create a sensor object

**Raises** **HandlerError** : `pytan.utils.HandlerError`

**Warning:** Not currently supported, too complicated to add. Use `create_from_json()` instead for this object type!

### Create a User

`Handler.create_user (username, rolename=[], roleid=[], properties=[])`

Create a user object

**Parameters** `username` : str

name of user to create

**rolename** : str or list of str, optional

name(s) of roles to add to user

**roleid** : int or list of int, optional

id(s) of roles to add to user

**properties**: list of list of str, optional

- each list must be a 2 item list:
- list item 1 property name
- list item 2 property value

**Returns** `user_obj` : `taniumpy.object_types.user.User`

TaniumPy object added to Tanium SOAP Server

### Create a Whitelisted URL

`Handler.create_whitelisted_url (url, regex=False, download_seconds=86400, properties=[])`

Create a whitelisted url object

**Parameters** `url` : str

text of new url

**regex** : bool, optional

- True: *url* is a regex pattern
- False: *url* is not a regex pattern

**download\_seconds** : int, optional

how often to re-download *url*

**properties**: list of list of str, optional

- each list must be a 2 item list:
- list item 1 property name
- list item 2 property value

**Returns** `url_obj` : `taniumpy.object_types.white_listed_url.WhiteListedUrl`

TaniumPy object added to Tanium SOAP Server

---

## Handler Methods: Deleting Objects

### Delete an Object

`Handler.delete (objtype, **kwargs)`

Delete an object type

**Parameters** `objtype` : string

type of object to delete

**id/name/hash** : int or string, list of int or string

search attributes of object to delete, must supply at least one valid search attr

**Returns** `ret` : dict

dict containing deploy action object and results from deploy action

**See also:**

`pytan.constants.GET_OBJ_MAP` maps objtype to supported 'search' keys

---

## Handler Methods: Getting Objects

### Get Single or Multiple Objects of a type

`Handler.get(objtype, **kwargs)`

Get an object type

**Parameters** `objtype` : string

type of object to get

**id/name/hash** : int or string, list of int or string

search attributes of object to get, must supply at least one valid search attr

**See also:**

`pytan.constants.GET_OBJ_MAP` maps objtype to supported 'search' keys

### Get All Objects of a type

`Handler.get_all(objtype, **kwargs)`

Get all objects of a type

**Parameters** `objtype` : string

type of object to get

**See also:**

`pytan.constants.GET_OBJ_MAP` maps objtype to supported 'search' keys

## Handler Methods: Getting Result Data / Result Info

`Handler.get_result_data(obj, aggregate=False, **kwargs)`

Get the result data for a python API object

This method issues a GetResultData command to the SOAP api for *obj*. GetResultData returns the columns and rows that are currently available for *obj*.

**Parameters** `obj` : `taniumpy.object_types.base.BaseType`

object to get result data for

**aggregate** : bool, optional

- False: get all the data
- True: get just the aggregate data (row counts of matches)

**Returns** `rd`: `taniumpy.object_types.result_set.ResultSet`

The return of `GetResultData` for *obj*

`Handler.get_result_info(obj, **kwargs)`

Get the result info for a python API object

This method issues a `GetResultInfo` command to the SOAP api for *obj*. `GetResultInfo` returns information about how many servers have passed the *obj*, total number of servers, and so on.

**Parameters** `obj`: `taniumpy.object_types.base.BaseType`

object to get result data for

**Returns** `ri`: `taniumpy.object_types.result_info.ResultInfo`

The return of `GetResultData` for *obj*

---

## Handler Methods: Private Methods

`Handler._find(api_object, **kwargs)`

Wrapper for interfacing with `taniumpy.session.Session.find()`

`Handler._get_multi(obj_map, **kwargs)`

Find multiple item wrapper using `_find()`

`Handler._get_single(obj_map, **kwargs)`

Find single item wrapper using `_find()`

`Handler._single_find(obj_map, k, v, **kwargs)`

Wrapper for single item searches interfacing with `taniumpy.session.Session.find()`

`Handler._get_sensor_defs(defs)`

Uses `get()` to update a definition with a sensor object

`Handler._get_package_def(d)`

Uses `get()` to update a definition with a package object

`Handler._export_class_BaseType(obj, export_format, **kwargs)`

Handles exporting `taniumpy.object_types.base.BaseType`

`Handler._export_class_ResultSet(obj, export_format, **kwargs)`

Handles exporting `taniumpy.object_types.result_set.ResultSet`

`Handler._export_format_csv(obj, **kwargs)`

Handles exporting format: CSV

`Handler._export_format_json(obj, **kwargs)`

Handles exporting format: JSON

`Handler._export_format_xml(obj, **kwargs)`

Handles exporting format: XML

### 1.2.3 pytan.constants module

#### PyTan Constants

This contains a number of constants that drive PyTan.

```
pytan.constants.ACTION_RESULT_STATUS = {'Verified.': ['no_verify_done', 'verify_done', 'verify_success'], 'Succeeded': ['no_verify_done', 'verify_done', 'verify_success']}
```

Maps a deploy action result status to it's respective end states.

```
pytan.constants.ASK_KWARGS = ['timeout', 'polling_interval', 'pct_complete_threshold']
```

A list of arguments that will be passed on to the question asker/poller  
`taniumpy.question_asker.QuestionAsker`

```
pytan.constants.DEBUG_FORMAT = "[% (lineno)-5d - %(filename)20s: %(funcName)s()] %(asctime)s\n%(levelname)-8s %s"
```

Logging format for debugformat=True

```
pytan.constants.EXPORT_MAPS = {'ResultSet': {'json': [], 'csv': [{'valid_list_types': ['str', 'unicode'], 'key': 'header_source', 'value': 'header_source'}]}}
```

Maps a given TaniumPy object to the list of supported export formats for each object type, and the valid optional arguments

- key: the optional argument name itself
- valid\_types: the valid python types that are allowed to be passed as a value to *key*
- valid\_list\_types: the valid python types in str format that are allowed to be passed in a list, if list is one of the *valid\_types*

```
pytan.constants.FILTER_MAPS = [{'operator': 'Less', 'not_flag': 0, 'help': 'Filter for less than VALUE', 'human': ['<', 'less than'], 'pre_value': 'less than', 'post_value': 'less than'}],
```

Maps a given set of human strings into the various filter attributes used by the SOAP API. Also used to verify that a manual search is allowed

- human: a list of human strings that can be used after '*that*'. Ex: '*that* contains value'
- operator: the filter operator used by the SOAP API when building a filter that matches *human*
- not\_flag: the value to set on *not\_flag* when building a filter that matches *human*
- pre\_value: the prefix to add to the *value* when building a filter
- post\_value: the postfix to add to the *value* when building a filter

```
pytan.constants.FILTER_RE = ',\\s*that'
```

The regex that is used to find filters in a string. Ex: *Sensor1, that contains blah*

```
pytan.constants.GET_OBJ_MAP = {'user': {'search': ['id'], 'all': 'UserList', 'manual': True, 'multi': None, 'single': 'UserList', 'delete': True, 'create_json': True}}
```

Maps an object type from a human friendly string into various aspects:

- single: The TaniumPy object used to find singular instances of this object type
- multi: The TaniumPy object used to find multiple instances of this object type
- all: The TaniumPy object used to find all instances of this object type
- search: The list of attributes that can be used with the Tanium SOAP API for searches
- manual: Whether or not this object type is allowed to do a manual search, that is – allow the user to specify an attribute that is not in search, which will get ALL objects of that type then search for a match based on attribute values for EVERY key/value pair supplied
- delete: Whether or not this object type can be deleted
- create\_json: Whether or not this object type can be created by importing from JSON

`pytan.constants.INFO_FORMAT = '%(asctime)s %(levelname)-8s %(name)s: %(message)s'`  
 Logging format for debugformat=False

`pytan.constants.LOG_LEVEL_MAPS = [(0, {'api.session.http': 'WARN', 'api.session': 'WARN', 'handler': 'WARN', 'ques`

**Map for loglevel(int) -> logger -> logger level(logging.INFO|WARN|DEBUG|...). Higher loglevels will include all levels up**

- int, loglevel
- dict, `{{logger_name: logger_level}}` for this loglevel

`pytan.constants.OPTION_MAPS = [{'destination': 'filter', 'help': 'Make the filter do a case insensitive match', 'attrs': {'ig`

**Maps a given human string into the various options for filters used by the SOAP API. Also used to verify that a manually**

- human: the human string that can be used after `'opt:'`. Ex: `'opt:value_type:value'`
- destination: the type of object this option can be applied to (filter or group)
- attrs: the attributes and their values used by the SOAP API when building a filter with an option that matches *human*
- attr: the attribute used by the SOAP API when building a filter with an option that matches *human*. value is pulled from after a `:` when only attr exists for an option map, and not attrs.
- valid\_values: if supplied, the list of valid values for this option
- valid\_type: performs type checking on the value supplied to verify it is correct
- human\_type: the human string for the value type if the option requires a value

`pytan.constants.OPTION_RE = ',\\s*opt:'`

The regex that is used to find options in a string. Ex: *Sensor1, that contains blah, opt:ignore\_case, opt:max\_data\_age:3600*

`pytan.constants.PARAM_DELIM = '||'`

The string to surround a parameter with when passing parameters to the SOAP API for a sensor in a question.  
 Ex: `||parameter_key||`

`pytan.constants.PARAM_KEY_SPLIT = '='`

The string that is used to split parameter key from parameter value. Ex: *key1=value1*

`pytan.constants.PARAM_RE = '(?<!\\|\\|)\\{(.*)\\}(?<!\\|\\|)'`

The regex that is used to parse parameters from a human string. Ex: *ala {key1=value1}*

`pytan.constants.PARAM_SPLIT_RE = '(?<!\\|\\|),'`

The regex that is used to split multiple parameters. Ex: *key1=value1, key2=value2*

`pytan.constants.Q_OBJ_MAP = {'manual': {'handler': 'ask_manual'}, 'saved': {'handler': 'ask_saved'}, 'manual_human`

Maps a question type from a human friendly string into the handler method that supports each type

`pytan.constants.REQ_KWARGS = ['hide_errors_flag', 'include_answer_times_flag', 'row_counts_only_flag', 'aggregate_ov`

A list of arguments that will be pulled from any respective kwargs for most calls to  
`taniumpy.session.Session`

`pytan.constants.SELECTORS = ['id', 'name', 'hash']`

The search selectors that can be extracted from a string. Ex: *name:Sensor1, or id:1, or hash:1111111*

`pytan.constants.SENSOR_TYPE_MAP = {0: 'Hash', 1: 'String', 2: 'Version', 3: 'NumericDecimal', 4: 'BESDate', 5: 'IPAC`

Maps a Result type from the Tanium SOAP API from an int to a string

## 1.2.4 pytan.utils module

Collection of exceptions, classes, and methods used throughout `pytan`

### Utility Classes: Exceptions

Exceptions used throughout `pytan`:

**exception** `pytan.utils.HandlerError`

Bases: `exceptions.Exception`

Exception thrown for most errors in `pytan.handler`

**exception** `pytan.utils.HumanParserError`

Bases: `exceptions.Exception`

Exception thrown for errors while parsing human strings from `pytan.handler`

**exception** `pytan.utils.DefinitionParserError`

Bases: `exceptions.Exception`

Exception thrown for errors while parsing definitions from `pytan.handler`

**exception** `pytan.utils.RunFalse`

Bases: `exceptions.Exception`

Exception thrown when `run=False` from `pytan.handler.Handler.deploy_action()`

### Utility Classes: Logging handlers

**class** `pytan.utils.SplitStreamHandler`

Bases: `logging.Handler`

Custom logging.Handler class that sends all messages that are logging.INFO and below to STDOUT, and all messages that are logging.WARNING and above to STDERR

**emit** (*record*)

### Utility Classes: Argument Parsers for Command Line Scripts

**class** `pytan.utils.CustomArgFormat` (*prog*, *indent\_increment=2*, *max\_help\_position=24*, *width=None*)

Bases: `argparse.ArgumentDefaultsHelpFormatter`, `argparse.RawDescriptionHelpFormatter`

Multiple inheritance Formatter class for `argparse.ArgumentParser`.

If a `argparse.ArgumentParser` class uses this as it's Formatter class, it will show the defaults for each argument in the *help* output

**class** `pytan.utils.CustomArgParse` (*\*args*, *\*\*kwargs*)

Bases: `argparse.ArgumentParser`

Custom `argparse.ArgumentParser` class which does a number of things:

- Uses `pytan.utils.CustomArgFormat` as it's Formatter class, if none was passed in
- Prints help if there is an error
- Prints the help for any subparsers that exist

**error** (*message*)



```
print_help (**kwargs)
```

## Utility Functions: Logging

```
pytan.utils.change_console_format (debug=False)
```

Changes the logging format for console handler to `pytan.constants.DEBUG_FORMAT` or `pytan.constants.INFO_FORMAT`

**Parameters** `debug` : bool, optional

- False : set logging format for console handler to `pytan.constants.INFO_FORMAT`
- True : set logging format for console handler to `pytan.constants.DEBUG_FORMAT`

```
pytan.utils.remove_logging_handler (name)
```

Removes a logging handler

**Parameters** `name` : str

name of logging handler to remove. if name == 'all' then all logging handlers are removed

```
pytan.utils.set_all_loglevels (level='DEBUG')
```

Sets all loggers that the logging system knows about to a given logger level

```
pytan.utils.set_log_levels (loglevel=0)
```

Enables loggers based on loglevel and `pytan.constants.LOG_LEVEL_MAPS`

**Parameters** `loglevel` : int, optional

loglevel to match against each item in `pytan.constants.LOG_LEVEL_MAPS` - each item that is greater than or equal to loglevel will have the according loggers set to their respective levels identified there-in.

```
pytan.utils.setup_console_logging ()
```

Creates a console logging handler using `SplitStreamHandler`

## Utility Functions: Type Checking

```
pytan.utils.is_dict (l)
```

returns True if `l` is a dictionary, False if not

```
pytan.utils.is_list (l)
```

returns True if `l` is a list, False if not

```
pytan.utils.is_num (l)
```

returns True if `l` is a number, False if not

```
pytan.utils.is_str (l)
```

returns True if `l` is a string, False if not

## Utility Functions: Misc

```
pytan.utils.get_dict_list_items (d, i)
```

Gets keys from dict `d` if any item in list `i` is in the list value for each key

**Parameters** `d` : dict of str

dict to get str from if list contains any item from `i`

**i** : list of str

list of strs to check if for existence in any lists in *d*

**Returns** **list** : list of str

list of strings from *d* that have *i* in their values

`pytan.utils.get_dict_list_len(d, keys=[], negate=False)`

Gets the sum of each list in dict *d*

**Parameters** **d** : dict of str

dict to sums of

**keys** : list of str

list of keys to get sums of, if empty gets a sum of all keys

**negate** : bool

- only used if keys supplied
- False : get the sums of *d* that do match keys
- True : get the sums of *d* that do not match keys

**Returns** **list\_len** : int

sum of lists in *d* that match keys

`pytan.utils.get_now()`

Get current time in human friendly format

**Returns** **str** :

str of current time return from `human_time()`

`pytan.utils.human_time(t, tformat='%Y_%m_%d-%H_%M_%S-%Z')`

Get time in human friendly format

**Parameters** **t** : int, float, time

either a unix epoch or struct\_time object to convert to string

**tformat** : str, optional

format of string to convert time to

**Returns** **str** :

*t* converted to str

`pytan.utils.jsonify(v, indent=2, sort_keys=True)`

Turns python object *v* into a pretty printed JSON string

**Parameters** **v** : object

python object to convert to JSON

**indent** : int, 2

number of spaces to indent JSON string when pretty printing

**sort\_keys** : bool, True

sort keys of JSON string when pretty printing

**Returns** **str** :

JSON pretty printed string

`pytan.utils.port_check(address, port, timeout=5)`  
 Check if *address:port* can be reached within *timeout*

**Parameters** *address* : str

hostname/ip address to check *port* on

**port** : int

port to check on *address*

**timeout** : int, optional

timeout after N seconds of not being able to connect

**Returns** `socket` or `False` :

if connection succeeds, the socket object is returned, else `False` is returned

`pytan.utils.seconds_from_now(secs=0, tz='utc')`  
 Get time in Tanium SOAP API format *secs* from now

**Parameters** *secs* : int

seconds from now to get time str

**tz** : str, optional

time zone to return string in, default is 'utc' - supplying anything else will supply local time

**Returns** str :

time *secs* from now in Tanium SOAP API format

`pytan.utils.test_app_port(host, port)`  
 Validates that *host:port* can be reached using `port_check()`

**Parameters** *host* : str

hostname/ip address to check *port* on

**port** : int

port to check on *host*

**Raises** `HandlerError` : `pytan.utils.HandlerError`

if *host:port* can not be reached

`pytan.utils.version_check(reqver)`  
 Allows scripts using `pytan` to validate the version of the script against the version of `pytan`

**Parameters** *reqver* : str

string containing version number to check against `Exception`

**Raises** `Exception` : `Exception`

if `pytan.__version__` is not greater or equal to *reqver*

`pytan.utils.xml_pretty(x)`  
 Uses `xmltodict` to pretty print an XML str *x*

**Parameters** *x* : str

XML string to pretty print

**Returns** str :

The pretty printed string of *x*

`pytan.utils.xml_pretty_resultobj(x)`

Uses `xmldict` to pretty print an the result-object element in XML str *x*

**Parameters** x : str

XML string to pretty print

**Returns** str :

The pretty printed string of result-object in *x*

`pytan.utils.xml_pretty_resultxml(x)`

Uses `xmldict` to pretty print an the ResultXML element in XML str *x*

**Parameters** x : str

XML string to pretty print

**Returns** str :

The pretty printed string of ResultXML in *x*

## Utility Functions: Argument Parsers for Command Line Scripts

`pytan.utils.setup_parser(desc, help=False)`

Method to setup the base `pytan.utils.CustomArgParse` class for command line scripts that use `pytan`. This establishes the basic arguments that are needed by all such scripts, such as:

- `--help`
- `--username`
- `--password`
- `--host`
- `--port`
- `--loglevel`
- `--debugformat` (not shown in `--help`)

`pytan.utils.setup_get_object_argparser(obj, doc)`

Method to setup the base `pytan.utils.CustomArgParse` class for command line scripts using `pytan.utils.setup_parser()`, then add specific arguments for scripts that use `pytan` to get objects.

`pytan.utils.setup_create_json_object_argparser(obj, doc)`

Method to setup the base `pytan.utils.CustomArgParse` class for command line scripts using `pytan.utils.setup_parser()`, then add specific arguments for scripts that use `pytan` to create objects from json files.

`pytan.utils.setup_delete_object_argparser(obj, doc)`

Method to setup the base `pytan.utils.CustomArgParse` class for command line scripts using `pytan.utils.setup_parser()`, then add specific arguments for scripts that use `pytan` to delete objects.

`pytan.utils.setup_ask_saved_argparser(doc)`

Method to setup the base `pytan.utils.CustomArgParse` class for command line scripts using `pytan.utils.setup_parser()`, then add specific arguments for scripts that use `pytan` to ask saved questions.

`pytan.utils.setup_stop_action_argparser` (*doc*)

Method to setup the base `pytan.utils.CustomArgParse` class for command line scripts using `pytan.utils.setup_parser()`, then add specific arguments for scripts that use `pytan` to stop actions.

`pytan.utils.setup_deploy_action_argparser` (*doc*)

Method to setup the base `pytan.utils.CustomArgParse` class for command line scripts using `pytan.utils.setup_parser()`, then add specific arguments for scripts that use `pytan` to deploy actions.

`pytan.utils.setup_get_result_argparser` (*doc*)

Method to setup the base `pytan.utils.CustomArgParse` class for command line scripts using `pytan.utils.setup_parser()`, then add specific arguments for scripts that use `pytan` to get results for questions or actions.

`pytan.utils.setup_ask_manual_argparser` (*doc*)

Method to setup the base `pytan.utils.CustomArgParse` class for command line scripts using `pytan.utils.setup_parser()`, then add specific arguments for scripts that use `pytan` to ask manual questions.

`pytan.utils.add_ask_report_argparser` (*parser*)

Method to extend a `pytan.utils.CustomArgParse` class for command line scripts with arguments for scripts that need to supply export format subparsers for asking questions.

`pytan.utils.add_report_file_options` (*parser*)

Method to extend a `pytan.utils.CustomArgParse` class for command line scripts with arguments for scripts that need to supply export file and directory options.

`pytan.utils.add_get_object_report_argparser` (*parser*)

Method to extend a `pytan.utils.CustomArgParse` class for command line scripts with arguments for scripts that need to supply export format subparsers for getting objects.

`pytan.utils.get_grp_opts` (*parser, grp\_names*)

Used to get arguments in *parser* that match argument group names in *grp\_names*

**Parameters** *parser* : `argparse.ArgumentParser`

ArgParse object

**grp\_names** : list of str

list of str of argument group names to get arguments for

**Returns** *grp\_opts* : list of str

list of arguments gathered from argument group names in *grp\_names*

`pytan.utils.process_create_json_object_args` (*parser, handler, obj, all\_args*)

Process command line args supplied by user for create json object

**Parameters** *parser* : `argparse.ArgumentParser`

ArgParse object used to parse *all\_args*

**handler** : `pytan.handler.Handler`

Instance of Handler created from command line args

**obj** : str

Object type for create json object

**all\_args** : dict

dict of args parsed from *parser*

**Returns** `response`: `taniumpy.object_types.base.BaseType`

response from `pytan.handler.Handler.create_from_json()`

`pytan.utils.process_delete_object_args` (*parser, handler, obj, all\_args*)

Process command line args supplied by user for delete object

**Parameters** `parser`: `argparse.ArgumentParser`

ArgParse object used to parse *all\_args*

`handler`: `pytan.handler.Handler`

Instance of Handler created from command line args

`obj`: str

Object type for delete object

`all_args`: dict

dict of args parsed from *parser*

**Returns** `response`: `taniumpy.object_types.base.BaseType`

response from `pytan.handler.Handler.delete()`

`pytan.utils.process_get_object_args` (*parser, handler, obj, all\_args*)

Process command line args supplied by user for get object

**Parameters** `parser`: `argparse.ArgumentParser`

ArgParse object used to parse *all\_args*

`handler`: `pytan.handler.Handler`

Instance of Handler created from command line args

`obj`: str

Object type for get object

`all_args`: dict

dict of args parsed from *parser*

**Returns** `response`: `taniumpy.object_types.base.BaseType`

response from `pytan.handler.Handler.get()`

## Utility Functions: Dehumanize human strings

`pytan.utils.dehumanize_package` (*package*)

Turns a package str into a package definition

**Parameters** `package`: str

A str that describes a package and optionally a selector and/or parameters

**Returns** `package_def`: dict

dict parsed from *sensors*

`pytan.utils.dehumanize_question_filters` (*question\_filters*)

Turns a question\_filters str or list of str into a question filter definition

**Parameters** `question_filters`: str, list of str

A str or list of str that describes a sensor for a question filter(s) and optionally a selector and/or filter

**Returns** `question_filter_defs` : list of dict

list of dict parsed from *question\_filters*

`pytan.utils.dehumanize_question_options(question_options)`

Turns a question\_options str or list of str into a question option definition

**Parameters** `question_options` : str, list of str

A str or list of str that describes question options

**Returns** `question_option_defs` : list of dict

list of dict parsed from *question\_options*

`pytan.utils.dehumanize_sensors(sensors, key='sensors', empty_ok=False)`

Turns a sensors str or list of str into a sensor definition

**Parameters** `sensors` : str, list of str

A str or list of str that describes a sensor(s) and optionally a selector, parameters, filter, and/or options

**key** : str, optional

Name of key that user should have provided *sensors* as

**empty\_ok** : bool, optional

False: *sensors* is not allowed to be empty, throw `HumanParserError` if it is empty

True: *sensors* is allowed to be empty

**Returns** `sensor_defs` : list of dict

list of dict parsed from *sensors*

`pytan.utils.extract_filter(s)`

Extracts a filter from str *s*

**Parameters** `s` : str

A str that may or may not have a filter identified by ‘, that HUMAN VALUE’

**Returns** `s` : str

str *s* without the parsed\_filter included

**parsed\_filter** : dict

filter attributes mapped from filter from *s* if any found

`pytan.utils.extract_options(s)`

Extracts options from str *s*

**Parameters** `s` : str

A str that may or may not have options identified by ‘, opt:name[:value]’

**Returns** `s` : str

str *s* without the parsed\_options included

**parsed\_options** : list

options extracted from *s* if any found

`pytan.utils.extract_params(s)`

Extracts parameters from str *s*

**Parameters** *s* : str

A str that may or may not have parameters identified by {key=value}

**Returns** *s* : str

str *s* without the parsed\_params included

**parsed\_params** : list

parameters extracted from *s* if any found

`pytan.utils.extract_selector(s)`

Extracts a selector from str *s*

**Parameters** *s* : str

A str that may or may not have a selector in the beginning in the form of id:, name:, or :hash – if no selector found, name will be assumed as the default selector

**Returns** *s* : str

str *s* without the parsed\_selector included

**parsed\_selector** : str

selector extracted from *s*, or 'name' if none found

`pytan.utils.map_filter(filter_str)`

Maps a filter str against `constants.FILTER_MAPS`

**Parameters** *filter\_str* : str

*filter\_str* str that should be validated

**Returns** *filter\_attrs* : dict

dict containing mapped filter attributes for SOAP API

`pytan.utils.map_option(opt, dest)`

Maps an opt str against `constants.OPTION_MAPS`

**Parameters** *opt* : str

option str that should be validated

**dest** : list of str

list of valid destinations (i.e. *filter* or *group*)

**Returns** *opt\_attrs* : dict

dict containing mapped option attributes for SOAP API

`pytan.utils.map_options(options, dest)`

Maps a list of options using `map_option()`

**Parameters** *options* : list of str

list of str that should be validated

**dest** : list of str

list of valid destinations (i.e. *filter* or *group*)

**Returns** *mapped\_options* : dict



dict of all mapped\_options

### Utility Functions: kwargs getters

`pytan.utils.get_ask_kwargs(**kwargs)`

Gets QuestionAsker args from kwargs and returns a dict with just those matching args

**Parameters** `**kwargs` : dict

kwargs to get keys from

**Returns** `ask_kwargs` : dict

args from kwargs that are found in `pytan.constants.ASK_KWARGS`

`pytan.utils.get_kwargs_int(key, default=None, **kwargs)`

Gets key from kwargs and validates it is an int

**Parameters** `key` : str

key to get from kwargs

**default** : int, optional

default value to use if key not found in kwargs

**\*\*kwargs** : dict

kwargs to get key from

**Returns** `val` : int

value from key, or default if supplied

`pytan.utils.get_req_kwargs(**kwargs)`

Gets SOAP API request args from kwargs and returns a dict with just those matching args

**Parameters** `**kwargs` : dict

kwargs to get keys from

**Returns** `req_kwargs` : dict

args from kwargs that are found in `pytan.constants.REQ_KWARGS`

### Utility Functions: Object mappers

`pytan.utils.get_obj_map(objtype)`

Gets an object map for *objtype*

**Parameters** `objtype` : str

object type to get object map from in `pytan.constants.GET_OBJ_MAP`

**Returns** `obj_map` : dict

matching object map for *objtype* from `pytan.constants.GET_OBJ_MAP`

`pytan.utils.get_q_obj_map(qtype)`

Gets an object map for *qtype*

**Parameters** `qtype` : str

question type to get object map from in `pytan.constants.Q_OBJ_MAP`

**Returns** `obj_map` : dict

matching object map for *qtype* from `pytan.constants.Q_OBJ_MAP`

## Utility Functions: Taniumpy objects

`pytan.utils.apply_options_obj(options, obj, dest)`

Updates an object with options

**Parameters** `options` : dict

dict containing options definition

`obj` : `taniumpy.object_types.base.BaseType`

Taniumpy object to apply *options* to

`dest` : list of str

list of valid destinations (i.e. *filter* or *group*)

**Returns** `obj` : `taniumpy.object_types.base.BaseType`

Taniumpy object updated with attributes from *options*

`pytan.utils.build_group_obj(q_filter_defs, q_option_defs)`

Creates a Group object from *q\_filter\_defs* and *q\_option\_defs*

**Parameters** `q_filter_defs` : list of dict

List of dict that are question filter definitions

`q_option_defs` : dict

dict of question filter options

**Returns** `group_obj` : `taniumpy.object_types.group.Group`

Group object with list of `taniumpy.object_types.filter.Filter` built from *q\_filter\_defs* and *q\_option\_defs*

`pytan.utils.build_manual_q(selectlist_obj, group_obj)`

Creates a Question object from *selectlist\_obj* and *group\_obj*

**Parameters** `selectlist_obj` : `taniumpy.object_types.select_list.SelectList`

SelectList object to add to Question object

`group_obj` : `taniumpy.object_types.group.Group`

Group object to add to Question object

**Returns** `add_q_obj` : `taniumpy.object_types.question.Question`

Question object built from *selectlist\_obj* and *group\_obj*

`pytan.utils.build_metadatalist_obj(properties, nameprefix='')`

Creates a MetadataList object from *properties*

**Parameters** `properties` : list of list of str

list of lists, each list having two str - str 1: property key, str2: property value

`nameprefix` : str

prefix to insert in front of property key when creating MetadataItem

**Returns** `metadatalist_obj` : `taniumpy.object_types.metadata_list.MetadataList`

MetadataList object with list of `taniumpy.object_types.metadata_item.MetadataItem` built from *properties*

`pytan.utils.build_param_obj (key, val, delim='')`

Creates a Parameter object from key and value, surrounding key with delim

**Parameters** `key` : str

key to use for parameter

**value** : str

value to use for parameter

**delim** : str

str to surround key with when adding to parameter object

**Returns** `param_obj` : `taniumpy.object_types.parameter.Parameter`

Parameter object built from key and val

`pytan.utils.build_param_objlist (obj, user_params, delim='', derive_def=False, empty_ok=False)`

Creates a ParameterList object from user\_params

**Parameters** `obj` : `taniumpy.object_types.base.BaseType`

TaniumPy object to verify parameters against

**user\_params** : dict

dict describing key and value of user supplied params

**delim** : str

str to surround key with when adding to parameter object

**derive\_def** : bool, optional

- False: Do not derive default values, and throw a `HandlerError` if user did not supply a value for a given parameter
- True: Try to derive a default value for each parameter if user did not supply one

**empty\_ok** : bool, optional

- False: If user did not supply a value for a given parameter, throw a `HandlerError`
- True: If user did not supply a value for a given parameter, do not add the parameter to the ParameterList object

**Returns** `param_objlist` : `taniumpy.object_types.parameter_list.ParameterList`

ParameterList object with list of `taniumpy.object_types.parameter.Parameter` built from user\_params

`pytan.utils.build_selectlist_obj (sensor_defs)`

Creates a SelectList object from sensor\_defs

**Parameters** `sensor_defs` : list of dict

List of dict that are sensor definitions

**Returns** `select_objlist` : `taniumpy.object_types.select_list.SelectList`

SelectList object with list of `taniumpy.object_types.select.Select` built from *sensor\_defs*

`pytan.utils.derive_param_default(obj_param)`

Derive a parameter default

**Parameters** `obj_param` : dict

parameter dict from Taniumpy object

**Returns** `def_val` : str

default value derived from `obj_param`

`pytan.utils.empty_obj(taniumpy_object)`

Validate that a given Taniumpy object is not empty

**Parameters** `taniumpy_object` : `taniumpy.object_types.base.BaseType`

object to check if empty

**Returns** bool

True if `taniumpy_object` is considered empty, False otherwise

`pytan.utils.get_filter_obj(sensor_def)`

Creates a Filter object from `sensor_def`

**Parameters** `sensor_def` : dict

dict containing sensor definition

**Returns** `filter_obj` : `taniumpy.object_types.filter.Filter`

Filter object created from `sensor_def`

`pytan.utils.get_obj_params(obj)`

Get the parameters from a Taniumpy object and JSON load them

**obj** [`taniumpy.object_types.base.BaseType`] Taniumpy object to get parameters from

**Returns** `params` : dict

JSON loaded dict of parameters from `obj`

`pytan.utils.question_progress(asker, pct)`

Call back method for `taniumpy.question_asker.QuestionAsker.run()` to report progress while waiting for results from a question

**Parameters** `asker` : `taniumpy.question_asker.QuestionAsker`

QuestionAsker instance

**pct** : float

Percentage completion of question

## Utility Functions: Definition objects

`pytan.utils.check_dictkey(d, key, valid_types, valid_list_types)`

Yet another method to check a dictionary for a key

**Parameters** `d` : dict

dictionary to check for key

**key** : str

key to check for in `d`

**valid\_types** : list of str

list of str of valid types for key

**valid\_list\_types** : list of str

if key is a list, validate that all values of list are in valid\_list\_types

`pytan.utils.chk_def_key(def_dict, key, keytypes, keysubtypes=None, req=False)`

Checks that def\_dict has key

**Parameters** **def\_dict** : dict

Definition dictionary

**key** : str

key to check for in def\_dict

**keytypes** : list of str

list of str of valid types for key

**keysubtypes** : list of str

if key is a dict or list, validate that all values of dict or list are in keysubtypes

**req** : bool

- False: key does not have to be in def\_dict
- True: key must be in def\_dict, throw `DefinitionParserError` if not

`pytan.utils.parse_defs(defname, deftypes, strconv=None, empty_ok=True, defs=None, **kwargs)`

Parses and validates defs into new\_defs

**Parameters** **defname** : str

Name of definition

**deftypes** : list of str

list of valid types that defs can be

**strconv** : str

if supplied, and defs is a str, turn defs into a dict with key = strconv, value = defs

**empty\_ok** : bool

- True: defs is allowed to be empty
- False: defs is not allowed to be empty

**Returns** **new\_defs** : list of dict

parsed and validated defs

`pytan.utils.val_package_def(package_def)`

Validates package definitions

Ensures package definition has a selector, and if a package definition has a params key, that key is valid

**Parameters** **package\_def** : dict

package definition

`pytan.utils.val_q_filter_defs(q_filter_defs)`

Validates question filter definitions

Ensures each question filter definition has a selector, and if a question filter definition has a filter key, that key is valid

**Parameters** `q_filter_defs` : list of dict

list of question filter definitions

`pytan.utils.val_sensor_defs(sensor_defs)`

Validates sensor definitions

Ensures each sensor definition has a selector, and if a sensor definition has a params, options, or filter key, that each key is valid

**Parameters** `sensor_defs` : list of dict

list of sensor definitions

## 1.2.5 pytan Unit Tests

This contains unit tests for pytan.

These unit tests do not require a connection to a Tanium server in order to run.

```
class test_pytan_unit.TestDehumanizeExtractionUtils (methodName='runTest')
```

```
    Bases: unittest.case.TestCase
```

```
    __module__ = 'test_pytan_unit'
```

```
    test_extract_filter_invalid()
```

```
    test_extract_filter_nofilter()
```

```
    test_extract_filter_valid()
```

```
    test_extract_filter_valid_all()
```

```
    test_extract_options_invalid_option()
```

```
    test_extract_options_many()
```

```
    test_extract_options_missing_value_max_data_age()
```

```
    test_extract_options_missing_value_value_type()
```

```
    test_extract_options_nooptions()
```

```
    test_extract_options_single()
```

```
    test_extract_params()
```

```
    test_extract_params_missing_seperator()
```

```
    test_extract_params_multiparams()
```

```
    test_extract_params_noparams()
```

```
    test_extract_selector()
```

```
    test_extract_selector_use_name_if_noselector()
```

```
class test_pytan_unit.TestDehumanizeQuestionFilterUtils (methodName='runTest')
```

```
    Bases: unittest.case.TestCase
```

```
    __module__ = 'test_pytan_unit'
```

```

    test_empty_filterlist()
    test_empty_filterstr()
    test_invalid_filter1()
    test_invalid_filter2()
    test_invalid_filter3()
    test_multi_filter_list()
    test_single_filter_list()
    test_single_filter_str()

class test_pytan_unit.TestDehumanizeQuestionOptionUtils (methodName='runTest')
    Bases: unittest.case.TestCase
    __module__ = 'test_pytan_unit'
    test_empty_optionlist()
    test_empty_optionstr()
    test_invalid_option1()
    test_invalid_option2()
    test_option_list_many()
    test_option_list_multi()
    test_option_list_single()
    test_option_str()

class test_pytan_unit.TestDehumanizeSensorUtils (methodName='runTest')
    Bases: unittest.case.TestCase
    __module__ = 'test_pytan_unit'
    test_empty_args_dict()
    test_empty_args_list()
    test_empty_args_str()
    test_multi_list_complex()
    test_single_str()
    test_single_str_complex1()
    test_single_str_complex2()
    test_single_str_with_filter()
    test_valid_simple_list()
    test_valid_simple_str_hash_selector()
    test_valid_simple_str_id_selector()
    test_valid_simple_str_name_selector()

class test_pytan_unit.TestDeserializeBadXML (methodName='runTest')
    Bases: unittest.case.TestCase
    __module__ = 'test_pytan_unit'

```

```
test_bad_chars_basetype()
test_bad_chars_resultset()
class test_pytan_unit.TestGenericUtils (methodName='runTest')
    Bases: unittest.case.TestCase
    __module__ = 'test_pytan_unit'
    test_ask_kwargs()
    test_empty_obj()
    test_get_now()
    test_get_obj_map()
    test_get_q_obj_map()
    test_invalid_port()
    test_is_dict()
    test_is_list()
    test_is_not_dict()
    test_is_not_list()
    test_is_not_num()
    test_is_not_str()
    test_is_num()
    test_is_str()
    test_jsonify()
    test_req_kwargs()
    test_version_higher()
    test_version_lower()
class test_pytan_unit.TestManualBuildObjectUtils (methodName='runTest')
    Bases: unittest.case.TestCase
    __module__ = 'test_pytan_unit'
    classmethod setUpClass()
    test_build_group_obj()
    test_build_manual_q()
    test_build_selectlist_obj_invalid_filter()
    test_build_selectlist_obj_missing_value()
    test_build_selectlist_obj_noparamssensorobj_noparams()
        builds a selectlist object using a sensor obj with no params
    test_build_selectlist_obj_noparamssensorobj_withparams()
        builds a selectlist object using a sensor obj with no params, but passing in params (which should be added
        as of 1.0.4)
```



```

    test_build_selectlist_obj_withparamssensorobj_noparams ()
        builds a selectlist object using a sensor obj with 4 params but not supplying any values for any of the
        params

    test_build_selectlist_obj_withparamssensorobj_withparams ()
        builds a selectlist object using a sensor obj with 4 params but supplying a value for only one param

class test_pytan_unit.TestManualPackageDefValidateUtils (methodName='runTest')
    Bases: unittest.case.TestCase

    __module__ = 'test_pytan_unit'

    test_invalid1 ()

    test_invalid2 ()

    test_valid1 ()

    test_valid2 ()

class test_pytan_unit.TestManualQuestionFilterDefParseUtils (methodName='runTest')
    Bases: unittest.case.TestCase

    __module__ = 'test_pytan_unit'

    test_parse_emptydict ()

    test_parse_emptylist ()

    test_parse_emptystr ()

    test_parse_multi_filter ()

    test_parse_noargs ()

    test_parse_none ()

    test_parse_single_filter ()

    test_parse_str ()

class test_pytan_unit.TestManualQuestionFilterDefValidateUtils (methodName='runTest')
    Bases: unittest.case.TestCase

    __module__ = 'test_pytan_unit'

    test_invalid1 ()

    test_valid1 ()

    test_valid2 ()

class test_pytan_unit.TestManualQuestionOptionDefParseUtils (methodName='runTest')
    Bases: unittest.case.TestCase

    __module__ = 'test_pytan_unit'

    test_parse_emptydict ()

    test_parse_emptylist ()

    test_parse_emptystr ()

    test_parse_list ()

    test_parse_noargs ()

    test_parse_none ()

```

```
test_parse_options_dict ()
test_parse_str ()
class test_pytan_unit.TestManualSensorDefParseUtils (methodName='runTest')
    Bases: unittest.case.TestCase
    __module__ = 'test_pytan_unit'
    test_parse_complex ()
        list with many items is parsed into same list
    test_parse_dict_hash ()
        dict with hash is parsed into list of same dict
    test_parse_dict_id ()
        dict with id is parsed into list of same dict
    test_parse_dict_name ()
        dict with name is parsed into list of same dict
    test_parse_emptydict ()
        args=={} throws exception
    test_parse_emptylist ()
        args==[] throws exception
    test_parse_emptystr ()
        args=='' throws exception
    test_parse_noargs ()
        no args throws exception
    test_parse_none ()
        args==None throws exception
    test_parse_str1 ()
        simple str is parsed into list of same str
class test_pytan_unit.TestManualSensorDefValidateUtils (methodName='runTest')
    Bases: unittest.case.TestCase
    __module__ = 'test_pytan_unit'
    test_invalid1 ()
    test_invalid2 ()
    test_invalid3 ()
    test_invalid4 ()
    test_valid1 ()
    test_valid2 ()
    test_valid3 ()
    test_valid4 ()
```

## 1.2.6 pytan Functional Tests

This contains functional tests for pytan.

These functional tests require a connection to a Tanium server in order to run. The connection info is pulled from the SERVER\_INFO dictionary in test/API\_INFO.py.

These tests all use `ddt`, a package that provides for data driven tests via JSON files.

```
class test_pytan_func.InvalidServerTests (methodName='runTest')
    Bases: unittest.case.TestCase

    __module__ = 'test_pytan_func'

    classmethod setUpClass ()

    test_invalid_connect_1_bad_username ()

    test_invalid_connect_2_bad_host_and_non_ssl_port ()

    test_invalid_connect_3_bad_password ()

    test_invalid_connect_4_bad_host_and_bad_port ()

class test_pytan_func.ValidServerTests (methodName='runTest')
    Bases: unittest.case.TestCase

    __module__ = 'test_pytan_func'

    classmethod setUpClass ()

    setup_test ()

    test_invalid_create_object_1_invalid_create_sensor ()

    test_invalid_create_object_from_json_1_invalid_create_saved_action_from_json ()

    test_invalid_create_object_from_json_2_invalid_create_client_from_json ()

    test_invalid_create_object_from_json_3_invalid_create_userrole_from_json ()

    test_invalid_create_object_from_json_4_invalid_create_setting_from_json ()

    test_invalid_deploy_action_1_invalid_deploy_action_run_false ()

    test_invalid_deploy_action_2_invalid_deploy_action_package_help ()

    test_invalid_deploy_action_3_invalid_deploy_action_package ()

    test_invalid_deploy_action_4_invalid_deploy_action_options_help ()

    test_invalid_deploy_action_5_invalid_deploy_action_empty_package ()

    test_invalid_deploy_action_6_invalid_deploy_action_filters_help ()

    test_invalid_deploy_action_7_invalid_deploy_action_missing_parameters ()

    test_invalid_export_basetype_1_invalid_export_basetype_csv_bad_explode_type ()

    test_invalid_export_basetype_2_invalid_export_basetype_csv_bad_sort_sub_type ()

    test_invalid_export_basetype_3_invalid_export_basetype_csv_bad_sort_type ()

    test_invalid_export_basetype_4_invalid_export_basetype_xml_bad_minimal_type ()

    test_invalid_export_basetype_5_invalid_export_basetype_json_bad_include_type ()

    test_invalid_export_basetype_6_invalid_export_basetype_json_bad_explode_type ()

    test_invalid_export_basetype_7_invalid_export_basetype_bad_format ()

    test_invalid_export_resultset_1_invalid_export_resultset_csv_bad_sort_sub_type ()

    test_invalid_export_resultset_2_invalid_export_resultset_csv_bad_sort_type ()
```

```
test_invalid_export_resultset_3_invalid_export_resultset_csv_bad_expand_type()
test_invalid_export_resultset_4_invalid_export_resultset_csv_bad_sensors_sub_type()
test_invalid_export_resultset_5_invalid_export_resultset_bad_format()
test_invalid_get_object_1_invalid_get_action_single_by_name()
test_invalid_get_object_2_invalid_get_question_by_name()
test_invalid_question_1_invalid_ask_manual_human_question_paramater_too_many()
test_invalid_question_2_invalid_ask_manual_human_question_filter_help()
test_invalid_question_3_invalid_ask_manual_human_question_option()
test_invalid_question_4_invalid_ask_manual_human_question_filter()
test_invalid_question_5_invalid_ask_manual_human_question_parameter_split()
test_invalid_question_6_invalid_ask_manual_human_question_option_help()
test_invalid_question_7_invalid_ask_manual_question_sensor()
test_invalid_question_8_invalid_ask_manual_human_question_sensor_help()
test_valid_create_object_1_create_user()
test_valid_create_object_2_create_package()
test_valid_create_object_3_create_group()
test_valid_create_object_4_create_whitelisted_url()
test_valid_create_object_from_json_1_create_package_from_json()
test_valid_create_object_from_json_2_create_user_from_json()
test_valid_create_object_from_json_3_create_saved_question_from_json()
test_valid_create_object_from_json_4_create_action_from_json()
test_valid_create_object_from_json_5_create_sensor_from_json()
test_valid_create_object_from_json_6_create_question_from_json()
test_valid_create_object_from_json_7_create_whitelisted_url_from_json()
test_valid_create_object_from_json_8_create_group_from_json()
test_valid_deploy_action_1_deploy_action_simple_against_windows_computers()
test_valid_deploy_action_2_deploy_action_simple_without_results()
test_valid_deploy_action_3_deploy_action_with_params_against_windows_computers()
test_valid_deploy_action_4_deploy_action_simple()
test_valid_export_basetype_10_export_basetype_xml_default_options()
test_valid_export_basetype_11_export_basetype_csv_with_explode_true()
test_valid_export_basetype_12_export_basetype_json_explode_false()
test_valid_export_basetype_13_export_basetype_json_type_false()
test_valid_export_basetype_14_export_basetype_json_default_options()
test_valid_export_basetype_1_export_basetype_csv_with_sort_list()
test_valid_export_basetype_2_export_basetype_csv_with_explode_false()
```

```
test_valid_export_basetype_3_export_basetype_json_type_true()
test_valid_export_basetype_4_export_basetype_xml_minimal_false()
test_valid_export_basetype_5_export_basetype_xml_minimal_true()
test_valid_export_basetype_6_export_basetype_csv_with_sort_empty_list()
test_valid_export_basetype_7_export_basetype_csv_default_options()
test_valid_export_basetype_8_export_basetype_json_explode_true()
test_valid_export_basetype_9_export_basetype_csv_with_sort_true()
test_valid_export_resultset_10_export_resultset_csv_default_options()
test_valid_export_resultset_11_export_resultset_csv_type_true()
test_valid_export_resultset_12_export_resultset_csv_all_options()
test_valid_export_resultset_13_export_resultset_csv_sort_false()
test_valid_export_resultset_1_export_resultset_json()
test_valid_export_resultset_2_export_resultset_csv_sensor_true()
test_valid_export_resultset_3_export_resultset_csv_type_false()
test_valid_export_resultset_4_export_resultset_csv_expand_false()
test_valid_export_resultset_5_export_resultset_csv_sort_empty()
test_valid_export_resultset_6_export_resultset_csv_sort_true()
test_valid_export_resultset_7_export_resultset_csv_sort_list()
test_valid_export_resultset_8_export_resultset_csv_sensor_false()
test_valid_export_resultset_9_export_resultset_csv_expand_true()
test_valid_get_object_10_get_all_saved_questions()
test_valid_get_object_11_get_user_by_name()
test_valid_get_object_12_get_all_userroless()
test_valid_get_object_13_get_all_questions()
test_valid_get_object_14_get_sensor_by_id()
test_valid_get_object_15_get_all_groups()
test_valid_get_object_16_get_all_sensors()
test_valid_get_object_17_get_sensor_by_mixed()
test_valid_get_object_18_get_whitelisted_url_by_id()
test_valid_get_object_19_get_group_by_name()
test_valid_get_object_1_get_all_users()
test_valid_get_object_20_get_all_whitelisted_urls()
test_valid_get_object_21_get_sensor_by_hash()
test_valid_get_object_22_get_package_by_name()
test_valid_get_object_23_get_all_clients()
test_valid_get_object_24_get_sensor_by_names()
```

```
test_valid_get_object_25_get_all_packages()
test_valid_get_object_26_get_saved_question_by_name()
test_valid_get_object_27_get_all_actions()
test_valid_get_object_28_get_user_by_id()
test_valid_get_object_29_get_sensor_by_name()
test_valid_get_object_2_get_action_by_id()
test_valid_get_object_30_get_saved_action_by_name()
test_valid_get_object_3_get_question_by_id()
test_valid_get_object_4_get_saved_question_by_names()
test_valid_get_object_5_get_userrole_by_id()
test_valid_get_object_6_get_all_saved_actions()
test_valid_get_object_7_get_leader_clients()
test_valid_get_object_8_get_all_settings()
test_valid_get_object_9_get_setting_by_name()
test_valid_question_10_ask_manual_human_question_sensor_with_parameters_and_filter_and
test_valid_question_11_ask_manual_human_question_sensor_with_filter_and_2_options()
test_valid_question_12_ask_manual_human_question_sensor_with_filter()
test_valid_question_13_ask_manual_human_question_simple_multiple_sensors()
test_valid_question_14_ask_manual_human_question_multiple_sensors_identified_by_name()
test_valid_question_15_ask_manual_human_question_sensor_with_parameters_and_filter()
test_valid_question_16_ask_saved_question_by_name()
test_valid_question_17_ask_manual_human_question_sensor_with_parameters_and_no_supplie
test_valid_question_1_ask_manual_human_question_sensor_with_parameters_and_some_supplie
test_valid_question_2_ask_manual_human_question_simple_single_sensor()
test_valid_question_3_ask_manual_human_question_sensor_with_filter_and_3_options()
test_valid_question_4_ask_manual_human_question_sensor_without_parameters_and_supplied
test_valid_question_5_ask_manual_human_question_complex_query2()
test_valid_question_6_ask_manual_human_question_complex_query1()
test_valid_question_7_ask_saved_question_by_name_in_list()
test_valid_question_8_ask_manual_human_question_multiple_sensors_with_parameters_and_s
test_valid_question_9_ask_manual_question_sensor_complex()

test_pytan_func.spew(m)
```

## 1.3 taniumpy package

A python package that handles the serialization/deserialization of XML SOAP requests/responses from Tanium to/from python objects.

### 1.3.1 taniumpy.session module

Session handler for Tanium API

**exception** `taniumpy.session.AuthorizationError`

Bases: `exceptions.Exception`

**exception** `taniumpy.session.BadResponseError`

Bases: `exceptions.Exception`

**class** `taniumpy.session.DynamicFormatter`

Bases: `string.Formatter`

**get\_value** (*key, args, kwargs*)

**exception** `taniumpy.session.HttpError`

Bases: `exceptions.Exception`

**class** `taniumpy.session.NoLogging`

Bases: `object`

**count** = 0

Disable logging while executing code block

**class** `taniumpy.session.Session` (*server, port=443*)

Bases: `object`

**ADD\_OBJECT** = 'AddObject'

**AUTH\_RES** = '/auth'

**DELETE\_OBJECT** = 'DeleteObject'

**FORMATTER** (*format\_string, \*args, \*\*kwargs*)

**GET\_OBJECT** = 'GetObject'

**GET\_RESULT\_DATA** = 'GetResultData'

**GET\_RESULT\_INFO** = 'GetResultInfo'

**INFO\_RES** = '/info.json'

**REQUEST\_BODY** = u'<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsd=

**SOAP\_PORT** = 444

**SOAP\_RES** = '/soap'

**UPDATE\_OBJECT** = 'UpdateObject'

**add** (*obj, \*\*kwargs*)

**authenticate** (*username=None, password=None, get\_version=True*)

**delete** (*obj, \*\*kwargs*)

**find** (*object\_type, \*\*kwargs*)

**getResultData** (*obj, \*\*kwargs*)

**getResultInfo** (*obj, \*\*kwargs*)

**get\_server\_info** ()

**is\_auth**

**save** (*obj, \*\*kwargs*)

**server\_version**

**session\_id**

`taniumpy.session.http_post (host, port, url, body=None, headers=None, timeout=5)`

`taniumpy.session.load_file (filename)`

`taniumpy.session.nologging (func)`  
decorator to disable logging on a function

`taniumpy.session.xml_fix (s)`  
# 1.0.4: added this function

this supports better handling of invalid XML, removing invalid control characters and re-encoding to utf-8 with `xmlcharrefreplace`

## 1.3.2 `taniumpy.question_asker` module

**class** `taniumpy.question_asker.QuestionAsker (session, question, polling_interval=None, pct_complete_threshold=99, timeout=300)`

Bases: `object`

A class to aid in asking a Question.

The primary function of this class is to poll for result info for question, and fire off events:

ProgressChanged AnswersChanged AnswersComplete

**POLLING\_INTERVAL = 5**

**run** (`callbacks={}, **kwargs`)

Poll for question data and issue callbacks.

Callbacks should be a dict with members: 'ProgressChanged' 'AnswersChanged' 'AnswersComplete'

Each should be a function that accepts a QuestionAsker and a percent complete.

Any callback can choose to get data from the session by calling `asker.session.getResultData(asker.question)`

Polling will be stopped only when one of the callbacks calls the `stop()` method or the answers are complete.

Note that callbacks can call `setPercentCompleteThreshold` to change what done means on the fly

**setPctCompleteThreshold** (`val`)

**stop** ()

**exception** `taniumpy.question_asker.QuestionTimeoutException`

Bases: `exceptions.Exception`

## 1.3.3 `taniumpy.object_types` package

**`taniumpy.object_types` module**

**`taniumpy.object_types.action` module**

**class** `taniumpy.object_types.action.Action`

Bases: `taniumpy.object_types.base.BaseType`



**taniumpy.object\_types.action\_list module**

```
class taniumpy.object_types.action_list.ActionList
    Bases: taniumpy.object_types.base.BaseType
```

**taniumpy.object\_types.action\_list\_info module**

```
class taniumpy.object_types.action_list_info.ActionListInfo
    Bases: taniumpy.object_types.base.BaseType
```

**taniumpy.object\_types.action\_stop module**

```
class taniumpy.object_types.action_stop.ActionStop
    Bases: taniumpy.object_types.base.BaseType
```

**taniumpy.object\_types.action\_stop\_list module**

```
class taniumpy.object_types.action_stop_list.ActionStopList
    Bases: taniumpy.object_types.base.BaseType
```

**taniumpy.object\_types.all\_objects module****taniumpy.object\_types.archived\_question module**

```
class taniumpy.object_types.archived_question.ArchivedQuestion
    Bases: taniumpy.object_types.base.BaseType
```

**taniumpy.object\_types.archived\_question\_list module**

```
class taniumpy.object_types.archived_question_list.ArchivedQuestionList
    Bases: taniumpy.object_types.base.BaseType
```

**taniumpy.object\_types.audit\_data module**

```
class taniumpy.object_types.audit_data.AuditData
    Bases: taniumpy.object_types.base.BaseType
```

**taniumpy.object\_types.base module**

```
class taniumpy.object_types.base.BaseType (simple_properties,           complex_properties,
                                             list_properties)
    Bases: object
    append (n)
        Allow adding to list.

        Only supported on types that have a single property that is in list_properties
    explode_json (val)
    flatten_jsonable (val, prefix)
```

**classmethod fromSOAPBody** (*body*)

Parse body (text) and produce Python tanium objects.

This method assumes a single result\_object, which may be a list or a single object.

**classmethod fromSOAPElement** (*el*)

**static from\_jsonable** (*jsonable*)

Inverse of to\_jsonable, with explode\_json\_string\_values=False.

This can be used to import objects from serialized JSON. This JSON should come from BaseType.to\_jsonable(explode\_json\_string\_values=False, include\_type=True)

### Examples

```
>>> with open('question_list.json') as fd:
...     questions = json.loads(fd.read())
...     # is a list of serialized questions
...     question_objects = BaseType.from_jsonable(questions)
...     # will return a list of api.Question
```

**toSOAPBody** (*minimal=False*)

**toSOAPElement** (*minimal=False*)

**to\_flat\_dict** (*prefix='', explode\_json\_string\_values=False*)

Convert the object to a dict, flattening any lists or nested types

**to\_flat\_dict\_explode\_json** (*val, prefix=''*)

see if the value is json. If so, flatten it out into a dict

**static to\_json** (*jsonable, \*\*kwargs*)

Convert to a json string.

jsonable can be a single BaseType instance or a list of BaseType

**to\_jsonable** (*explode\_json\_string\_values=False, include\_type=True*)

**static write\_csv** (*fd, val, explode\_json\_string\_values=False, \*\*kwargs*)

Write 'val' to CSV. val can be a BaseType instance or a list of BaseType

This does a two-pass, calling to\_flat\_dict for each object, then finding the union of all headers, then writing out the value of each column for each object sorted by header name

explode\_json\_string\_values attempts to see if any of the str values are parseable by json.loads, and if so treat each property as a column value

fd is a file-like object

**exception** `taniumpy.object_types.base.IncorrectTypeException` (*property, expected, actual*)

Bases: `exceptions.Exception`

Raised when a property is not of the expected type

`taniumpy.object_types.base.xml_fix` (*s*)

# 1.0.4: added this function

this supports better handling of invalid XML, removing invalid control characters and re-encoding to utf-8 with xmlcharrefreplace

**taniumpy.object\_types.cache\_filter module**

```
class taniumpy.object_types.cache_filter.CacheFilter
    Bases: taniumpy.object_types.base.BaseType
```

**taniumpy.object\_types.cache\_filter\_list module**

```
class taniumpy.object_types.cache_filter_list.CacheFilterList
    Bases: taniumpy.object_types.base.BaseType
```

**taniumpy.object\_types.cache\_info module**

```
class taniumpy.object_types.cache_info.CacheInfo
    Bases: taniumpy.object_types.base.BaseType
```

**taniumpy.object\_types.client\_count module**

```
class taniumpy.object_types.client_count.ClientCount
    Bases: taniumpy.object_types.base.BaseType
```

**taniumpy.object\_types.client\_status module**

```
class taniumpy.object_types.client_status.ClientStatus
    Bases: taniumpy.object_types.base.BaseType
```

**taniumpy.object\_types.column module**

```
class taniumpy.object_types.column.Column
    Bases: object
    classmethod fromSOAPElement (el)
```

**taniumpy.object\_types.column\_set module**

```
class taniumpy.object_types.column_set.ColumnSet
    Bases: object
    classmethod fromSOAPElement (el)
```

**taniumpy.object\_types.computer\_group module**

```
class taniumpy.object_types.computer_group.ComputerGroup
    Bases: taniumpy.object_types.base.BaseType
```

**taniumpy.object\_types.computer\_group\_list module**

```
class taniumpy.object_types.computer_group_list.ComputerGroupList
    Bases: taniumpy.object_types.base.BaseType
```

### **taniumpy.object\_types.computer\_group\_spec module**

**class** `taniumpy.object_types.computer_group_spec.ComputerGroupSpec`  
Bases: `taniumpy.object_types.base.BaseType`

### **taniumpy.object\_types.computer\_spec\_list module**

**class** `taniumpy.object_types.computer_spec_list.ComputerSpecList`  
Bases: `taniumpy.object_types.base.BaseType`

### **taniumpy.object\_types.error\_list module**

**class** `taniumpy.object_types.error_list.ErrorList`  
Bases: `taniumpy.object_types.base.BaseType`

### **taniumpy.object\_types.filter module**

**class** `taniumpy.object_types.filter.Filter`  
Bases: `taniumpy.object_types.base.BaseType`

### **taniumpy.object\_types.filter\_list module**

**class** `taniumpy.object_types.filter_list.FilterList`  
Bases: `taniumpy.object_types.base.BaseType`

### **taniumpy.object\_types.group module**

**class** `taniumpy.object_types.group.Group`  
Bases: `taniumpy.object_types.base.BaseType`

### **taniumpy.object\_types.group\_list module**

**class** `taniumpy.object_types.group_list.GroupList`  
Bases: `taniumpy.object_types.base.BaseType`

### **taniumpy.object\_types.metadata\_item module**

**class** `taniumpy.object_types.metadata_item.MetadataItem`  
Bases: `taniumpy.object_types.base.BaseType`

### **taniumpy.object\_types.metadata\_list module**

**class** `taniumpy.object_types.metadata_list.MetadataList`  
Bases: `taniumpy.object_types.base.BaseType`

**taniumpy.object\_types.object\_list module**

```
class taniumpy.object_types.object_list.ObjectList
    Bases: taniumpy.object_types.base.BaseType
```

**taniumpy.object\_types.object\_list\_types module****taniumpy.object\_types.options module**

```
class taniumpy.object_types.options.Options
    Bases: taniumpy.object_types.base.BaseType
```

**taniumpy.object\_types.package\_file module**

```
class taniumpy.object_types.package_file.PackageFile
    Bases: taniumpy.object_types.base.BaseType
```

**taniumpy.object\_types.package\_file\_list module**

```
class taniumpy.object_types.package_file_list.PackageFileList
    Bases: taniumpy.object_types.base.BaseType
```

**taniumpy.object\_types.package\_file\_status module**

```
class taniumpy.object_types.package_file_status.PackageFileStatus
    Bases: taniumpy.object_types.base.BaseType
```

**taniumpy.object\_types.package\_file\_status\_list module**

```
class taniumpy.object_types.package_file_status_list.PackageFileStatusList
    Bases: taniumpy.object_types.base.BaseType
```

**taniumpy.object\_types.package\_file\_template module**

```
class taniumpy.object_types.package_file_template.PackageFileTemplate
    Bases: taniumpy.object_types.base.BaseType
```

**taniumpy.object\_types.package\_file\_template\_list module**

```
class taniumpy.object_types.package_file_template_list.PackageFileTemplateList
    Bases: taniumpy.object_types.base.BaseType
```

**taniumpy.object\_types.package\_spec module**

```
class taniumpy.object_types.package_spec.PackageSpec
    Bases: taniumpy.object_types.base.BaseType
```

### **taniumpy.object\_types.package\_spec\_list module**

**class** `taniumpy.object_types.package_spec_list.PackageSpecList`  
Bases: `taniumpy.object_types.base.BaseType`

### **taniumpy.object\_types.parameter module**

**class** `taniumpy.object_types.parameter.Parameter`  
Bases: `taniumpy.object_types.base.BaseType`

### **taniumpy.object\_types.parameter\_list module**

**class** `taniumpy.object_types.parameter_list.ParameterList`  
Bases: `taniumpy.object_types.base.BaseType`

### **taniumpy.object\_types.parse\_job module**

**class** `taniumpy.object_types.parse_job.ParseJob`  
Bases: `taniumpy.object_types.base.BaseType`

### **taniumpy.object\_types.parse\_job\_list module**

**class** `taniumpy.object_types.parse_job_list.ParseJobList`  
Bases: `taniumpy.object_types.base.BaseType`

### **taniumpy.object\_types.parse\_result module**

**class** `taniumpy.object_types.parse_result.ParseResult`  
Bases: `taniumpy.object_types.base.BaseType`

### **taniumpy.object\_types.parse\_result\_group module**

**class** `taniumpy.object_types.parse_result_group.ParseResultGroup`  
Bases: `taniumpy.object_types.base.BaseType`

### **taniumpy.object\_types.parse\_result\_group\_list module**

**class** `taniumpy.object_types.parse_result_group_list.ParseResultGroupList`  
Bases: `taniumpy.object_types.base.BaseType`

### **taniumpy.object\_types.parse\_result\_list module**

**class** `taniumpy.object_types.parse_result_list.ParseResultList`  
Bases: `taniumpy.object_types.base.BaseType`

**taniumpy.object\_types.plugin module**

```
class taniumpy.object_types.plugin.Plugin  
    Bases: taniumpy.object_types.base.BaseType
```

**taniumpy.object\_types.plugin\_argument module**

```
class taniumpy.object_types.plugin_argument.PluginArgument  
    Bases: taniumpy.object_types.base.BaseType
```

**taniumpy.object\_types.plugin\_argument\_list module**

```
class taniumpy.object_types.plugin_argument_list.PluginArgumentList  
    Bases: taniumpy.object_types.base.BaseType
```

**taniumpy.object\_types.plugin\_command\_list module**

```
class taniumpy.object_types.plugin_command_list.PluginCommandList  
    Bases: taniumpy.object_types.base.BaseType
```

**taniumpy.object\_types.plugin\_list module**

```
class taniumpy.object_types.plugin_list.PluginList  
    Bases: taniumpy.object_types.base.BaseType
```

**taniumpy.object\_types.plugin\_schedule module**

```
class taniumpy.object_types.plugin_schedule.PluginSchedule  
    Bases: taniumpy.object_types.base.BaseType
```

**taniumpy.object\_types.plugin\_schedule\_list module**

```
class taniumpy.object_types.plugin_schedule_list.PluginScheduleList  
    Bases: taniumpy.object_types.base.BaseType
```

**taniumpy.object\_types.plugin\_sql module**

```
class taniumpy.object_types.plugin_sql.PluginSql  
    Bases: taniumpy.object_types.base.BaseType
```

**taniumpy.object\_types.plugin\_sql\_column module**

```
class taniumpy.object_types.plugin_sql_column.PluginSqlColumn  
    Bases: taniumpy.object_types.base.BaseType
```

**taniumpy.object\_types.plugin\_sql\_result module**

```
class taniumpy.object_types.plugin_sql_result.PluginSqlResult
    Bases: taniumpy.object_types.base.BaseType
```

**taniumpy.object\_types.question module**

```
class taniumpy.object_types.question.Question
    Bases: taniumpy.object_types.base.BaseType
```

**taniumpy.object\_types.question\_list module**

```
class taniumpy.object_types.question_list.QuestionList
    Bases: taniumpy.object_types.base.BaseType
```

**taniumpy.object\_types.question\_list\_info module**

```
class taniumpy.object_types.question_list_info.QuestionListInfo
    Bases: taniumpy.object_types.base.BaseType
```

**taniumpy.object\_types.result\_info module**

```
class taniumpy.object_types.result_info.ResultInfo
    Bases: object
```

Wrap the result of GetResultInfo

```
classmethod fromSOAPElement (el)
    Deserialize a ResultInfo from a result_info SOAPElement
    Assumes all properties are integer values (true today)
```

**taniumpy.object\_types.result\_set module**

```
class taniumpy.object_types.result_set.ResultSet
    Bases: object
```

Wrap the result of GetResultData

```
classmethod fromSOAPElement (el)
    Deserialize a ResultInfo from a result_info SOAPElement
    Assumes all properties are integer values (true today)
```

```
static to_json (jsonable, **kwargs)
    Convert to a json string.
    jsonable must be a ResultSet instance
```

```
to_jsonable (**kwargs)
```

```
static write_csv (fd, val, **kwargs)
```



**taniumpy.object\_types.row module**

```
class taniumpy.object_types.row.Row(columns)
    Bases: object

    A row in a result set.

    Values are stored in column order, also accessible by key using []

    classmethod fromSOAPElement(el, columns)
```

**taniumpy.object\_types.saved\_action module**

```
class taniumpy.object_types.saved_action.SavedAction
    Bases: taniumpy.object_types.base.BaseType
```

**taniumpy.object\_types.saved\_action\_approval module**

```
class taniumpy.object_types.saved_action_approval.SavedActionApproval
    Bases: taniumpy.object_types.base.BaseType
```

**taniumpy.object\_types.saved\_action\_list module**

```
class taniumpy.object_types.saved_action_list.SavedActionList
    Bases: taniumpy.object_types.base.BaseType
```

**taniumpy.object\_types.saved\_action\_policy module**

```
class taniumpy.object_types.saved_action_policy.SavedActionPolicy
    Bases: taniumpy.object_types.base.BaseType
```

**taniumpy.object\_types.saved\_action\_row\_id\_list module**

```
class taniumpy.object_types.saved_action_row_id_list.SavedActionRowIdList
    Bases: taniumpy.object_types.base.BaseType
```

**taniumpy.object\_types.saved\_question module**

```
class taniumpy.object_types.saved_question.SavedQuestion
    Bases: taniumpy.object_types.base.BaseType
```

**taniumpy.object\_types.saved\_question\_list module**

```
class taniumpy.object_types.saved_question_list.SavedQuestionList
    Bases: taniumpy.object_types.base.BaseType
```

**taniumpy.object\_types.select module**

```
class taniumpy.object_types.select.Select
    Bases: taniumpy.object_types.base.BaseType
```

### **taniumpy.object\_types.select\_list module**

```
class taniumpy.object_types.select_list.SelectList
    Bases: taniumpy.object_types.base.BaseType
```

### **taniumpy.object\_types.sensor module**

```
class taniumpy.object_types.sensor.Sensor
    Bases: taniumpy.object_types.base.BaseType
```

### **taniumpy.object\_types.sensor\_list module**

```
class taniumpy.object_types.sensor_list.SensorList
    Bases: taniumpy.object_types.base.BaseType
```

### **taniumpy.object\_types.sensor\_query module**

```
class taniumpy.object_types.sensor_query.SensorQuery
    Bases: taniumpy.object_types.base.BaseType
```

### **taniumpy.object\_types.sensor\_query\_list module**

```
class taniumpy.object_types.sensor_query_list.SensorQueryList
    Bases: taniumpy.object_types.base.BaseType
```

### **taniumpy.object\_types.sensor\_string\_hints module**

```
class taniumpy.object_types.sensor_string_hints.SensorStringHints
    Bases: taniumpy.object_types.base.BaseType
```

### **taniumpy.object\_types.sensor\_subcolumn module**

```
class taniumpy.object_types.sensor_subcolumn.SensorSubcolumn
    Bases: taniumpy.object_types.base.BaseType
```

### **taniumpy.object\_types.sensor\_subcolumn\_list module**

```
class taniumpy.object_types.sensor_subcolumn_list.SensorSubcolumnList
    Bases: taniumpy.object_types.base.BaseType
```

### **taniumpy.object\_types.sensor\_types module**

### **taniumpy.object\_types.soap\_error module**

```
class taniumpy.object_types.soap_error.SoapError
    Bases: taniumpy.object_types.base.BaseType
```

**taniumpy.object\_types.system\_setting module**

```
class taniumpy.object_types.system_setting.SystemSetting  
    Bases: taniumpy.object_types.base.BaseType
```

**taniumpy.object\_types.system\_settings\_list module**

```
class taniumpy.object_types.system_settings_list.SystemSettingsList  
    Bases: taniumpy.object_types.base.BaseType
```

**taniumpy.object\_types.system\_status\_aggregate module**

```
class taniumpy.object_types.system_status_aggregate.SystemStatusAggregate  
    Bases: taniumpy.object_types.base.BaseType
```

**taniumpy.object\_types.system\_status\_list module**

```
class taniumpy.object_types.system_status_list.SystemStatusList  
    Bases: taniumpy.object_types.base.BaseType
```

**taniumpy.object\_types.upload\_file module**

```
class taniumpy.object_types.upload_file.UploadFile  
    Bases: taniumpy.object_types.base.BaseType
```

**taniumpy.object\_types.upload\_file\_list module**

```
class taniumpy.object_types.upload_file_list.UploadFileList  
    Bases: taniumpy.object_types.base.BaseType
```

**taniumpy.object\_types.upload\_file\_status module**

```
class taniumpy.object_types.upload_file_status.UploadFileStatus  
    Bases: taniumpy.object_types.base.BaseType
```

**taniumpy.object\_types.user module**

```
class taniumpy.object_types.user.User  
    Bases: taniumpy.object_types.base.BaseType
```

**taniumpy.object\_types.user\_list module**

```
class taniumpy.object_types.user_list.UserList  
    Bases: taniumpy.object_types.base.BaseType
```

### **taniumpy.object\_types.user\_permissions module**

```
class taniumpy.object_types.user_permissions.UserPermissions  
    Bases: taniumpy.object_types.base.BaseType
```

### **taniumpy.object\_types.user\_role module**

```
class taniumpy.object_types.user_role.UserRole  
    Bases: taniumpy.object_types.base.BaseType
```

### **taniumpy.object\_types.user\_role\_list module**

```
class taniumpy.object_types.user_role_list.UserRoleList  
    Bases: taniumpy.object_types.base.BaseType
```

### **taniumpy.object\_types.version\_aggregate module**

```
class taniumpy.object_types.version_aggregate.VersionAggregate  
    Bases: taniumpy.object_types.base.BaseType
```

### **taniumpy.object\_types.version\_aggregate\_list module**

```
class taniumpy.object_types.version_aggregate_list.VersionAggregateList  
    Bases: taniumpy.object_types.base.BaseType
```

### **taniumpy.object\_types.white\_listed\_url module**

```
class taniumpy.object_types.white_listed_url.WhiteListedUrl  
    Bases: taniumpy.object_types.base.BaseType
```

### **taniumpy.object\_types.white\_listed\_url\_list module**

```
class taniumpy.object_types.white_listed_url_list.WhiteListedUrlList  
    Bases: taniumpy.object_types.base.BaseType
```

### **taniumpy.object\_types.xml\_error module**

```
class taniumpy.object_types.xml_error.XmlError  
    Bases: taniumpy.object_types.base.BaseType
```

## **1.4 xmltodict module**

Makes working with XML feel like you are working with JSON

```
xmlltodict.parse(xml_input, encoding=None, expat=<module 'xml.parsers.expat' from
                  '/Library/Python/2.7/site-packages/_xmlplus/parsers/expat.pyc'>,
                  process_namespaces=False, namespace_separator=':', **kwargs)
```

Parse the given XML input and convert it into a dictionary.

*xml\_input* can either be a *string* or a file-like object.

If *xml\_attribs* is *True*, element attributes are put in the dictionary among regular child elements, using *@* as a prefix to avoid collisions. If set to *False*, they are just ignored.

Simple example:

```
>>> import xmlltodict
>>> doc = xmlltodict.parse("""
... <a prop="x">
...   <b>1</b>
...   <b>2</b>
... </a>
... """)
>>> doc['a']['@prop']
u'x'
>>> doc['a']['b']
[u'1', u'2']
```

If *item\_depth* is 0, the function returns a dictionary for the root element (default behavior). Otherwise, it calls *item\_callback* every time an item at the specified depth is found and returns *None* in the end (streaming mode).

The callback function receives two parameters: the *path* from the document root to the item (name-attrs pairs), and the *item* (dict). If the callback's return value is false-ish, parsing will be stopped with the *ParsingInterrupted* exception.

Streaming example:

```
>>> def handle(path, item):
...     print 'path:%s item:%s' % (path, item)
...     return True
...
>>> xmlltodict.parse("""
... <a prop="x">
...   <b>1</b>
...   <b>2</b>
... </a>""", item_depth=2, item_callback=handle)
path:[(u'a', {u'prop': u'x'})], (u'b', None)] item:1
path:[(u'a', {u'prop': u'x'})], (u'b', None)] item:2
```

The optional argument *postprocessor* is a function that takes *path*, *key* and *value* as positional arguments and returns a new (*key*, *value*) pair where both *key* and *value* may have changed. Usage example:

```
>>> def postprocessor(path, key, value):
...     try:
...         return key + ':int', int(value)
...     except (ValueError, TypeError):
...         return key, value
>>> xmlltodict.parse('<a><b>1</b><b>2</b><b>x</b></a>',
...                  postprocessor=postprocessor)
OrderedDict([(u'a', OrderedDict([(u'b:int', [1, 2]), (u'b', u'x')]))])
```

You can pass an alternate version of *expat* (such as *defusedexpat*) by using the *expat* parameter. E.g:

```
>>> import defusedexpat
>>> xmldict.parse('<a>hello</a>', expat=defusedexpat.pyexpat)
OrderedDict([(u'a', u'hello')])
```

`xmldict.unparse(input_dict, output=None, encoding='utf-8', full_document=True, **kwargs)`

Emit an XML document for the given `input_dict` (reverse of `parse`).

The resulting XML document is returned as a string, but if `output` (a file-like object) is specified, it is written there instead.

Dictionary keys prefixed with `attr_prefix` (default='@') are interpreted as XML node attributes, whereas keys equal to `cdata_key` (default='#text') are treated as character data.

The `pretty` parameter (default='False') enables pretty-printing. In this mode, lines are terminated with 'n' and indented with 't', but this can be customized with the `newl` and `indent` parameters.

## 1.5 ddt module

`ddt.data(*values)`

Method decorator to add to your test methods.

Should be added to methods of instances of `unittest.TestCase`.

`ddt.ddt(cls)`

Class decorator for subclasses of `unittest.TestCase`.

Apply this decorator to the test case class, and then decorate test methods with `@data`.

For each method decorated with `@data`, this will effectively create as many methods as data items are passed as parameters to `@data`.

The names of the test methods follow the pattern `original_test_name_{ordinal}_{data}`. `ordinal` is the position of the data argument, starting with 1.

For data we use a string representation of the data value converted into a valid python identifier. If `data.__name__` exists, we use that instead.

For each method decorated with `@file_data('test_data.json')`, the decorator will try to load the `test_data.json` file located relative to the python file containing the method that is decorated. It will, for each `test_name` key create as many methods in the list of values from the `data` key.

`ddt.file_data(value)`

Method decorator to add to your test methods.

Should be added to methods of instances of `unittest.TestCase`.

`value` should be a path relative to the directory of the file containing the decorated `unittest.TestCase`. The file should contain JSON encoded data, that can either be a list or a dict.

In case of a list, each value in the list will correspond to one test case, and the value will be concatenated to the test method name.

In case of a dict, keys will be used as suffixes to the name of the test case, and values will be fed as test data.

`ddt.is_hash_randomized()`

`ddt.mk_test_name(name, value, index=0)`

Generate a new name for a test case.

It will take the original test name and append an ordinal index and a string representation of the value, and convert the result into a valid python identifier by replacing extraneous characters with `_`.

If hash randomization is enabled (a feature available since 2.7.3/3.2.3 and enabled by default since 3.3) and a “non-trivial” value is passed this will omit the name argument by default. Set `PYTHONHASHSEED` to a fixed value before running tests in these cases to get the names back consistently or use the `__name__` attribute on data values.

A “trivial” value is a plain scalar, or a tuple or list consisting only of trivial values.

`ddt.unpack(func)`

Method decorator to add unpack feature.

## 1.6 threaded\_http module

Simple HTTP server for testing purposes

```
class threaded_http.Handler(request, client_address, server)
```

Bases: `BaseHTTPServer.BaseHTTPRequestHandler`

`__module__ = 'threaded_http'`

`do_GET()`

`log_message(format, *args)`

```
class threaded_http.ThreadedHTTPServer(server_address, RequestHandlerClass,
                                         bind_and_activate=True)
```

Bases: `SocketServer.ThreadingMixIn`, `BaseHTTPServer.HTTPServer`

Handle requests in a separate thread.

`__module__ = 'threaded_http'`

```
threaded_http.threaded_http(host='localhost', port=4443, verbosity=2)
```

establishes an HTTP server on host:port in a thread





## INDICES AND TABLES

- *genindex*
- *modindex*
- *search*



**d**

ddt, 322

**p**

pytan, 3

pytan.constants, 282

pytan.handler, 266

pytan.utils, 284

**t**

taniumpy, 306

taniumpy.object\_types, 308

taniumpy.object\_types.action, 308

taniumpy.object\_types.action\_list, 309

taniumpy.object\_types.action\_list\_info, 309

taniumpy.object\_types.action\_stop, 309

taniumpy.object\_types.action\_stop\_list, 309

taniumpy.object\_types.all\_objects, 309

taniumpy.object\_types.archived\_question, 309

taniumpy.object\_types.archived\_question\_list, 309

taniumpy.object\_types.audit\_data, 309

taniumpy.object\_types.base, 309

taniumpy.object\_types.cache\_filter, 311

taniumpy.object\_types.cache\_filter\_list, 311

taniumpy.object\_types.cache\_info, 311

taniumpy.object\_types.client\_count, 311

taniumpy.object\_types.client\_status, 311

taniumpy.object\_types.column, 311

taniumpy.object\_types.column\_set, 311

taniumpy.object\_types.computer\_group, 311

taniumpy.object\_types.computer\_group\_list, 311

taniumpy.object\_types.computer\_group\_spec, 312

taniumpy.object\_types.computer\_spec\_list, 312

taniumpy.object\_types.error\_list, 312

taniumpy.object\_types.filter, 312

taniumpy.object\_types.filter\_list, 312

taniumpy.object\_types.group, 312

taniumpy.object\_types.group\_list, 312

taniumpy.object\_types.metadata\_item, 312

taniumpy.object\_types.metadata\_list, 312

taniumpy.object\_types.object\_list, 313

taniumpy.object\_types.object\_list\_types, 313

taniumpy.object\_types.options, 313

taniumpy.object\_types.package\_file, 313

taniumpy.object\_types.package\_file\_list, 313

taniumpy.object\_types.package\_file\_status, 313

taniumpy.object\_types.package\_file\_status\_list, 313

taniumpy.object\_types.package\_file\_template, 313

taniumpy.object\_types.package\_file\_template\_list, 313

taniumpy.object\_types.package\_spec, 313

taniumpy.object\_types.package\_spec\_list, 314

taniumpy.object\_types.parameter, 314

taniumpy.object\_types.parameter\_list, 314

taniumpy.object\_types.parse\_job, 314

taniumpy.object\_types.parse\_job\_list, 314

taniumpy.object\_types.parse\_result, 314

taniumpy.object\_types.parse\_result\_group, 314

taniumpy.object\_types.parse\_result\_group\_list, 314

taniumpy.object\_types.parse\_result\_list, 314

taniumpy.object\_types.plugin, 315

taniumpy.object\_types.plugin\_argument, 315

`taniumpy.object_types.plugin_argument_list`, 315

`taniumpy.object_types.plugin_command_list`, 315

`taniumpy.object_types.plugin_list`, 315

`taniumpy.object_types.plugin_schedule`, 315

`taniumpy.object_types.plugin_schedule_list`, 315

`taniumpy.object_types.plugin_sql`, 315

`taniumpy.object_types.plugin_sql_column`, 315

`taniumpy.object_types.plugin_sql_result`, 316

`taniumpy.object_types.question`, 316

`taniumpy.object_types.question_list`, 316

`taniumpy.object_types.question_list_info`, 316

`taniumpy.object_types.result_info`, 316

`taniumpy.object_types.result_set`, 316

`taniumpy.object_types.row`, 317

`taniumpy.object_types.saved_action`, 317

`taniumpy.object_types.saved_action_approval`, 317

`taniumpy.object_types.saved_action_list`, 317

`taniumpy.object_types.saved_action_policy`, 317

`taniumpy.object_types.saved_action_row_id`, 317

`taniumpy.object_types.saved_question`, 317

`taniumpy.object_types.saved_question_list`, 317

`taniumpy.object_types.select`, 317

`taniumpy.object_types.select_list`, 318

`taniumpy.object_types.sensor`, 318

`taniumpy.object_types.sensor_list`, 318

`taniumpy.object_types.sensor_query`, 318

`taniumpy.object_types.sensor_query_list`, 318

`taniumpy.object_types.sensor_string_hints`, 318

`taniumpy.object_types.sensor_subcolumn`, 318

`taniumpy.object_types.sensor_subcolumn_list`, 318

`taniumpy.object_types.sensor_types`, 318

`taniumpy.object_types.soap_error`, 318

`taniumpy.object_types.system_setting`, 319

`taniumpy.object_types.system_settings_list`, 319

`taniumpy.object_types.system_status_aggregate`, 319

`taniumpy.object_types.system_status_list`, 319

`taniumpy.object_types.upload_file`, 319

`taniumpy.object_types.upload_file_list`, 319

`taniumpy.object_types.upload_file_status`, 319

`taniumpy.object_types.user`, 319

`taniumpy.object_types.user_list`, 319

`taniumpy.object_types.user_permissions`, 320

`taniumpy.object_types.user_role`, 320

`taniumpy.object_types.user_role_list`, 320

`taniumpy.object_types.version_aggregate`, 320

`taniumpy.object_types.version_aggregate_list`, 320

`taniumpy.object_types.white_listed_url`, 320

`taniumpy.object_types.white_listed_url_list`, 320

`taniumpy.object_types.xml_error`, 320

`taniumpy.question_asker`, 308

`taniumpy.session`, 307

`test_pytan_func`, 302

`test_pytan_unit`, 298

`threaded_http`, 323

**X**

`xmltodict`, 320

## Symbols

- `__author__` (in module `pytan`), 3
  - `__copyright__` (in module `pytan`), 3
  - `__license__` (in module `pytan`), 3
  - `__module__` (`test_pytan_func.InvalidServerTests` attribute), 303
  - `__module__` (`test_pytan_func.ValidServerTests` attribute), 303
  - `__module__` (`test_pytan_unit.TestDehumanizeExtractionUtils` attribute), 298
  - `__module__` (`test_pytan_unit.TestDehumanizeQuestionFilterUtils` attribute), 298
  - `__module__` (`test_pytan_unit.TestDehumanizeQuestionOptionUtils` attribute), 299
  - `__module__` (`test_pytan_unit.TestDehumanizeSensorUtils` attribute), 299
  - `__module__` (`test_pytan_unit.TestDeserializeBadXML` attribute), 299
  - `__module__` (`test_pytan_unit.TestGenericUtils` attribute), 300
  - `__module__` (`test_pytan_unit.TestManualBuildObjectUtils` attribute), 300
  - `__module__` (`test_pytan_unit.TestManualPackageDefValidateUtils` attribute), 301
  - `__module__` (`test_pytan_unit.TestManualQuestionFilterDefParseUtils` attribute), 301
  - `__module__` (`test_pytan_unit.TestManualQuestionFilterDefValidateUtils` attribute), 301
  - `__module__` (`test_pytan_unit.TestManualQuestionOptionDefParseUtils` attribute), 301
  - `__module__` (`test_pytan_unit.TestManualSensorDefParseUtils` attribute), 302
  - `__module__` (`test_pytan_unit.TestManualSensorDefValidateUtils` attribute), 302
  - `__module__` (`threaded_http.Handler` attribute), 323
  - `__module__` (`threaded_http.ThreadedHTTPServer` attribute), 323
  - `__version__` (in module `pytan`), 3
  - `_export_class_BaseType()` (`pytan.handler.Handler` method), 281
  - `_export_class_ResultSet()` (`pytan.handler.Handler` method), 281
  - `_export_format_csv()` (`pytan.handler.Handler` method), 281
  - `_export_format_json()` (`pytan.handler.Handler` method), 281
  - `_export_format_xml()` (`pytan.handler.Handler` method), 281
  - `_find()` (`pytan.handler.Handler` method), 281
  - `_get_multi()` (`pytan.handler.Handler` method), 281
  - `_get_package_def()` (`pytan.handler.Handler` method), 281
  - `_get_sensor_defs()` (`pytan.handler.Handler` method), 281
  - `_get_single()` (`pytan.handler.Handler` method), 281
  - `_single_find()` (`pytan.handler.Handler` method), 281
- ## A
- `Action` (class in `taniumpy.object_types.action`), 308
  - `ACTION_RESULT_STATUS` (in module `pytan.constants`), 282
  - `ActionList` (class in `taniumpy.object_types.action_list`), 309
  - `ActionListInfo` (class in `taniumpy.object_types.action_list_info`), 309
  - `ActionStop` (class in `taniumpy.object_types.action_stop`), 309
  - `ActionStopList` (class in `taniumpy.object_types.action_stop_list`), 309
  - `add()` (`taniumpy.session.Session` method), 307
  - `add_get_report_argparser()` (in module `pytan.utils`), 289
  - `add_get_object_report_argparser()` (in module `pytan.utils`), 289
  - `ADD_OBJECT` (`taniumpy.session.Session` attribute), 307
  - `add_report_file_options()` (in module `pytan.utils`), 289
  - `append()` (`taniumpy.object_types.base.BaseType` method), 309
  - `apply_options_obj()` (in module `pytan.utils`), 294
  - `ArchivedQuestion` (class in `taniumpy.object_types.archived_question`), 309
  - `ArchivedQuestionList` (class in `taniumpy.object_types.archived_question_list`), 309
  - `ask()` (`pytan.handler.Handler` method), 267
  - `ASK_KWARGS` (in module `pytan.constants`), 282
  - `ask_manual()` (`pytan.handler.Handler` method), 268

ask\_manual\_human() (pytan.handler.Handler method), 269

ask\_saved() (pytan.handler.Handler method), 267

AuditData (class in taniumpy.object\_types.audit\_data), 309

AUTH\_RES (taniumpy.session.Session attribute), 307

authenticate() (taniumpy.session.Session method), 307

AuthorizationError, 307

## B

BadResponseError, 307

BaseType (class in taniumpy.object\_types.base), 309

build\_group\_obj() (in module pytan.utils), 294

build\_manual\_q() (in module pytan.utils), 294

build\_metadatalist\_obj() (in module pytan.utils), 294

build\_param\_obj() (in module pytan.utils), 295

build\_param\_objlist() (in module pytan.utils), 295

build\_selectlist\_obj() (in module pytan.utils), 295

## C

CacheFilter (class in taniumpy.object\_types.cache\_filter), 311

CacheFilterList (class in taniumpy.object\_types.cache\_filter\_list), 311

CacheInfo (class in taniumpy.object\_types.cache\_info), 311

change\_console\_format() (in module pytan.utils), 285

check\_dictkey() (in module pytan.utils), 296

chk\_def\_key() (in module pytan.utils), 297

ClientCount (class in taniumpy.object\_types.client\_count), 311

ClientStatus (class in taniumpy.object\_types.client\_status), 311

Column (class in taniumpy.object\_types.column), 311

ColumnSet (class in taniumpy.object\_types.column\_set), 311

ComputerGroup (class in taniumpy.object\_types.computer\_group), 311

ComputerGroupList (class in taniumpy.object\_types.computer\_group\_list), 311

ComputerGroupSpec (class in taniumpy.object\_types.computer\_group\_spec), 312

ComputerSpecList (class in taniumpy.object\_types.computer\_spec\_list), 312

count (taniumpy.session.NoLogging attribute), 307

create\_from\_json() (pytan.handler.Handler method), 274

create\_group() (pytan.handler.Handler method), 277

create\_package() (pytan.handler.Handler method), 277

create\_sensor() (pytan.handler.Handler method), 278

create\_user() (pytan.handler.Handler method), 279

create\_whitelisted\_url() (pytan.handler.Handler method), 279

CustomArgFormat (class in pytan.utils), 284

CustomArgParse (class in pytan.utils), 284

## D

data() (in module ddt), 322

ddt (module), 322

ddt() (in module ddt), 322

DEBUG\_FORMAT (in module pytan.constants), 282

DefinitionParserError, 284

dehumanize\_package() (in module pytan.utils), 290

dehumanize\_question\_filters() (in module pytan.utils), 290

dehumanize\_question\_options() (in module pytan.utils), 291

dehumanize\_sensors() (in module pytan.utils), 291

delete() (pytan.handler.Handler method), 279

delete() (taniumpy.session.Session method), 307

DELETE\_OBJECT (taniumpy.session.Session attribute), 307

deploy\_action() (pytan.handler.Handler method), 270

deploy\_action\_asker() (pytan.handler.Handler method), 273

deploy\_action\_human() (pytan.handler.Handler method), 271

derive\_param\_default() (in module pytan.utils), 295

do\_GET() (threaded\_http.Handler method), 323

DynamicFormatter (class in taniumpy.session), 307

## E

emit() (pytan.utils.SplitStreamHandler method), 284

empty\_obj() (in module pytan.utils), 296

error() (pytan.utils.CustomArgParse method), 284

ErrorList (class in taniumpy.object\_types.error\_list), 312

explode\_json() (taniumpy.object\_types.base.BaseType method), 309

EXPORT\_MAPS (in module pytan.constants), 282

export\_obj() (pytan.handler.Handler method), 274

export\_to\_report\_file() (pytan.handler.Handler method), 275

extract\_filter() (in module pytan.utils), 291

extract\_options() (in module pytan.utils), 291

extract\_params() (in module pytan.utils), 291

extract\_selector() (in module pytan.utils), 292

## F

file\_data() (in module ddt), 322

Filter (class in taniumpy.object\_types.filter), 312

FILTER\_MAPS (in module pytan.constants), 282

FILTER\_RE (in module pytan.constants), 282

FilterList (class in taniumpy.object\_types.filter\_list), 312

find() (taniumpy.session.Session method), 307

- flatten\_jsonable() (taniumpy.object\_types.base.BaseType method), 309
- FORMATTER() (taniumpy.session.Session method), 307
- from\_jsonable() (taniumpy.object\_types.base.BaseType static method), 310
- fromSOAPBody() (taniumpy.object\_types.base.BaseType method), 309
- fromSOAPElement() (taniumpy.object\_types.base.BaseType method), 310
- fromSOAPElement() (taniumpy.object\_types.column.Column method), 311
- fromSOAPElement() (taniumpy.object\_types.column\_set.ColumnSet class method), 311
- fromSOAPElement() (taniumpy.object\_types.result\_info.ResultInfo class method), 316
- fromSOAPElement() (taniumpy.object\_types.result\_set.ResultSet class method), 316
- fromSOAPElement() (taniumpy.object\_types.row.Row class method), 317
- ## G
- get() (pytan.handler.Handler method), 280
- get\_all() (pytan.handler.Handler method), 280
- get\_ask\_kwargs() (in module pytan.utils), 293
- get\_dict\_list\_items() (in module pytan.utils), 285
- get\_dict\_list\_len() (in module pytan.utils), 286
- get\_filter\_obj() (in module pytan.utils), 296
- get\_grp\_opts() (in module pytan.utils), 289
- get\_kwargs\_int() (in module pytan.utils), 293
- get\_now() (in module pytan.utils), 286
- GET\_OBJ\_MAP (in module pytan.constants), 282
- get\_obj\_map() (in module pytan.utils), 293
- get\_obj\_params() (in module pytan.utils), 296
- GET\_OBJECT (taniumpy.session.Session attribute), 307
- get\_q\_obj\_map() (in module pytan.utils), 293
- get\_req\_kwargs() (in module pytan.utils), 293
- GET\_RESULT\_DATA (taniumpy.session.Session attribute), 307
- get\_result\_data() (pytan.handler.Handler method), 280
- GET\_RESULT\_INFO (taniumpy.session.Session attribute), 307
- get\_result\_info() (pytan.handler.Handler method), 281
- get\_server\_info() (taniumpy.session.Session method), 307
- get\_value() (taniumpy.session.DynamicFormatter method), 307
- getResultData() (taniumpy.session.Session method), 307
- getResultInfo() (taniumpy.session.Session method), 307
- Group (class in taniumpy.object\_types.group), 312
- GroupList (class in taniumpy.object\_types.group\_list), 312
- ## H
- Handler (class in pytan.handler), 266
- Handler (class in threaded\_http), 323
- HandlerError, 284
- http\_post() (in module taniumpy.session), 308
- HttpError, 307
- human\_time() (in module pytan.utils), 286
- HumanParserError, 284
- ## I
- IncorrectTypeException, 310
- INFO\_FORMAT (in module pytan.constants), 282
- INFO\_RES (taniumpy.session.Session attribute), 307
- InvalidServerTests (class in test\_pytan\_func), 303
- is\_auth (taniumpy.session.Session attribute), 307
- is\_dict() (in module pytan.utils), 285
- is\_hash\_randomized() (in module ddt), 322
- is\_list() (in module pytan.utils), 285
- is\_num() (in module pytan.utils), 285
- is\_str() (in module pytan.utils), 285
- ## J
- jsonify() (in module pytan.utils), 286
- ## L
- load\_file() (in module taniumpy.session), 308
- load\_taniumpy\_from\_json() (pytan.handler.Handler method), 274
- LOG\_LEVEL\_MAPS (in module pytan.constants), 283
- log\_message() (threaded\_http.Handler method), 323
- ## M
- map\_filter() (in module pytan.utils), 292
- map\_option() (in module pytan.utils), 292
- map\_options() (in module pytan.utils), 292
- MetadataItem (class in taniumpy.object\_types.metadata\_item), 312
- MetadataList (class in taniumpy.object\_types.metadata\_list), 312
- mk\_test\_name() (in module ddt), 322
- ## N
- NoLogging (class in taniumpy.session), 307
- nologging() (in module taniumpy.session), 308
- ## O
- ObjectList (class in taniumpy.object\_types.object\_list), 313
- OPTION\_MAPS (in module pytan.constants), 283

OPTION\_RE (in module pytan.constants), [283](#)  
Options (class in taniumpy.object\_types.options), [313](#)

## P

PackageFile (class in taniumpy.object\_types.package\_file), [313](#)  
PackageFileList (class in taniumpy.object\_types.package\_file\_list), [313](#)  
PackageFileStatus (class in taniumpy.object\_types.package\_file\_status), [313](#)  
PackageFileStatusList (class in taniumpy.object\_types.package\_file\_status\_list), [313](#)  
PackageFileTemplate (class in taniumpy.object\_types.package\_file\_template), [313](#)  
PackageFileTemplateList (class in taniumpy.object\_types.package\_file\_template\_list), [313](#)  
PackageSpec (class in taniumpy.object\_types.package\_spec), [313](#)  
PackageSpecList (class in taniumpy.object\_types.package\_spec\_list), [314](#)  
PARAM\_DELIM (in module pytan.constants), [283](#)  
PARAM\_KEY\_SPLIT (in module pytan.constants), [283](#)  
PARAM\_RE (in module pytan.constants), [283](#)  
PARAM\_SPLIT\_RE (in module pytan.constants), [283](#)  
Parameter (class in taniumpy.object\_types.parameter), [314](#)  
ParameterList (class in taniumpy.object\_types.parameter\_list), [314](#)  
parse() (in module xmltodict), [320](#)  
parse\_defs() (in module pytan.utils), [297](#)  
ParseJob (class in taniumpy.object\_types.parse\_job), [314](#)  
ParseJobList (class in taniumpy.object\_types.parse\_job\_list), [314](#)  
ParseResult (class in taniumpy.object\_types.parse\_result), [314](#)  
ParseResultGroup (class in taniumpy.object\_types.parse\_result\_group), [314](#)  
ParseResultGroupList (class in taniumpy.object\_types.parse\_result\_group\_list), [314](#)  
ParseResultList (class in taniumpy.object\_types.parse\_result\_list), [314](#)  
Plugin (class in taniumpy.object\_types.plugin), [315](#)  
PluginArgument (class in taniumpy.object\_types.plugin\_argument), [315](#)  
PluginArgumentList (class in taniumpy.object\_types.plugin\_argument\_list), [315](#)

PluginCommandList (class in taniumpy.object\_types.plugin\_command\_list), [315](#)  
PluginList (class in taniumpy.object\_types.plugin\_list), [315](#)  
PluginSchedule (class in taniumpy.object\_types.plugin\_schedule), [315](#)  
PluginScheduleList (class in taniumpy.object\_types.plugin\_schedule\_list), [315](#)  
PluginSql (class in taniumpy.object\_types.plugin\_sql), [315](#)  
PluginSqlColumn (class in taniumpy.object\_types.plugin\_sql\_column), [315](#)  
PluginSqlResult (class in taniumpy.object\_types.plugin\_sql\_result), [316](#)  
POLLING\_INTERVAL (taniumpy.question\_asker.QuestionAsker attribute), [308](#)  
port\_check() (in module pytan.utils), [287](#)  
print\_help() (pytan.utils.CustomArgParse method), [284](#)  
process\_create\_json\_object\_args() (in module pytan.utils), [289](#)  
process\_delete\_object\_args() (in module pytan.utils), [290](#)  
process\_get\_object\_args() (in module pytan.utils), [290](#)  
pytan (module), [3](#)  
pytan.constants (module), [282](#)  
pytan.handler (module), [266](#)  
pytan.utils (module), [284](#)

## Q

Q\_OBJ\_MAP (in module pytan.constants), [283](#)  
Question (class in taniumpy.object\_types.question), [316](#)  
question\_progress() (in module pytan.utils), [296](#)  
QuestionAsker (class in taniumpy.question\_asker), [308](#)  
QuestionList (class in taniumpy.object\_types.question\_list), [316](#)  
QuestionListInfo (class in taniumpy.object\_types.question\_list\_info), [316](#)  
QuestionTimeoutException, [308](#)

## R

remove\_logging\_handler() (in module pytan.utils), [285](#)  
REQ\_KWARGS (in module pytan.constants), [283](#)  
REQUEST\_BODY (taniumpy.session.Session attribute), [307](#)  
ResultInfo (class in taniumpy.object\_types.result\_info), [316](#)  
ResultSet (class in taniumpy.object\_types.result\_set), [316](#)  
Row (class in taniumpy.object\_types.row), [317](#)  
run() (taniumpy.question\_asker.QuestionAsker method), [308](#)  
RunFalse, [284](#)



## S

- `save()` (taniumpy.session.Session method), 307
- `SavedAction` (class in `taniumpy.object_types.saved_action`), 317
- `SavedActionApproval` (class in `taniumpy.object_types.saved_action_approval`), 317
- `SavedActionList` (class in `taniumpy.object_types.saved_action_list`), 317
- `SavedActionPolicy` (class in `taniumpy.object_types.saved_action_policy`), 317
- `SavedActionRowIdList` (class in `taniumpy.object_types.saved_action_row_id_list`), 317
- `SavedQuestion` (class in `taniumpy.object_types.saved_question`), 317
- `SavedQuestionList` (class in `taniumpy.object_types.saved_question_list`), 317
- `seconds_from_now()` (in module `pytan.utils`), 287
- `Select` (class in `taniumpy.object_types.select`), 317
- `SelectList` (class in `taniumpy.object_types.select_list`), 318
- `SELECTORS` (in module `pytan.constants`), 283
- `Sensor` (class in `taniumpy.object_types.sensor`), 318
- `SENSOR_TYPE_MAP` (in module `pytan.constants`), 283
- `SensorList` (class in `taniumpy.object_types.sensor_list`), 318
- `SensorQuery` (class in `taniumpy.object_types.sensor_query`), 318
- `SensorQueryList` (class in `taniumpy.object_types.sensor_query_list`), 318
- `SensorStringHints` (class in `taniumpy.object_types.sensor_string_hints`), 318
- `SensorSubcolumn` (class in `taniumpy.object_types.sensor_subcolumn`), 318
- `SensorSubcolumnList` (class in `taniumpy.object_types.sensor_subcolumn_list`), 318
- `server_version` (taniumpy.session.Session attribute), 307
- `Session` (class in `taniumpy.session`), 307
- `session_id` (taniumpy.session.Session attribute), 308
- `set_all_loglevels()` (in module `pytan.utils`), 285
- `set_log_levels()` (in module `pytan.utils`), 285
- `setPctCompleteThreshold()` (taniumpy.question\_asker.QuestionAsker method), 308
- `setup_ask_manual_argparser()` (in module `pytan.utils`), 289
- `setup_ask_saved_argparser()` (in module `pytan.utils`), 288
- `setup_console_logging()` (in module `pytan.utils`), 285
- `setup_create_json_object_argparser()` (in module `pytan.utils`), 288
- `setup_delete_object_argparser()` (in module `pytan.utils`), 288
- `setup_deploy_action_argparser()` (in module `pytan.utils`), 289
- `setup_get_object_argparser()` (in module `pytan.utils`), 288
- `setup_get_result_argparser()` (in module `pytan.utils`), 289
- `setup_parser()` (in module `pytan.utils`), 288
- `setup_stop_action_argparser()` (in module `pytan.utils`), 288
- `setup_test()` (`test_pytan_func.ValidServerTests` method), 303
- `setUpClass()` (`test_pytan_func.InvalidServerTests` class method), 303
- `setUpClass()` (`test_pytan_func.ValidServerTests` class method), 303
- `setUpClass()` (`test_pytan_unit.TestManualBuildObjectUtils` class method), 300
- `SOAP_PORT` (taniumpy.session.Session attribute), 307
- `SOAP_RES` (taniumpy.session.Session attribute), 307
- `SoapError` (class in `taniumpy.object_types.soap_error`), 318
- `spew()` (in module `test_pytan_func`), 306
- `SplitStreamHandler` (class in `pytan.utils`), 284
- `stop()` (taniumpy.question\_asker.QuestionAsker method), 308
- `stop_action()` (`pytan.handler.Handler` method), 273
- `SystemSetting` (class in `taniumpy.object_types.system_setting`), 319
- `SystemSettingsList` (class in `taniumpy.object_types.system_settings_list`), 319
- `SystemStatusAggregate` (class in `taniumpy.object_types.system_status_aggregate`), 319
- `SystemStatusList` (class in `taniumpy.object_types.system_status_list`), 319

## T

- `taniumpy` (module), 306
- `taniumpy.object_types` (module), 308
- `taniumpy.object_types.action` (module), 308
- `taniumpy.object_types.action_list` (module), 309
- `taniumpy.object_types.action_list_info` (module), 309
- `taniumpy.object_types.action_stop` (module), 309
- `taniumpy.object_types.action_stop_list` (module), 309
- `taniumpy.object_types.all_objects` (module), 309
- `taniumpy.object_types.archived_question` (module), 309
- `taniumpy.object_types.archived_question_list` (module), 309
- `taniumpy.object_types.audit_data` (module), 309
- `taniumpy.object_types.base` (module), 309
- `taniumpy.object_types.cache_filter` (module), 311

`taniumpy.object_types.cache_filter_list` (module), 311  
`taniumpy.object_types.cache_info` (module), 311  
`taniumpy.object_types.client_count` (module), 311  
`taniumpy.object_types.client_status` (module), 311  
`taniumpy.object_types.column` (module), 311  
`taniumpy.object_types.column_set` (module), 311  
`taniumpy.object_types.computer_group` (module), 311  
`taniumpy.object_types.computer_group_list` (module), 311  
`taniumpy.object_types.computer_group_spec` (module), 312  
`taniumpy.object_types.computer_spec_list` (module), 312  
`taniumpy.object_types.error_list` (module), 312  
`taniumpy.object_types.filter` (module), 312  
`taniumpy.object_types.filter_list` (module), 312  
`taniumpy.object_types.group` (module), 312  
`taniumpy.object_types.group_list` (module), 312  
`taniumpy.object_types.metadata_item` (module), 312  
`taniumpy.object_types.metadata_list` (module), 312  
`taniumpy.object_types.object_list` (module), 313  
`taniumpy.object_types.object_list_types` (module), 313  
`taniumpy.object_types.options` (module), 313  
`taniumpy.object_types.package_file` (module), 313  
`taniumpy.object_types.package_file_list` (module), 313  
`taniumpy.object_types.package_file_status` (module), 313  
`taniumpy.object_types.package_file_status_list` (module), 313  
`taniumpy.object_types.package_file_template` (module), 313  
`taniumpy.object_types.package_file_template_list` (module), 313  
`taniumpy.object_types.package_spec` (module), 313  
`taniumpy.object_types.package_spec_list` (module), 314  
`taniumpy.object_types.parameter` (module), 314  
`taniumpy.object_types.parameter_list` (module), 314  
`taniumpy.object_types.parse_job` (module), 314  
`taniumpy.object_types.parse_job_list` (module), 314  
`taniumpy.object_types.parse_result` (module), 314  
`taniumpy.object_types.parse_result_group` (module), 314  
`taniumpy.object_types.parse_result_group_list` (module), 314  
`taniumpy.object_types.parse_result_list` (module), 314  
`taniumpy.object_types.plugin` (module), 315  
`taniumpy.object_types.plugin_argument` (module), 315  
`taniumpy.object_types.plugin_argument_list` (module), 315  
`taniumpy.object_types.plugin_command_list` (module), 315  
`taniumpy.object_types.plugin_list` (module), 315  
`taniumpy.object_types.plugin_schedule` (module), 315  
`taniumpy.object_types.plugin_schedule_list` (module), 315  
`taniumpy.object_types.plugin_sql` (module), 315  
`taniumpy.object_types.plugin_sql_column` (module), 315  
`taniumpy.object_types.plugin_sql_result` (module), 316  
`taniumpy.object_types.question` (module), 316  
`taniumpy.object_types.question_list` (module), 316  
`taniumpy.object_types.question_list_info` (module), 316  
`taniumpy.object_types.result_info` (module), 316  
`taniumpy.object_types.result_set` (module), 316  
`taniumpy.object_types.row` (module), 317  
`taniumpy.object_types.saved_action` (module), 317  
`taniumpy.object_types.saved_action_approval` (module), 317  
`taniumpy.object_types.saved_action_list` (module), 317  
`taniumpy.object_types.saved_action_policy` (module), 317  
`taniumpy.object_types.saved_action_row_id_list` (module), 317  
`taniumpy.object_types.saved_question` (module), 317  
`taniumpy.object_types.saved_question_list` (module), 317  
`taniumpy.object_types.select` (module), 317  
`taniumpy.object_types.select_list` (module), 318  
`taniumpy.object_types.sensor` (module), 318  
`taniumpy.object_types.sensor_list` (module), 318  
`taniumpy.object_types.sensor_query` (module), 318  
`taniumpy.object_types.sensor_query_list` (module), 318  
`taniumpy.object_types.sensor_string_hints` (module), 318  
`taniumpy.object_types.sensor_subcolumn` (module), 318  
`taniumpy.object_types.sensor_subcolumn_list` (module), 318  
`taniumpy.object_types.sensor_types` (module), 318  
`taniumpy.object_types.soap_error` (module), 318  
`taniumpy.object_types.system_setting` (module), 319  
`taniumpy.object_types.system_settings_list` (module), 319  
`taniumpy.object_types.system_status_aggregate` (module), 319  
`taniumpy.object_types.system_status_list` (module), 319  
`taniumpy.object_types.upload_file` (module), 319  
`taniumpy.object_types.upload_file_list` (module), 319  
`taniumpy.object_types.upload_file_status` (module), 319  
`taniumpy.object_types.user` (module), 319  
`taniumpy.object_types.user_list` (module), 319  
`taniumpy.object_types.user_permissions` (module), 320  
`taniumpy.object_types.user_role` (module), 320  
`taniumpy.object_types.user_role_list` (module), 320  
`taniumpy.object_types.version_aggregate` (module), 320  
`taniumpy.object_types.version_aggregate_list` (module), 320  
`taniumpy.object_types.white_listed_url` (module), 320  
`taniumpy.object_types.white_listed_url_list` (module), 320  
`taniumpy.object_types.xml_error` (module), 320  
`taniumpy.question_asker` (module), 308  
`taniumpy.session` (module), 307  
`test_app_port()` (in module `pytan.utils`), 287

[test\\_ask\\_kwargs\(\)](#) (test\_pytan\_unit.TestGenericUtils method), 298  
[test\\_bad\\_chars\\_basetype\(\)](#) (test\_pytan\_unit.TestDeserializeBadXML method), 299  
[test\\_bad\\_chars\\_resultset\(\)](#) (test\_pytan\_unit.TestDeserializeBadXML method), 300  
[test\\_build\\_group\\_obj\(\)](#) (test\_pytan\_unit.TestManualBuildObjectUtils method), 300  
[test\\_build\\_manual\\_q\(\)](#) (test\_pytan\_unit.TestManualBuildObjectUtils method), 300  
[test\\_build\\_selectlist\\_obj\\_invalid\\_filter\(\)](#) (test\_pytan\_unit.TestManualBuildObjectUtils method), 300  
[test\\_build\\_selectlist\\_obj\\_missing\\_value\(\)](#) (test\_pytan\_unit.TestManualBuildObjectUtils method), 300  
[test\\_build\\_selectlist\\_obj\\_noparamssensorobj\\_noparams\(\)](#) (test\_pytan\_unit.TestManualBuildObjectUtils method), 300  
[test\\_build\\_selectlist\\_obj\\_noparamssensorobj\\_withparams\(\)](#) (test\_pytan\_unit.TestManualBuildObjectUtils method), 300  
[test\\_build\\_selectlist\\_obj\\_withparamssensorobj\\_noparams\(\)](#) (test\_pytan\_unit.TestManualBuildObjectUtils method), 300  
[test\\_build\\_selectlist\\_obj\\_withparamssensorobj\\_withparams\(\)](#) (test\_pytan\_unit.TestManualBuildObjectUtils method), 301  
[test\\_empty\\_args\\_dict\(\)](#) (test\_pytan\_unit.TestDehumanizeSensorUtils method), 299  
[test\\_empty\\_args\\_list\(\)](#) (test\_pytan\_unit.TestDehumanizeSensorUtils method), 299  
[test\\_empty\\_args\\_str\(\)](#) (test\_pytan\_unit.TestDehumanizeSensorUtils method), 299  
[test\\_empty\\_filterlist\(\)](#) (test\_pytan\_unit.TestDehumanizeQuestionFilterUtils method), 298  
[test\\_empty\\_filterstr\(\)](#) (test\_pytan\_unit.TestDehumanizeQuestionFilterUtils method), 299  
[test\\_empty\\_obj\(\)](#) (test\_pytan\_unit.TestGenericUtils method), 300  
[test\\_empty\\_optionlist\(\)](#) (test\_pytan\_unit.TestDehumanizeQuestionOptionUtils method), 299  
[test\\_empty\\_optionstr\(\)](#) (test\_pytan\_unit.TestDehumanizeQuestionOptionUtils method), 299  
[test\\_extract\\_filter\\_invalid\(\)](#) (test\_pytan\_unit.TestDehumanizeExtractionUtils method), 298  
[test\\_extract\\_filter\\_nofilter\(\)](#) (test\_pytan\_unit.TestDehumanizeExtractionUtils method), 298  
[test\\_extract\\_filter\\_valid\(\)](#) (test\_pytan\_unit.TestDehumanizeExtractionUtils method), 298  
[test\\_extract\\_filter\\_valid\\_all\(\)](#) (test\_pytan\_unit.TestDehumanizeExtractionUtils method), 298  
[test\\_extract\\_options\\_invalid\\_option\(\)](#) (test\_pytan\_unit.TestDehumanizeExtractionUtils method), 298  
[test\\_extract\\_options\\_many\(\)](#) (test\_pytan\_unit.TestDehumanizeExtractionUtils method), 298  
[test\\_extract\\_options\\_missing\\_value\\_max\\_data\\_age\(\)](#) (test\_pytan\_unit.TestDehumanizeExtractionUtils method), 298  
[test\\_extract\\_options\\_missing\\_value\\_value\\_type\(\)](#) (test\_pytan\_unit.TestDehumanizeExtractionUtils method), 298  
[test\\_extract\\_options\\_nooptions\(\)](#) (test\_pytan\_unit.TestDehumanizeExtractionUtils method), 298  
[test\\_extract\\_options\\_single\(\)](#) (test\_pytan\_unit.TestDehumanizeExtractionUtils method), 298  
[test\\_extract\\_params\(\)](#) (test\_pytan\_unit.TestDehumanizeExtractionUtils method), 298  
[test\\_extract\\_params\\_missing\\_seperator\(\)](#) (test\_pytan\_unit.TestDehumanizeExtractionUtils method), 298  
[test\\_extract\\_params\\_multiparams\(\)](#) (test\_pytan\_unit.TestDehumanizeExtractionUtils method), 298  
[test\\_extract\\_params\\_noparams\(\)](#) (test\_pytan\_unit.TestDehumanizeExtractionUtils method), 298  
[test\\_extract\\_selector\(\)](#) (test\_pytan\_unit.TestDehumanizeExtractionUtils method), 298  
[test\\_extract\\_selector\\_use\\_name\\_if\\_noselector\(\)](#) (test\_pytan\_unit.TestDehumanizeExtractionUtils method), 298  
[test\\_get\\_obj\\_map\(\)](#) (test\_pytan\_unit.TestGenericUtils method), 300  
[test\\_get\\_obj\\_map\\_by\\_id\(\)](#) (test\_pytan\_unit.TestGenericUtils method), 300  
[test\\_invalid1\(\)](#) (test\_pytan\_unit.TestManualQuestionFilterDefValidateUtils method), 301  
[test\\_invalid1\(\)](#) (test\_pytan\_unit.TestManualSensorDefValidateUtils method), 302  
[test\\_invalid2\(\)](#) (test\_pytan\_unit.TestManualPackageDefValidateUtils method), 301  
[test\\_invalid2\(\)](#) (test\_pytan\_unit.TestManualSensorDefValidateUtils method), 302

test_invalid3() (test_pytan_unit.TestManualSensorDefValidateUtils method), 302	test_invalid_export_basetype_2_invalid_export_basetype_csv_bad_sort_subj (test_pytan_func.ValidServerTests method), 303
test_invalid4() (test_pytan_unit.TestManualSensorDefValidateUtils method), 302	test_invalid_export_basetype_3_invalid_export_basetype_csv_bad_sort_type (test_pytan_func.ValidServerTests method), 303
test_invalid_connect_1_bad_username() (test_pytan_func.InvalidServerTests method), 303	test_invalid_export_basetype_4_invalid_export_basetype_xml_bad_minimal (test_pytan_func.ValidServerTests method), 303
test_invalid_connect_2_bad_host_and_non_ssl_port() (test_pytan_func.InvalidServerTests method), 303	test_invalid_export_basetype_5_invalid_export_basetype_json_bad_include (test_pytan_func.ValidServerTests method), 303
test_invalid_connect_3_bad_password() (test_pytan_func.InvalidServerTests method), 303	test_invalid_export_basetype_6_invalid_export_basetype_json_bad_explosion (test_pytan_func.ValidServerTests method), 303
test_invalid_connect_4_bad_host_and_bad_port() (test_pytan_func.InvalidServerTests method), 303	test_invalid_export_basetype_7_invalid_export_basetype_bad_format() (test_pytan_func.ValidServerTests method), 303
test_invalid_create_object_1_invalid_create_sensor() (test_pytan_func.ValidServerTests method), 303	test_invalid_export_resultset_1_invalid_export_resultset_csv_bad_sort_subj (test_pytan_func.ValidServerTests method), 303
test_invalid_create_object_from_json_1_invalid_create_savedata_invalid_json() (test_pytan_func.ValidServerTests method), 303	test_invalid_export_resultset_2_invalid_export_resultset_csv_bad_sort_type (test_pytan_func.ValidServerTests method), 303
test_invalid_create_object_from_json_2_invalid_create_client_invalid_json() (test_pytan_func.ValidServerTests method), 303	test_invalid_export_resultset_3_invalid_export_resultset_csv_bad_expansion (test_pytan_func.ValidServerTests method), 303
test_invalid_create_object_from_json_3_invalid_create_user_invalid_json() (test_pytan_func.ValidServerTests method), 303	test_invalid_export_resultset_4_invalid_export_resultset_csv_bad_sensors (test_pytan_func.ValidServerTests method), 304
test_invalid_create_object_from_json_4_invalid_create_settings_invalid_json() (test_pytan_func.ValidServerTests method), 303	test_invalid_export_resultset_5_invalid_export_resultset_bad_format() (test_pytan_func.ValidServerTests method), 304
test_invalid_deploy_action_1_invalid_deploy_action_run_failed() (test_pytan_func.ValidServerTests method), 303	test_invalid_filter1() (test_pytan_unit.TestDehumanizeQuestionFilterUtils method), 299
test_invalid_deploy_action_2_invalid_deploy_action_package_help() (test_pytan_func.ValidServerTests method), 303	test_invalid_filter2() (test_pytan_unit.TestDehumanizeQuestionFilterUtils method), 299
test_invalid_deploy_action_3_invalid_deploy_action_package() (test_pytan_func.ValidServerTests method), 303	test_invalid_filter3() (test_pytan_unit.TestDehumanizeQuestionFilterUtils method), 299
test_invalid_deploy_action_4_invalid_deploy_action_options_help() (test_pytan_func.ValidServerTests method), 303	test_invalid_get_object_1_invalid_get_action_single_by_name() (test_pytan_func.ValidServerTests method), 304
test_invalid_deploy_action_5_invalid_deploy_action_empty_package() (test_pytan_func.ValidServerTests method), 303	test_invalid_get_object_2_invalid_get_question_by_name() (test_pytan_func.ValidServerTests method), 304
test_invalid_deploy_action_6_invalid_deploy_action_filters_help() (test_pytan_func.ValidServerTests method), 303	test_invalid_option1() (test_pytan_unit.TestDehumanizeQuestionOptionUtils method), 299
test_invalid_deploy_action_7_invalid_deploy_action_missing_parameters() (test_pytan_func.ValidServerTests method), 303	test_invalid_option2() (test_pytan_unit.TestDehumanizeQuestionOptionUtils method), 299
test_invalid_export_basetype_1_invalid_export_basetype_csv_bad_sort_subj (test_pytan_func.ValidServerTests method), 303	test_invalid_port() (test_pytan_unit.TestGenericUtils method), 300
test_invalid_export_basetype_2_invalid_export_basetype_csv_bad_sort_type (test_pytan_func.ValidServerTests method), 303	test_invalid_question1_invalid_ask_manual_human_question_paramater_type (test_pytan_func.ValidServerTests method), 303

304 test\_parse\_complex() (test\_pytan\_unit.TestManualSensorDefParseUtils method), 302  
 test\_invalid\_question\_2\_invalid\_ask\_manual\_human\_question\_filter\_help() (test\_pytan\_func.ValidServerTests method), test\_parse\_dict\_hash() (test\_pytan\_unit.TestManualSensorDefParseUtils method), 302  
 304 test\_invalid\_question\_3\_invalid\_ask\_manual\_human\_question\_option\_help() (test\_pytan\_unit.TestManualSensorDefParseUtils method), test\_parse\_dict\_id() (test\_pytan\_unit.TestManualSensorDefParseUtils method), 302  
 304 test\_invalid\_question\_4\_invalid\_ask\_manual\_human\_question\_filter\_help() (test\_pytan\_func.ValidServerTests method), test\_parse\_dict\_name() (test\_pytan\_unit.TestManualSensorDefParseUtils method), 302  
 test\_invalid\_question\_5\_invalid\_ask\_manual\_human\_question\_option\_help() (test\_pytan\_unit.TestManualQuestionOptionDefParseUtils method), test\_parse\_emptydict() (test\_pytan\_unit.TestManualQuestionFilterDefParseUtils method), 301  
 304 test\_invalid\_question\_6\_invalid\_ask\_manual\_human\_question\_option\_help() (test\_pytan\_func.ValidServerTests method), test\_parse\_emptydict() (test\_pytan\_unit.TestManualSensorDefParseUtils method), 301  
 test\_invalid\_question\_7\_invalid\_ask\_manual\_question\_sensor\_help() (test\_pytan\_unit.TestManualQuestionOptionDefParseUtils method), test\_parse\_emptylist() (test\_pytan\_unit.TestManualQuestionOptionDefParseUtils method), 301  
 304 test\_invalid\_question\_8\_invalid\_ask\_manual\_human\_question\_sensor\_help() (test\_pytan\_func.ValidServerTests method), test\_parse\_emptylist() (test\_pytan\_unit.TestManualSensorDefParseUtils method), 302  
 test\_is\_dict() (test\_pytan\_unit.TestGenericUtils method), test\_parse\_emptystr() (test\_pytan\_unit.TestManualQuestionFilterDefParseUtils method), 301  
 300 test\_is\_list() (test\_pytan\_unit.TestGenericUtils method), test\_parse\_emptystr() (test\_pytan\_unit.TestManualQuestionOptionDefParseUtils method), 301  
 test\_is\_not\_dict() (test\_pytan\_unit.TestGenericUtils method), test\_parse\_emptystr() (test\_pytan\_unit.TestManualSensorDefParseUtils method), 302  
 300 test\_is\_not\_list() (test\_pytan\_unit.TestGenericUtils method), test\_parse\_list() (test\_pytan\_unit.TestManualQuestionOptionDefParseUtils method), 301  
 test\_is\_not\_num() (test\_pytan\_unit.TestGenericUtils method), test\_parse\_multi\_filter() (test\_pytan\_unit.TestManualQuestionFilterDefParseUtils method), 301  
 300 test\_is\_not\_str() (test\_pytan\_unit.TestGenericUtils method), test\_parse\_noargs() (test\_pytan\_unit.TestManualQuestionFilterDefParseUtils method), 301  
 test\_is\_num() (test\_pytan\_unit.TestGenericUtils method), test\_parse\_noargs() (test\_pytan\_unit.TestManualQuestionOptionDefParseUtils method), 301  
 300 test\_is\_str() (test\_pytan\_unit.TestGenericUtils method), test\_parse\_noargs() (test\_pytan\_unit.TestManualSensorDefParseUtils method), 302  
 test\_jsonify() (test\_pytan\_unit.TestGenericUtils method), test\_parse\_none() (test\_pytan\_unit.TestManualQuestionFilterDefParseUtils method), 301  
 300 test\_multi\_filter\_list() (test\_pytan\_unit.TestDehumanizeQuestionFilterUtils method), test\_parse\_none() (test\_pytan\_unit.TestManualQuestionOptionDefParseUtils method), 301  
 test\_multi\_list\_complex() (test\_pytan\_unit.TestDehumanizeSensorUtils method), test\_parse\_options\_dict() (test\_pytan\_unit.TestManualQuestionOptionDefParseUtils method), 301  
 test\_option\_list\_many() (test\_pytan\_unit.TestDehumanizeQuestionOptionUtils method), test\_parse\_single\_filter() (test\_pytan\_unit.TestManualQuestionFilterDefParseUtils method), 301  
 299 test\_option\_list\_multi() (test\_pytan\_unit.TestDehumanizeQuestionOptionUtils method), test\_parse\_str() (test\_pytan\_unit.TestManualQuestionFilterDefParseUtils method), 301  
 test\_option\_list\_single() (test\_pytan\_unit.TestDehumanizeQuestionOptionUtils method), test\_parse\_str() (test\_pytan\_unit.TestManualQuestionOptionDefParseUtils method), 302  
 299 test\_option\_str() (test\_pytan\_unit.TestDehumanizeQuestionOptionUtils method), test\_parse\_strl() (test\_pytan\_unit.TestManualSensorDefParseUtils method), 302  
 test\_parse\_strl() (test\_pytan\_unit.TestManualSensorDefParseUtils method), 302  
 test\_pytan\_func (module), 302



test_pytan_unit (module), 298	304
test_req_kwargs() (test_pytan_unit.TestGenericUtils method), 300	test_valid_create_object_from_json_4_create_action_from_json() (test_pytan_func.ValidServerTests method), 304
test_single_filter_list() (test_pytan_unit.TestDehumanizeQuestionFilterUtils method), 299	test_valid_create_object_from_json_5_create_sensor_from_json() (test_pytan_func.ValidServerTests method), 304
test_single_filter_str() (test_pytan_unit.TestDehumanizeQuestionFilterUtils method), 299	test_valid_create_object_from_json_6_create_question_from_json() (test_pytan_func.ValidServerTests method), 304
test_single_str() (test_pytan_unit.TestDehumanizeSensorUtils method), 299	test_valid_create_object_from_json_7_create_whitelisted_url_from_json() (test_pytan_func.ValidServerTests method), 304
test_single_str_complex1() (test_pytan_unit.TestDehumanizeSensorUtils method), 299	test_valid_create_object_from_json_8_create_group_from_json() (test_pytan_func.ValidServerTests method), 304
test_single_str_complex2() (test_pytan_unit.TestDehumanizeSensorUtils method), 299	test_valid_deploy_action_1_deploy_action_simple_against_windows_comp (test_pytan_func.ValidServerTests method), 304
test_single_str_with_filter() (test_pytan_unit.TestDehumanizeSensorUtils method), 299	test_valid_deploy_action_2_deploy_action_simple_without_results() (test_pytan_func.ValidServerTests method), 304
test_valid1() (test_pytan_unit.TestManualPackageDefValidateUtils method), 301	test_valid_deploy_action_3_deploy_action_with_params_against_windows_comp (test_pytan_func.ValidServerTests method), 304
test_valid1() (test_pytan_unit.TestManualQuestionFilterDefValidateUtils method), 301	test_valid_deploy_action_4_deploy_action_simple() (test_pytan_func.ValidServerTests method), 304
test_valid1() (test_pytan_unit.TestManualSensorDefValidateUtils method), 302	test_valid_export_basetype_10_export_basetype_xml_default_options() (test_pytan_func.ValidServerTests method), 304
test_valid2() (test_pytan_unit.TestManualPackageDefValidateUtils method), 301	test_valid_export_basetype_11_export_basetype_csv_with_explode_true() (test_pytan_func.ValidServerTests method), 304
test_valid2() (test_pytan_unit.TestManualQuestionFilterDefValidateUtils method), 301	test_valid_export_basetype_12_export_basetype_json_explode_false() (test_pytan_func.ValidServerTests method), 304
test_valid2() (test_pytan_unit.TestManualSensorDefValidateUtils method), 302	test_valid_export_basetype_13_export_basetype_json_type_false() (test_pytan_func.ValidServerTests method), 304
test_valid3() (test_pytan_unit.TestManualSensorDefValidateUtils method), 302	test_valid_export_basetype_14_export_basetype_json_default_options() (test_pytan_func.ValidServerTests method), 304
test_valid4() (test_pytan_unit.TestManualSensorDefValidateUtils method), 302	test_valid_export_basetype_1_export_basetype_csv_with_sort_list() (test_pytan_func.ValidServerTests method), 304
test_valid_create_object_1_create_user() (test_pytan_func.ValidServerTests method), 304	test_valid_export_basetype_2_export_basetype_csv_with_explode_false() (test_pytan_func.ValidServerTests method), 304
test_valid_create_object_2_create_package() (test_pytan_func.ValidServerTests method), 304	test_valid_export_basetype_3_export_basetype_json_type_true() (test_pytan_func.ValidServerTests method), 304
test_valid_create_object_3_create_group() (test_pytan_func.ValidServerTests method), 304	test_valid_export_basetype_4_export_basetype_xml_minimal_false() (test_pytan_func.ValidServerTests method), 304
test_valid_create_object_4_create_whitelisted_url() (test_pytan_func.ValidServerTests method), 304	
test_valid_create_object_from_json_1_create_package_from_json() (test_pytan_func.ValidServerTests method), 304	
test_valid_create_object_from_json_2_create_user_from_json() (test_pytan_func.ValidServerTests method), 304	
test_valid_create_object_from_json_3_create_saved_question_from_json() (test_pytan_func.ValidServerTests method), 304	

305	test_valid_export_basetype_5_export_basetype_xml_minimal_destroy() (test_pytan_func.ValidServerTests method),	305	test_valid_get_object_10_get_all_saved_questions() (test_pytan_func.ValidServerTests method),
305	test_valid_export_basetype_6_export_basetype_csv_with_sensor_valid_destroy() (test_pytan_func.ValidServerTests method),	305	test_valid_get_object_11_get_user_by_name() (test_pytan_func.ValidServerTests method),
305	test_valid_export_basetype_7_export_basetype_csv_defaults_optional_destroy() (test_pytan_func.ValidServerTests method),	305	test_valid_get_object_12_get_all_userroles() (test_pytan_func.ValidServerTests method),
305	test_valid_export_basetype_8_export_basetype_json_explode_destroy() (test_pytan_func.ValidServerTests method),	305	test_valid_get_object_13_get_all_questions() (test_pytan_func.ValidServerTests method),
305	test_valid_export_basetype_9_export_basetype_csv_with_sensor_valid_destroy() (test_pytan_func.ValidServerTests method),	305	test_valid_get_object_14_get_sensor_by_id() (test_pytan_func.ValidServerTests method),
305	test_valid_export_resultset_10_export_resultset_csv_defaults_optional_destroy() (test_pytan_func.ValidServerTests method),	305	test_valid_get_object_15_get_all_groups() (test_pytan_func.ValidServerTests method),
305	test_valid_export_resultset_11_export_resultset_csv_type_true_destroy() (test_pytan_func.ValidServerTests method),	305	test_valid_get_object_16_get_all_sensors() (test_pytan_func.ValidServerTests method),
305	test_valid_export_resultset_12_export_resultset_csv_all_options_destroy() (test_pytan_func.ValidServerTests method),	305	test_valid_get_object_17_get_sensor_by_mixed() (test_pytan_func.ValidServerTests method),
305	test_valid_export_resultset_13_export_resultset_csv_sort_false_destroy() (test_pytan_func.ValidServerTests method),	305	test_valid_get_object_18_get_whitelisted_url_by_id() (test_pytan_func.ValidServerTests method),
305	test_valid_export_resultset_1_export_resultset_json() (test_pytan_func.ValidServerTests method),	305	test_valid_get_object_19_get_group_by_name() (test_pytan_func.ValidServerTests method),
305	test_valid_export_resultset_2_export_resultset_csv_sensor_true_destroy() (test_pytan_func.ValidServerTests method),	305	test_valid_get_object_1_get_all_users() (test_pytan_func.ValidServerTests method),
305	test_valid_export_resultset_3_export_resultset_csv_type_false_destroy() (test_pytan_func.ValidServerTests method),	305	test_valid_get_object_20_get_all_whitelisted_urls() (test_pytan_func.ValidServerTests method),
305	test_valid_export_resultset_4_export_resultset_csv_expand_false_destroy() (test_pytan_func.ValidServerTests method),	305	test_valid_get_object_21_get_sensor_by_hash() (test_pytan_func.ValidServerTests method),
305	test_valid_export_resultset_5_export_resultset_csv_sort_empty_destroy() (test_pytan_func.ValidServerTests method),	305	test_valid_get_object_22_get_package_by_name() (test_pytan_func.ValidServerTests method),
305	test_valid_export_resultset_6_export_resultset_csv_sort_true_destroy() (test_pytan_func.ValidServerTests method),	305	test_valid_get_object_23_get_all_clients() (test_pytan_func.ValidServerTests method),
305	test_valid_export_resultset_7_export_resultset_csv_sort_list_destroy() (test_pytan_func.ValidServerTests method),	305	test_valid_get_object_24_get_sensor_by_names() (test_pytan_func.ValidServerTests method),
305	test_valid_export_resultset_8_export_resultset_csv_sensor_false_destroy() (test_pytan_func.ValidServerTests method),	305	test_valid_get_object_25_get_all_packages() (test_pytan_func.ValidServerTests method),
305	test_valid_export_resultset_9_export_resultset_csv_expand_true_destroy() (test_pytan_func.ValidServerTests method),	305	test_valid_get_object_26_get_saved_question_by_name() (test_pytan_func.ValidServerTests method),

306	test_valid_get_object_27_get_all_actions() (test_pytan_func.ValidServerTests method), 306	306	test_valid_question_16_ask_saved_question_by_name() (test_pytan_func.ValidServerTests method), 306
306	test_valid_get_object_28_get_user_by_id() (test_pytan_func.ValidServerTests method), 306	306	test_valid_question_17_ask_manual_human_question_sensor_with_paramet (test_pytan_func.ValidServerTests method), 306
306	test_valid_get_object_29_get_sensor_by_name() (test_pytan_func.ValidServerTests method), 306	306	test_valid_question_1_ask_manual_human_question_sensor_with_paramet (test_pytan_func.ValidServerTests method), 306
306	test_valid_get_object_2_get_action_by_id() (test_pytan_func.ValidServerTests method), 306	306	test_valid_question_2_ask_manual_human_question_simple_single_sensor (test_pytan_func.ValidServerTests method), 306
306	test_valid_get_object_30_get_saved_action_by_name() (test_pytan_func.ValidServerTests method), 306	306	test_valid_question_3_ask_manual_human_question_sensor_with_filter_and (test_pytan_func.ValidServerTests method), 306
306	test_valid_get_object_3_get_question_by_id() (test_pytan_func.ValidServerTests method), 306	306	test_valid_question_4_ask_manual_human_question_sensor_without_param (test_pytan_func.ValidServerTests method), 306
306	test_valid_get_object_4_get_saved_question_by_names() (test_pytan_func.ValidServerTests method), 306	306	test_valid_question_5_ask_manual_human_question_complex_query2() (test_pytan_func.ValidServerTests method), 306
306	test_valid_get_object_5_get_userrole_by_id() (test_pytan_func.ValidServerTests method), 306	306	test_valid_question_6_ask_manual_human_question_complex_query1() (test_pytan_func.ValidServerTests method), 306
306	test_valid_get_object_6_get_all_saved_actions() (test_pytan_func.ValidServerTests method), 306	306	test_valid_question_7_ask_saved_question_by_name_in_list() (test_pytan_func.ValidServerTests method), 306
306	test_valid_get_object_7_get_leader_clients() (test_pytan_func.ValidServerTests method), 306	306	test_valid_question_8_ask_manual_human_question_multiple_sensors_wit (test_pytan_func.ValidServerTests method), 306
306	test_valid_get_object_8_get_all_settings() (test_pytan_func.ValidServerTests method), 306	306	test_valid_question_9_ask_manual_question_sensor_complex() (test_pytan_func.ValidServerTests method), 306
306	test_valid_get_object_9_get_setting_by_name() (test_pytan_func.ValidServerTests method), 306	306	test_valid_simple_list() (test_pytan_unit.TestDehumanizeSensorUtils method), 299
306	test_valid_question_10_ask_manual_human_question_sensor_with_paramet (test_pytan_func.ValidServerTests method), 306	306	test_valid_simple_str_hash_selector() (test_pytan_unit.TestDehumanizeSensorUtils method), 299
306	test_valid_question_11_ask_manual_human_question_sensor_with_filter (test_pytan_func.ValidServerTests method), 306	306	test_valid_simple_str_id_selector() (test_pytan_unit.TestDehumanizeSensorUtils method), 299
306	test_valid_question_12_ask_manual_human_question_sensor_with_filters (test_pytan_func.ValidServerTests method), 306	306	test_valid_simple_str_name_selector() (test_pytan_unit.TestDehumanizeSensorUtils method), 299
306	test_valid_question_13_ask_manual_human_question_simple_multiple_sens (test_pytan_func.ValidServerTests method), 306	306	test_version_higher() (test_pytan_unit.TestGenericUtils method), 300
306	test_valid_question_14_ask_manual_human_question_multiple_sensors (test_pytan_func.ValidServerTests method), 306	306	test_version_lower() (test_pytan_unit.TestGenericUtils method), 300
306	test_valid_question_15_ask_manual_human_question_sensor_with_paramet (test_pytan_func.ValidServerTests method), 306	306	TestDehumanizeFixedIoUtils (class in test_pytan_unit), 298
306		306	TestDehumanizeQuestionFilterUtils (class in test_pytan_unit), 298



TestDehumanizeQuestionOptionUtils (class in test\_pytan\_unit), 299

TestDehumanizeSensorUtils (class in test\_pytan\_unit), 299

TestDeserializeBadXML (class in test\_pytan\_unit), 299

TestGenericUtils (class in test\_pytan\_unit), 300

TestManualBuildObjectUtils (class in test\_pytan\_unit), 300

TestManualPackageDefValidateUtils (class in test\_pytan\_unit), 301

TestManualQuestionFilterDefParseUtils (class in test\_pytan\_unit), 301

TestManualQuestionFilterDefValidateUtils (class in test\_pytan\_unit), 301

TestManualQuestionOptionDefParseUtils (class in test\_pytan\_unit), 301

TestManualSensorDefParseUtils (class in test\_pytan\_unit), 302

TestManualSensorDefValidateUtils (class in test\_pytan\_unit), 302

threaded\_http (module), 323

threaded\_http() (in module threaded\_http), 323

ThreadedHTTPServer (class in threaded\_http), 323

to\_flat\_dict() (taniumpy.object\_types.base.BaseType method), 310

to\_flat\_dict\_explode\_json() (taniumpy.object\_types.base.BaseType method), 310

to\_json() (taniumpy.object\_types.base.BaseType static method), 310

to\_json() (taniumpy.object\_types.result\_set.ResultSet static method), 316

to\_jsonable() (taniumpy.object\_types.base.BaseType method), 310

to\_jsonable() (taniumpy.object\_types.result\_set.ResultSet method), 316

toSOAPBody() (taniumpy.object\_types.base.BaseType method), 310

toSOAPElement() (taniumpy.object\_types.base.BaseType method), 310

## U

unpack() (in module ddt), 323

unparse() (in module xmldict), 322

UPDATE\_OBJECT (taniumpy.session.Session attribute), 307

UploadFile (class in taniumpy.object\_types.upload\_file), 319

UploadFileList (class in taniumpy.object\_types.upload\_file\_list), 319

UploadFileStatus (class in taniumpy.object\_types.upload\_file\_status), 319

User (class in taniumpy.object\_types.user), 319

UserList (class in taniumpy.object\_types.user\_list), 319

UserPermissions (class in taniumpy.object\_types.user\_permissions), 320

UserRole (class in taniumpy.object\_types.user\_role), 320

UserRoleList (class in taniumpy.object\_types.user\_role\_list), 320

## V

val\_package\_def() (in module pytan.utils), 297

val\_q\_filter\_defs() (in module pytan.utils), 297

val\_sensor\_defs() (in module pytan.utils), 298

ValidServerTests (class in test\_pytan\_func), 303

version\_check() (in module pytan.utils), 287

VersionAggregate (class in taniumpy.object\_types.version\_aggregate), 320

VersionAggregateList (class in taniumpy.object\_types.version\_aggregate\_list), 320

## W

WhiteListedUrl (class in taniumpy.object\_types.white\_listed\_url), 320

WhiteListedUrlList (class in taniumpy.object\_types.white\_listed\_url\_list), 320

write\_csv() (taniumpy.object\_types.base.BaseType static method), 310

write\_csv() (taniumpy.object\_types.result\_set.ResultSet static method), 316

## X

xml\_fix() (in module taniumpy.object\_types.base), 310

xml\_fix() (in module taniumpy.session), 308

xml\_pretty() (in module pytan.utils), 287

xml\_pretty\_resultobj() (in module pytan.utils), 288

xml\_pretty\_resultxml() (in module pytan.utils), 288

XmlError (class in taniumpy.object\_types.xml\_error), 320

xmldict (module), 320