

---

# PyTan Documentation

*Release 1.0.1*

**Jim Olsen**

December 08, 2014



## CONTENTS

<b>1</b>	<b>Table of Contents</b>	<b>1</b>
1.1	Description . . . . .	1
1.2	Why it was created . . . . .	1
1.3	Requirements . . . . .	1
1.4	Installation . . . . .	2
1.5	Usage . . . . .	2
1.6	Directory Layout . . . . .	2
1.7	pytan package . . . . .	3
1.8	taniumpy package . . . . .	274
1.9	xmldict module . . . . .	288
1.10	ddt module . . . . .	289
1.11	threaded_http module . . . . .	290
<b>2</b>	<b>Indices and tables</b>	<b>291</b>
	<b>Python Module Index</b>	<b>293</b>
	<b>Index</b>	<b>295</b>



## TABLE OF CONTENTS

### 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.

### 1.2 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.

### 1.3 Requirements

- Python 2.7
- A working install of Tanium Server 6.2

## 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

## 1.5 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*

## 1.6 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/`

## 1.7 pytan package

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

```
pytan.__version__ = '1.0.1'
    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
```

### 1.7.1 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  # Path to lib directory which contains pytan package
2  PYTAN_LIB_PATH = '../lib'
3
4  # connection info for Tanium Server
5  USERNAME = "Tanium User"
6  PASSWORD = "T@n!um"
7  HOST = "172.16.31.128"
8  PORT = "444"
9
10 # Logging conrols
11 LOGLEVEL = 2
12 DEBUGFORMAT = False
13
14 import sys, tempfile
15 sys.path.append(PYTAN_LIB_PATH)
16
17 import pytan
18 handler = pytan.Handler(
19     username=USERNAME,
20     password=PASSWORD,
21     host=HOST,
22     port=PORT,
23     loglevel=LOGLEVEL,
24     debugformat=DEBUGFORMAT,
25 )
26
27 print handler
```

## Output from Python Code

```
1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
```

## 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  # Path to lib directory which contains pytan package
2  PYTAN_LIB_PATH = '../lib'
3
4  # connection info for Tanium Server
5  USERNAME = "Tanium User"
6  PASSWORD = "T@n!um"
7  HOST = "172.16.31.128"
8  PORT = "444"
9
10 # Logging conrols
11 LOGLEVEL = 2
12 DEBUGFORMAT = False
13
14 import sys, tempfile
15 sys.path.append(PYTAN_LIB_PATH)
16
17 import pytan
18 handler = pytan.Handler(
19     username=USERNAME,
20     password=PASSWORD,
21     host=HOST,
22     port=PORT,
23     loglevel=LOGLEVEL,
24     debugformat=DEBUGFORMAT,
25 )
26
27 print handler
28
29 # setup the arguments for the handler method
30 kwargs = {}
31 kwargs["qtype"] = u'saved'
32 kwargs["name"] = [u'Installed Applications']
33
34 # call the handler with the ask method, passing in kwargs for arguments
35 response = handler.ask(**kwargs)
36 import pprint, io
37
38 print ""
39 print "Type of response: ", type(response)
40
41 print ""
42 print "Pretty print of response:"
```



```

43 print pprint.pformat(response)
44
45 print ""
46 print "Equivalent Question if it were to be asked in the Tanium Console: "
47 print response['question_object'].query_text
48
49 # create an IO stream to store CSV results to
50 out = io.BytesIO()
51
52 # call the write_csv() method to convert response to CSV and store it in out
53 response['question_results'].write_csv(out, response['question_results'])
54
55 print ""
56 print "CSV Results of response: "
57 out = out.getvalue()
58 if len(out.splitlines()) > 15:
59     out = out.splitlines()[0:15]
60     out.append('..trimmed for brevity..')
61     out = '\n'.join(out)
62 print out

```

### Output from Python Code

```

1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
2 2014-12-08 16:19:51,121 INFO      question_progress: Results 60000% (Get Installed Applications from
3
4 Type of response: <type 'dict'>
5
6 Pretty print of response:
7 {'question_object': <taniumpy.object_types.saved_question.SavedQuestion object at 0x10212e290>,
8  'question_results': <taniumpy.object_types.result_set.ResultSet object at 0x102b1fa90>}
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 Count,Name,Silent Uninstall String,Uninstallable,Version
15 714,[too many results],None,None,None
16 1,update-manager-core,nothing,Not Uninstallable,1:0.196.12
17 1,libminiupnpc8,nothing,Not Uninstallable,1.6-3ubuntu2.14.04.1
18 1,iso-codes,nothing,Not Uninstallable,3.52-1
19 1,docbook-dtds,nothing,Not Uninstallable,1.0
20 1,libexttextcat-2.0-0,nothing,Not Uninstallable,3.4.3-1ubuntu1
21 1,Google Search,nothing,Not Uninstallable,37.0.2062.120
22 1,gnome-user-share,nothing,Not Uninstallable,2.28.2
23 1,libblkid1:amd64,nothing,Not Uninstallable,2.20.1-5.1ubuntu20.1
24 1,fipscheck-lib,nothing,Not Uninstallable,1.2.0
25 1,gsm,nothing,Not Uninstallable,1.0.13
26 1,VoiceOver Quickstart,nothing,Not Uninstallable,6.0
27 1,VoiceOver Utility,nothing,Not Uninstallable,6.0
28 1,growisofs,nothing,Not Uninstallable,7.1-10build1
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  # Path to lib directory which contains pytan package
2  PYTAN_LIB_PATH = '../lib'
3
4  # connection info for Tanium Server
5  USERNAME = "Tanium User"
6  PASSWORD = "T@n!um"
7  HOST = "172.16.31.128"
8  PORT = "444"
9
10 # Logging conrols
11 LOGLEVEL = 2
12 DEBUGFORMAT = False
13
14 import sys, tempfile
15 sys.path.append(PYTAN_LIB_PATH)
16
17 import pytan
18 handler = pytan.Handler(
19     username=USERNAME,
20     password=PASSWORD,
21     host=HOST,
22     port=PORT,
23     loglevel=LOGLEVEL,
24     debugformat=DEBUGFORMAT,
25 )
26
27 print handler
28
29 # setup the arguments for the handler method
30 kwargs = {}
31 kwargs["qtype"] = u'saved'
32 kwargs["name"] = u'Installed Applications'
33
34 # call the handler with the ask method, passing in kwargs for arguments
35 response = handler.ask(**kwargs)
36 import pprint, io
37
38 print ""
39 print "Type of response: ", type(response)
40
41 print ""
42 print "Pretty print of response:"
43 print pprint.pformat(response)
44
45 print ""
46 print "Equivalent Question if it were to be asked in the Tanium Console: "
47 print response['question_object'].query_text
48
49 # create an IO stream to store CSV results to
50 out = io.BytesIO()
51
52 # call the write_csv() method to convert response to CSV and store it in out
```

```

53 response['question_results'].write_csv(out, response['question_results'])
54
55 print ""
56 print "CSV Results of response: "
57 out = out.getvalue()
58 if len(out.splitlines()) > 15:
59     out = out.splitlines()[0:15]
60     out.append('..trimmed for brevity..')
61     out = '\n'.join(out)
62 print out

```

### Output from Python Code

```

1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
2 2014-12-08 16:19:51,778 INFO      question_progress: Results 60000% (Get Installed Applications from
3
4 Type of response:  <type 'dict'>
5
6 Pretty print of response:
7 {'question_object': <taniumpy.object_types.saved_question.SavedQuestion object at 0x102b24790>,
8  'question_results': <taniumpy.object_types.result_set.ResultSet object at 0x1029e7fd0>}
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 Count,Name,Silent Uninstall String,Uninstallable,Version
15 714,[too many results],None,None,None
16 1,update-manager-core,nothing,Not Uninstallable,1:0.196.12
17 1,libminiupnpc8,nothing,Not Uninstallable,1.6-3ubuntu2.14.04.1
18 1,iso-codes,nothing,Not Uninstallable,3.52-1
19 1,docbook-dtds,nothing,Not Uninstallable,1.0
20 1,libexttextcat-2.0-0,nothing,Not Uninstallable,3.4.3-1ubuntu1
21 1,Google Search,nothing,Not Uninstallable,37.0.2062.120
22 1,gnome-user-share,nothing,Not Uninstallable,2.28.2
23 1,libblkid1:amd64,nothing,Not Uninstallable,2.20.1-5.1ubuntu20.1
24 1,fipscheck-lib,nothing,Not Uninstallable,1.2.0
25 1,gsm,nothing,Not Uninstallable,1.0.13
26 1,VoiceOver Quickstart,nothing,Not Uninstallable,6.0
27 1,VoiceOver Utility,nothing,Not Uninstallable,6.0
28 1,growisofs,nothing,Not Uninstallable,7.1-10build1
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 # Path to lib directory which contains pytan package
2 PYTAN_LIB_PATH = '../lib'
3

```

```
4  # connection info for Tanium Server
5  USERNAME = "Tanium User"
6  PASSWORD = "T@n!um"
7  HOST = "172.16.31.128"
8  PORT = "444"
9
10 # Logging conrols
11 LOGLEVEL = 2
12 DEBUGFORMAT = False
13
14 import sys, tempfile
15 sys.path.append(PYTAN_LIB_PATH)
16
17 import pytan
18 handler = pytan.Handler(
19     username=USERNAME,
20     password=PASSWORD,
21     host=HOST,
22     port=PORT,
23     loglevel=LOGLEVEL,
24     debugformat=DEBUGFORMAT,
25 )
26
27 print handler
28
29 # setup the arguments for the handler method
30 kwargs = {}
31 kwargs["sensors"] = u'Computer Name'
32 kwargs["qtype"] = u'manual_human'
33
34 # call the handler with the ask method, passing in kwargs for arguments
35 response = handler.ask(**kwargs)
36 import pprint, io
37
38 print ""
39 print "Type of response: ", type(response)
40
41 print ""
42 print "Pretty print of response:"
43 print pprint.pformat(response)
44
45 print ""
46 print "Equivalent Question if it were to be asked in the Tanium Console: "
47 print response['question_object'].query_text
48
49 # create an IO stream to store CSV results to
50 out = io.BytesIO()
51
52 # call the write_csv() method to convert response to CSV and store it in out
53 response['question_results'].write_csv(out, response['question_results'])
54
55 print ""
56 print "CSV Results of response: "
57 out = out.getvalue()
58 if len(out.splitlines()) > 15:
59     out = out.splitlines()[0:15]
60     out.append('..trimmed for brevity..')
61     out = '\n'.join(out)
```

```
62 print out
```

### Output from Python Code

```
1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
2 2014-12-08 16:19:52,403 INFO      question_progress: Results 0% (Get Computer Name from all machines)
3 2014-12-08 16:19:57,417 INFO      question_progress: Results 67% (Get Computer Name from all machines)
4 2014-12-08 16:20:02,434 INFO      question_progress: Results 100% (Get Computer Name from all machines)
5
6 Type of response: <type 'dict'>
7
8 Pretty print of response:
9 {'question_object': <taniumpy.object_types.question.Question object at 0x1021b6dd0>,
10  'question_results': <taniumpy.object_types.result_set.ResultSet object at 0x1029fe610>}
11
12 Equivalent Question if it were to be asked in the Tanium Console:
13 Get Computer Name from all machines
14
15 CSV Results of response:
16 Computer Name
17 Casus-Belli.local
18 jtanium1.localdomain
19 ubuntu.(none)
20 localhost.(none)
21 Jims-Mac.local
22 WIN-A12SC6N6T7Q
```

### 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 # Path to lib directory which contains pytan package
2 PYTAN_LIB_PATH = '../lib'
3
4 # connection info for Tanium Server
5 USERNAME = "Tanium User"
6 PASSWORD = "T@n!um"
7 HOST = "172.16.31.128"
8 PORT = "444"
9
10 # Logging conrols
11 LOGLEVEL = 2
12 DEBUGFORMAT = False
13
14 import sys, tempfile
15 sys.path.append(PYTAN_LIB_PATH)
16
17 import pytan
18 handler = pytan.Handler(
19     username=USERNAME,
```

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

### Output from Python Code

```
1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
2 2014-12-08 16:20:02,573 INFO      question_progress: Results 0% (Get Computer Name and Installed Appl
3 2014-12-08 16:20:07,596 INFO      question_progress: Results 67% (Get Computer Name and Installed Appl
4 2014-12-08 16:20:12,620 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 0x1029f7390>,
10  'question_results': <taniumpy.object_types.result_set.ResultSet object at 0x1028dfa10>}
```

```

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 MakePDF
19 Wish
20 Time Machine
21 AppleGraphicsWarning
22 soagent
23 SpeechService
24 AinuIM
25 Pass Viewer
26 PressAndHold
27 PluginIM
28 UserNotificationCenter
29 FaceTime
30 ScreenSaverEngine
31 ..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  # Path to lib directory which contains pytan package
2  PYTAN_LIB_PATH = '../lib'
3
4  # connection info for Tanium Server
5  USERNAME = "Tanium User"
6  PASSWORD = "T@n!um"
7  HOST = "172.16.31.128"
8  PORT = "444"
9
10 # Logging conrols
11 LOGLEVEL = 2
12 DEBUGFORMAT = False
13
14 import sys, tempfile
15 sys.path.append(PYTAN_LIB_PATH)
16
17 import pytan
18 handler = pytan.Handler(
19     username=USERNAME,
20     password=PASSWORD,
21     host=HOST,
22     port=PORT,
23     loglevel=LOGLEVEL,
24     debugformat=DEBUGFORMAT,
25 )
26

```

```
27 print handler
28
29 # setup the arguments for the handler method
30 kwargs = {}
31 kwargs["sensors"] = [u'name:Computer Name', u'name:Installed Applications']
32 kwargs["qtype"] = u'manual_human'
33
34 # call the handler with the ask method, passing in kwargs for arguments
35 response = handler.ask(**kwargs)
36 import pprint, io
37
38 print ""
39 print "Type of response: ", type(response)
40
41 print ""
42 print "Pretty print of response:"
43 print pprint.pformat(response)
44
45 print ""
46 print "Equivalent Question if it were to be asked in the Tanium Console: "
47 print response['question_object'].query_text
48
49 # create an IO stream to store CSV results to
50 out = io.BytesIO()
51
52 # call the write_csv() method to convert response to CSV and store it in out
53 response['question_results'].write_csv(out, response['question_results'])
54
55 print ""
56 print "CSV Results of response: "
57 out = out.getvalue()
58 if len(out.splitlines()) > 15:
59     out = out.splitlines()[0:15]
60     out.append('..trimmed for brevity..')
61     out = '\n'.join(out)
62 print out
```

### Output from Python Code

```
1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
2 2014-12-08 16:20:12,928 INFO      question_progress: Results 0% (Get Computer Name and Installed Appl
3 2014-12-08 16:20:17,953 INFO      question_progress: Results 83% (Get Computer Name and Installed App
4 2014-12-08 16:20:22,978 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 0x1028e2950>,
10  'question_results': <taniumpy.object_types.result_set.ResultSet object at 0x102b6f550>}
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 MakePDF
```



```

19 Wish
20 Time Machine
21 AppleGraphicsWarning
22 soagent
23 SpeechService
24 AinuIM
25 Pass Viewer
26 PressAndHold
27 PluginIM
28 UserNotificationCenter
29 FaceTime
30 ScreenSaverEngine
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  # Path to lib directory which contains pytan package
2  PYTAN_LIB_PATH = '../lib'
3
4  # connection info for Tanium Server
5  USERNAME = "Tanium User"
6  PASSWORD = "T@n!um"
7  HOST = "172.16.31.128"
8  PORT = "444"
9
10 # Logging conrols
11 LOGLEVEL = 2
12 DEBUGFORMAT = False
13
14 import sys, tempfile
15 sys.path.append(PYTAN_LIB_PATH)
16
17 import pytan
18 handler = pytan.Handler(
19     username=USERNAME,
20     password=PASSWORD,
21     host=HOST,
22     port=PORT,
23     loglevel=LOGLEVEL,
24     debugformat=DEBUGFORMAT,
25 )
26
27 print handler
28
29 # setup the arguments for the handler method
30 kwargs = {}
31 kwargs["sensors"] = u'Folder Name Search with RegEx Match(dirname=Program Files,regex=Microsoft.*)'

```

```

32 kwargs["qtype"] = u'manual_human'
33
34 # call the handler with the ask method, passing in kwargs for arguments
35 response = handler.ask(**kwargs)
36 import pprint, io
37
38 print ""
39 print "Type of response: ", type(response)
40
41 print ""
42 print "Pretty print of response:"
43 print pprint.pformat(response)
44
45 print ""
46 print "Equivalent Question if it were to be asked in the Tanium Console: "
47 print response['question_object'].query_text
48
49 # create an IO stream to store CSV results to
50 out = io.BytesIO()
51
52 # call the write_csv() method to convert response to CSV and store it in out
53 response['question_results'].write_csv(out, response['question_results'])
54
55 print ""
56 print "CSV Results of response: "
57 out = out.getvalue()
58 if len(out.splitlines()) > 15:
59     out = out.splitlines()[0:15]
60     out.append('..trimmed for brevity..')
61     out = '\n'.join(out)
62 print out

```

### Output from Python Code

```

1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
2 2014-12-08 16:20:23,296 INFO      question_progress: Results 0% (Get Folder Name Search with RegEx Ma
3 2014-12-08 16:20:28,314 INFO      question_progress: Results 0% (Get Folder Name Search with RegEx Ma
4 2014-12-08 16:20:33,331 INFO      question_progress: Results 0% (Get Folder Name Search with RegEx Ma
5 2014-12-08 16:20:38,351 INFO      question_progress: Results 0% (Get Folder Name Search with RegEx Ma
6 2014-12-08 16:20:43,372 INFO      question_progress: Results 17% (Get Folder Name Search with RegEx M
7 2014-12-08 16:20:48,394 INFO      question_progress: Results 83% (Get Folder Name Search with RegEx M
8 2014-12-08 16:20:53,410 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 0x102b6fd90>,
14  'question_results': <taniumpy.object_types.result_set.ResultSet object at 0x102bb1810>}
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, ] from all machines
18
19 CSV Results of response:
20 Count,"Folder Name Search with RegEx Match[No, Program Files, No, ]"
21 1,C:\Program Files\tanium\tanium server\ApacheBackup2014-09-16-20-44-23\cgi-bin
22 2,C:\Program Files\VMware\VMware Tools\plugins\vmtoolsd
23 1,C:\Program Files\Microsoft SQL Server\110\Setup Bootstrap\SQLServer2012\1040_ITA_LP\64\1040\help

```

```

24 1,C:\Program Files\Common Files\Microsoft Shared\VS7Debug
25 1,C:\Program Files\Tanium\Tanium Server\Apache24\manual\style
26 1,C:\Program Files\Tanium\Tanium Server\Apache24\htdocs\console\history
27 2,C:\Program Files\Common Files\VMware\Drivers\vmci\sockets\include
28 2,C:\Program Files\Common Files\Microsoft Shared\ink\ar-SA
29 1,C:\Program Files\Tanium\Tanium Server\plugins\console\Dashboards
30 1,C:\Program Files\Tanium\Tanium Server\CertificateBackup2014-11-17-11-17-33
31 2,C:\Program Files\Common Files\SpeechEngines\Microsoft
32 1,C:\Program Files\Tanium\Tanium Server\ApacheBackup2014-09-16-20-44-23\modules
33 2,C:\Program Files\Common Files\Microsoft Shared\ink\ru-RU
34 1,C:\Program Files\Microsoft SQL Server\110\DTS\ForEachEnumerators\en
35 ..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  # Path to lib directory which contains pytan package
2  PYTAN_LIB_PATH = '../lib'
3
4  # connection info for Tanium Server
5  USERNAME = "Tanium User"
6  PASSWORD = "T@n!um"
7  HOST = "172.16.31.128"
8  PORT = "444"
9
10 # Logging conrols
11 LOGLEVEL = 2
12 DEBUGFORMAT = False
13
14 import sys, tempfile
15 sys.path.append(PYTAN_LIB_PATH)
16
17 import pytan
18 handler = pytan.Handler(
19     username=USERNAME,
20     password=PASSWORD,
21     host=HOST,
22     port=PORT,
23     loglevel=LOGLEVEL,
24     debugformat=DEBUGFORMAT,
25 )
26
27 print handler
28
29 # setup the arguments for the handler method
30 kwargs = {}
31 kwargs["sensors"] = u'Computer Name{fake=Dweedle}'
32 kwargs["qtype"] = u'manual_human'
33
34 # call the handler with the ask method, passing in kwargs for arguments

```

```
35 response = handler.ask(**kwargs)
36 import pprint, io
37
38 print ""
39 print "Type of response: ", type(response)
40
41 print ""
42 print "Pretty print of response:"
43 print pprint.pformat(response)
44
45 print ""
46 print "Equivalent Question if it were to be asked in the Tanium Console: "
47 print response['question_object'].query_text
48
49 # create an IO stream to store CSV results to
50 out = io.BytesIO()
51
52 # call the write_csv() method to convert response to CSV and store it in out
53 response['question_results'].write_csv(out, response['question_results'])
54
55 print ""
56 print "CSV Results of response: "
57 out = out.getvalue()
58 if len(out.splitlines()) > 15:
59     out = out.splitlines()[0:15]
60     out.append('..trimmed for brevity..')
61     out = '\n'.join(out)
62 print out
```

### Output from Python Code

```
1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
2 2014-12-08 16:20:53,557 INFO      question_progress: Results 0% (Get Computer Name from all machines)
3 2014-12-08 16:20:58,574 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 0x102b82450>,
9  'question_results': <taniumpy.object_types.result_set.ResultSet object at 0x1029f70d0>}
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 jtanium1.localdomain
18 ubuntu.(none)
19 localhost.(none)
20 Jims-Mac.local
21 WIN-A12SC6N6T7Q
```

### 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  # Path to lib directory which contains pytan package
2  PYTAN_LIB_PATH = '../lib'
3
4  # connection info for Tanium Server
5  USERNAME = "Tanium User"
6  PASSWORD = "T@n!um"
7  HOST = "172.16.31.128"
8  PORT = "444"
9
10 # Logging conrols
11 LOGLEVEL = 2
12 DEBUGFORMAT = False
13
14 import sys, tempfile
15 sys.path.append(PYTAN_LIB_PATH)
16
17 import pytan
18 handler = pytan.Handler(
19     username=USERNAME,
20     password=PASSWORD,
21     host=HOST,
22     port=PORT,
23     loglevel=LOGLEVEL,
24     debugformat=DEBUGFORMAT,
25 )
26
27 print handler
28
29 # setup the arguments for the handler method
30 kwargs = {}
31 kwargs["sensors"] = [u'Folder Name Search with RegEx Match{dirname=Program Files,regex=Microsoft.*}',
32     u'Computer Name']
33 kwargs["qtype"] = u'manual_human'
34
35 # call the handler with the ask method, passing in kwargs for arguments
36 response = handler.ask(**kwargs)
37 import pprint, io
38
39 print ""
40 print "Type of response: ", type(response)
41
42 print ""
43 print "Pretty print of response:"
44 print pprint.pformat(response)
45
46 print ""
47 print "Equivalent Question if it were to be asked in the Tanium Console: "

```

```

48 print response['question_object'].query_text
49
50 # create an IO stream to store CSV results to
51 out = io.BytesIO()
52
53 # call the write_csv() method to convert response to CSV and store it in out
54 response['question_results'].write_csv(out, response['question_results'])
55
56 print ""
57 print "CSV Results of response: "
58 out = out.getvalue()
59 if len(out.splitlines()) > 15:
60     out = out.splitlines()[0:15]
61     out.append('..trimmed for brevity..')
62     out = '\n'.join(out)
63 print out

```

### Output from Python Code

```

1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
2 2014-12-08 16:20:58,703 INFO question_progress: Results 0% (Get Computer Name from all machines)
3 2014-12-08 16:21:03,720 INFO question_progress: Results 0% (Get Computer Name from all machines)
4 2014-12-08 16:21:08,737 INFO question_progress: Results 0% (Get Computer Name from all machines)
5 2014-12-08 16:21:13,757 INFO question_progress: Results 0% (Get Computer Name from all machines)
6 2014-12-08 16:21:18,777 INFO question_progress: Results 0% (Get Computer Name from all machines)
7 2014-12-08 16:21:23,793 INFO question_progress: Results 0% (Get Computer Name from all machines)
8 2014-12-08 16:21:28,811 INFO question_progress: Results 0% (Get Computer Name from all machines)
9 2014-12-08 16:21:33,827 INFO question_progress: Results 0% (Get Computer Name from all machines)
10 2014-12-08 16:21:38,843 INFO question_progress: Results 0% (Get Computer Name from all machines)
11 2014-12-08 16:21:43,863 INFO question_progress: Results 0% (Get Computer Name from all machines)
12 2014-12-08 16:21:48,878 INFO question_progress: Results 0% (Get Computer Name from all machines)
13 2014-12-08 16:21:53,901 INFO question_progress: Results 0% (Get Computer Name from all machines)
14 2014-12-08 16:21:58,915 INFO question_progress: Results 0% (Get Computer Name from all machines)
15 2014-12-08 16:22:03,935 INFO question_progress: Results 33% (Get Computer Name from all machines)
16 2014-12-08 16:22:08,954 INFO question_progress: Results 100% (Get Computer Name from all machine
17
18 Type of response: <type 'dict'>
19
20 Pretty print of response:
21 {'question_object': <taniumpy.object_types.question.Question object at 0x102129990>,
22  'question_results': <taniumpy.object_types.result_set.ResultSet object at 0x102967e10>}
23
24 Equivalent Question if it were to be asked in the Tanium Console:
25 Get Computer Name from all machines
26
27 CSV Results of response:
28 Computer Name,"Folder Name Search with RegEx Match[No, Program Files, No, ]"
29 Casus-Belli.local,Windows Only
30 ubuntu.(none),Windows Only
31 localhost.(none),Windows Only
32 Jims-Mac.local,Windows Only
33 jtanium1.localdomain,"C:\Program Files\tanium\tanium server\ApacheBackup2014-09-16-20-44-23\cgi-bin
34 C:\Program Files\VMware\VMware Tools\plugins\vmtoolsd
35 C:\Program Files\Microsoft SQL Server\110\Setup Bootstrap\SQLServer2012\1040_ITA_LP\x64\1040\help
36 C:\Program Files\Common Files\Microsoft Shared\VS7Debug
37 C:\Program Files\tanium\tanium server\Apache24>manual\style
38 C:\Program Files\tanium\tanium server\Apache24\htdocs\console\history

```

```

39 C:\Program Files\Common Files\VMware\Drivers\vmci\sockets\include
40 C:\Program Files\Common Files\Microsoft Shared\ink\ar-SA
41 C:\Program Files\Tanium\Tanium Server\plugins\console\Dashboards
42 C:\Program Files\Tanium\Tanium Server\CertificateBackup2014-11-17-11-17-33
43 ..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  # Path to lib directory which contains pytan package
2  PYTAN_LIB_PATH = '../lib'
3
4  # connection info for Tanium Server
5  USERNAME = "Tanium User"
6  PASSWORD = "T@n!um"
7  HOST = "172.16.31.128"
8  PORT = "444"
9
10 # Logging conrols
11 LOGLEVEL = 2
12 DEBUGFORMAT = False
13
14 import sys, tempfile
15 sys.path.append(PYTAN_LIB_PATH)
16
17 import pytan
18 handler = pytan.Handler(
19     username=USERNAME,
20     password=PASSWORD,
21     host=HOST,
22     port=PORT,
23     loglevel=LOGLEVEL,
24     debugformat=DEBUGFORMAT,
25 )
26
27 print handler
28
29 # setup the arguments for the handler method
30 kwargs = {}
31 kwargs["sensors"] = u'Folder Name Search with RegEx Match'
32 kwargs["qtype"] = u'manual_human'
33
34 # call the handler with the ask method, passing in kwargs for arguments
35 response = handler.ask(**kwargs)
36 import pprint, io
37
38 print ""
39 print "Type of response: ", type(response)

```

```

40
41 print ""
42 print "Pretty print of response:"
43 print pprint.pformat(response)
44
45 print ""
46 print "Equivalent Question if it were to be asked in the Tanium Console: "
47 print response['question_object'].query_text
48
49 # create an IO stream to store CSV results to
50 out = io.BytesIO()
51
52 # call the write_csv() method to convert response to CSV and store it in out
53 response['question_results'].write_csv(out, response['question_results'])
54
55 print ""
56 print "CSV Results of response: "
57 out = out.getvalue()
58 if len(out.splitlines()) > 15:
59     out = out.splitlines()[0:15]
60     out.append('..trimmed for brevity..')
61     out = '\n'.join(out)
62 print out

```

### Output from Python Code

```

1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
2 2014-12-08 16:22:09,102 INFO question_progress: Results 0% (Get Folder Name Search with RegEx Ma
3 2014-12-08 16:22:14,122 INFO question_progress: Results 0% (Get Folder Name Search with RegEx Ma
4 2014-12-08 16:22:19,141 INFO question_progress: Results 0% (Get Folder Name Search with RegEx Ma
5 2014-12-08 16:22:24,157 INFO question_progress: Results 0% (Get Folder Name Search with RegEx Ma
6 2014-12-08 16:22:29,173 INFO question_progress: Results 0% (Get Folder Name Search with RegEx Ma
7 2014-12-08 16:22:34,194 INFO question_progress: Results 0% (Get Folder Name Search with RegEx Ma
8 2014-12-08 16:22:39,212 INFO question_progress: Results 0% (Get Folder Name Search with RegEx Ma
9 2014-12-08 16:22:44,230 INFO question_progress: Results 0% (Get Folder Name Search with RegEx Ma
10 2014-12-08 16:22:49,247 INFO question_progress: Results 0% (Get Folder Name Search with RegEx Ma
11 2014-12-08 16:22:54,270 INFO question_progress: Results 33% (Get Folder Name Search with RegEx M
12 2014-12-08 16:22:59,299 INFO question_progress: Results 67% (Get Folder Name Search with RegEx M
13 2014-12-08 16:23:04,324 INFO question_progress: Results 67% (Get Folder Name Search with RegEx M
14 2014-12-08 16:23:09,344 INFO question_progress: Results 83% (Get Folder Name Search with RegEx M
15 2014-12-08 16:23:14,368 INFO question_progress: Results 100% (Get Folder Name Search with RegEx
16
17 Type of response: <type 'dict'>
18
19 Pretty print of response:
20 {'question_object': <taniumpy.object_types.question.Question object at 0x1021a5ad0>,
21  'question_results': <taniumpy.object_types.result_set.ResultSet object at 0x102b2e110>}
22
23 Equivalent Question if it were to be asked in the Tanium Console:
24 Get Folder Name Search with RegEx Match[No, , No, ] from all machines
25
26 CSV Results of response:
27 Count,"Folder Name Search with RegEx Match[No, , No, ]"
28 40981,[too many results]
29 2,C:\Windows\winsxs\amd64_microsoft-windows-s..structure.resources_31bf3856ad364e35_6.1.7600.16385_e
30 2,C:\Windows\winsxs\x86_microsoft-windows-e..-host-authenticator_31bf3856ad364e35_6.1.7601.17514_non
31 1,C:\Windows\winsxs\amd64_microsoft-windows-ocspsvc_31bf3856ad364e35_6.1.7601.22807_nope_3bfeae72930

```



```

32 1,C:\Windows\winsxs\amd64_microsoft-windows-c..ityclient.resources_31bf3856ad364e35_6.1.7601.22865_en-u
33 1,C:\Program Files (x86)\Tanium\Tanium Client\Downloads\Action_192\grep\share\locale\bg\LC_MESSAGES
34 2,C:\Windows\assembly\NativeImages_v2.0.50727_64\System.Xml
35 1,C:\Users\Jim Olsen\Desktop\SysinternalsSuite
36 2,C:\Windows\winsxs\amd64_microsoft-windows-scripting.resources_31bf3856ad364e35_6.1.7600.16385_en-u
37 2,C:\Windows\winsxs\x86_microsoft-windows-mlang.resources_31bf3856ad364e35_6.1.7600.16385_ru-ru_cf3a
38 1,C:\Windows\winsxs\x86_microsoft-windows-directshow-dvdsupport_31bf3856ad364e35_6.1.7601.21987_none
39 1,C:\Windows\winsxs\amd64_microsoft-windows-ie-internetexplorer_31bf3856ad364e35_11.2.9600.17041_non
40 1,C:\Users\Jim Olsen\AppData\Local\Google
41 2,C:\Windows\winsxs\x86_microsoft-windows-e..nt-client.resources_31bf3856ad364e35_6.1.7600.16385_en-
42 ..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  # Path to lib directory which contains pytan package
2  PYTAN_LIB_PATH = '../lib'
3
4  # connection info for Tanium Server
5  USERNAME = "Tanium User"
6  PASSWORD = "T@n!um"
7  HOST = "172.16.31.128"
8  PORT = "444"
9
10 # Logging conrols
11 LOGLEVEL = 2
12 DEBUGFORMAT = False
13
14 import sys, tempfile
15 sys.path.append(PYTAN_LIB_PATH)
16
17 import pytan
18 handler = pytan.Handler(
19     username=USERNAME,
20     password=PASSWORD,
21     host=HOST,
22     port=PORT,
23     loglevel=LOGLEVEL,
24     debugformat=DEBUGFORMAT,
25 )
26
27 print handler
28
29 # setup the arguments for the handler method
30 kwargs = {}
31 kwargs["sensors"] = u'Folder Name Search with RegEx Match(dirname=Program Files,regex=Microsoft.*)',
32 kwargs["qtype"] = u'manual_human'
33

```

```

34 # call the handler with the ask method, passing in kwargs for arguments
35 response = handler.ask(**kwargs)
36 import pprint, io
37
38 print ""
39 print "Type of response: ", type(response)
40
41 print ""
42 print "Pretty print of response:"
43 print pprint.pformat(response)
44
45 print ""
46 print "Equivalent Question if it were to be asked in the Tanium Console: "
47 print response['question_object'].query_text
48
49 # create an IO stream to store CSV results to
50 out = io.BytesIO()
51
52 # call the write_csv() method to convert response to CSV and store it in out
53 response['question_results'].write_csv(out, response['question_results'])
54
55 print ""
56 print "CSV Results of response: "
57 out = out.getvalue()
58 if len(out.splitlines()) > 15:
59     out = out.splitlines()[0:15]
60     out.append('..trimmed for brevity..')
61     out = '\n'.join(out)
62 print out

```

## Output from Python Code

```

1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
2 2014-12-08 16:23:14,712 INFO question_progress: Results 0% (Get Folder Name Search with RegEx Ma
3 2014-12-08 16:23:19,729 INFO question_progress: Results 0% (Get Folder Name Search with RegEx Ma
4 2014-12-08 16:23:24,747 INFO question_progress: Results 0% (Get Folder Name Search with RegEx Ma
5 2014-12-08 16:23:29,765 INFO question_progress: Results 0% (Get Folder Name Search with RegEx Ma
6 2014-12-08 16:23:34,783 INFO question_progress: Results 0% (Get Folder Name Search with RegEx Ma
7 2014-12-08 16:23:39,800 INFO question_progress: Results 0% (Get Folder Name Search with RegEx Ma
8 2014-12-08 16:23:44,820 INFO question_progress: Results 0% (Get Folder Name Search with RegEx Ma
9 2014-12-08 16:23:49,838 INFO question_progress: Results 0% (Get Folder Name Search with RegEx Ma
10 2014-12-08 16:23:54,858 INFO question_progress: Results 0% (Get Folder Name Search with RegEx Ma
11 2014-12-08 16:23:59,886 INFO question_progress: Results 0% (Get Folder Name Search with RegEx Ma
12 2014-12-08 16:24:04,902 INFO question_progress: Results 0% (Get Folder Name Search with RegEx Ma
13 2014-12-08 16:24:09,919 INFO question_progress: Results 0% (Get Folder Name Search with RegEx Ma
14 2014-12-08 16:24:14,940 INFO question_progress: Results 0% (Get Folder Name Search with RegEx Ma
15 2014-12-08 16:24:19,957 INFO question_progress: Results 0% (Get Folder Name Search with RegEx Ma
16 2014-12-08 16:24:24,975 INFO question_progress: Results 50% (Get Folder Name Search with RegEx M
17 2014-12-08 16:24:29,990 INFO question_progress: Results 83% (Get Folder Name Search with RegEx M
18 2014-12-08 16:24:35,009 INFO question_progress: Results 83% (Get Folder Name Search with RegEx M
19 2014-12-08 16:24:40,028 INFO question_progress: Results 100% (Get Folder Name Search with RegEx
20
21 Type of response: <type 'dict'>
22
23 Pretty print of response:
24 {'question_object': <taniumpy.object_types.question.Question object at 0x102b34650>,
25  'question_results': <taniumpy.object_types.result_set.ResultSet object at 0x102116e50>}

```

```

26
27 Equivalent Question if it were to be asked in the Tanium Console:
28 Get Folder Name Search with RegEx Match[No, Program Files, No, ] contains "Shared" from all machines
29
30 CSV Results of response:
31 Count,"Folder Name Search with RegEx Match[No, Program Files, No, ]"
32 4,[no results]
33 1,C:\Program Files\Common Files\Microsoft Shared\VS7Debug
34 2,C:\Program Files\Common Files\Microsoft Shared\ink\ar-SA
35 2,C:\Program Files\Common Files\Microsoft Shared\ink\ru-RU
36 2,C:\Program Files\Common Files\Microsoft Shared\ink\fsdefinitions\keypad
37 2,C:\Program Files\Common Files\Microsoft Shared\ink
38 2,C:\Program Files\Common Files\Microsoft Shared\ink\sv-SE
39 2,C:\Program Files\Common Files\Microsoft Shared\ink\uk-UA
40 2,C:\Program Files\Common Files\Microsoft Shared\ink\sl-SI
41 2,C:\Program Files\Common Files\Microsoft Shared\ink\hu-HU
42 2,C:\Program Files\Common Files\Microsoft Shared\ink\zh-TW
43 2,C:\Program Files\Common Files\Microsoft Shared\ink\zh-CN
44 2,C:\Program Files\Common Files\Microsoft Shared\ink\fi-FI
45 2,C:\Program Files\Common Files\Microsoft Shared
46 ..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  # Path to lib directory which contains pytan package
2  PYTAN_LIB_PATH = '../lib'
3
4  # connection info for Tanium Server
5  USERNAME = "Tanium User"
6  PASSWORD = "T@n!um"
7  HOST = "172.16.31.128"
8  PORT = "444"
9
10 # Logging conrols
11 LOGLEVEL = 2
12 DEBUGFORMAT = False
13
14 import sys, tempfile
15 sys.path.append(PYTAN_LIB_PATH)
16
17 import pytan
18 handler = pytan.Handler(
19     username=USERNAME,
20     password=PASSWORD,

```

```

21     host=HOST,
22     port=PORT,
23     loglevel=LOGLEVEL,
24     debugformat=DEBUGFORMAT,
25 )
26
27 print handler
28
29 # setup the arguments for the handler method
30 kwargs = {}
31 kwargs["sensors"] = u'Operating System, that contains:Windows, opt:match_all_values, opt:ignore_case
32 kwargs["qtype"] = u'manual_human'
33
34 # call the handler with the ask method, passing in kwargs for arguments
35 response = handler.ask(**kwargs)
36 import pprint, io
37
38 print ""
39 print "Type of response: ", type(response)
40
41 print ""
42 print "Pretty print of response:"
43 print pprint.pformat(response)
44
45 print ""
46 print "Equivalent Question if it were to be asked in the Tanium Console: "
47 print response['question_object'].query_text
48
49 # create an IO stream to store CSV results to
50 out = io.BytesIO()
51
52 # call the write_csv() method to convert response to CSV and store it in out
53 response['question_results'].write_csv(out, response['question_results'])
54
55 print ""
56 print "CSV Results of response: "
57 out = out.getvalue()
58 if len(out.splitlines()) > 15:
59     out = out.splitlines()[0:15]
60     out.append('..trimmed for brevity..')
61     out = '\n'.join(out)
62 print out

```

### Output from Python Code

```

1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
2 2014-12-08 16:24:40,124 INFO      question_progress: Results 0% (Get Operating System contains "Wind
3 2014-12-08 16:24:45,141 INFO      question_progress: Results 50% (Get Operating System contains "Wind
4 2014-12-08 16:24:50,158 INFO      question_progress: Results 100% (Get Operating System contains "Win
5
6 Type of response:  <type 'dict'>
7
8 Pretty print of response:
9 {'question_object': <taniumpy.object_types.question.Question object at 0x102130210>,
10  'question_results': <taniumpy.object_types.result_set.ResultSet object at 0x1021236d0>}
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 | Count,Operating System
17 | 4,[no results]
18 | 2,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 | # Path to lib directory which contains pytan package
2 | PYTAN_LIB_PATH = '../lib'
3 |
4 | # connection info for Tanium Server
5 | USERNAME = "Tanium User"
6 | PASSWORD = "T@n!um"
7 | HOST = "172.16.31.128"
8 | PORT = "444"
9 |
10 | # Logging conrols
11 | LOGLEVEL = 2
12 | DEBUGFORMAT = False
13 |
14 | import sys, tempfile
15 | sys.path.append(PYTAN_LIB_PATH)
16 |
17 | import pytan
18 | handler = pytan.Handler(
19 |     username=USERNAME,
20 |     password=PASSWORD,
21 |     host=HOST,
22 |     port=PORT,
23 |     loglevel=LOGLEVEL,
24 |     debugformat=DEBUGFORMAT,
25 | )
26 |
27 | print handler
28 |
29 | # setup the arguments for the handler method
30 | kwargs = {}
31 | kwargs["sensors"] = u'Folder Name Search with RegEx Match(dirname=Program Files,regex=Microsoft.*)',
32 | kwargs["qtype"] = u'manual_human'
33 |
34 | # call the handler with the ask method, passing in kwargs for arguments
35 | response = handler.ask(**kwargs)
36 | import pprint, io
37 |

```

```

38 print ""
39 print "Type of response: ", type(response)
40
41 print ""
42 print "Pretty print of response:"
43 print pprint.pformat(response)
44
45 print ""
46 print "Equivalent Question if it were to be asked in the Tanium Console: "
47 print response['question_object'].query_text
48
49 # create an IO stream to store CSV results to
50 out = io.BytesIO()
51
52 # call the write_csv() method to convert response to CSV and store it in out
53 response['question_results'].write_csv(out, response['question_results'])
54
55 print ""
56 print "CSV Results of response: "
57 out = out.getvalue()
58 if len(out.splitlines()) > 15:
59     out = out.splitlines()[0:15]
60     out.append('..trimmed for brevity..')
61     out = '\n'.join(out)
62 print out

```

### Output from Python Code

```

1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
2 2014-12-08 16:24:50,266 INFO      question_progress: Results 0% (Get Folder Name Search with RegEx Ma
3 2014-12-08 16:24:55,284 INFO      question_progress: Results 0% (Get Folder Name Search with RegEx Ma
4 2014-12-08 16:25:00,307 INFO      question_progress: Results 17% (Get Folder Name Search with RegEx M
5 2014-12-08 16:25:05,326 INFO      question_progress: Results 67% (Get Folder Name Search with RegEx M
6 2014-12-08 16:25:10,344 INFO      question_progress: Results 83% (Get Folder Name Search with RegEx M
7 2014-12-08 16:25:15,369 INFO      question_progress: Results 100% (Get Folder Name Search with RegEx
8
9 Type of response:  <type 'dict'>
10
11 Pretty print of response:
12 {'question_object': <taniumpy.object_types.question.Question object at 0x102967b90>,
13  'question_results': <taniumpy.object_types.result_set.ResultSet object at 0x102b2fd50>}
14
15 Equivalent Question if it were to be asked in the Tanium Console:
16 Get Folder Name Search with RegEx Match[No, Program Files, No, ] contains "Shared" from all machines
17
18 CSV Results of response:
19 Count,"Folder Name Search with RegEx Match[No, Program Files, No, ]"
20 4,[no results]
21 1,C:\Program Files\Common Files\Microsoft Shared\VS7Debug
22 2,C:\Program Files\Common Files\Microsoft Shared\ink\ar-SA
23 2,C:\Program Files\Common Files\Microsoft Shared\ink\ru-RU
24 2,C:\Program Files\Common Files\Microsoft Shared\ink\fsdefinitions\keypad
25 2,C:\Program Files\Common Files\Microsoft Shared\ink
26 2,C:\Program Files\Common Files\Microsoft Shared\ink\sv-SE
27 2,C:\Program Files\Common Files\Microsoft Shared\ink\uk-UA
28 2,C:\Program Files\Common Files\Microsoft Shared\ink\sl-SI
29 2,C:\Program Files\Common Files\Microsoft Shared\ink\hu-HU

```

```

30 2,C:\Program Files\Common Files\Microsoft Shared\ink\zh-TW
31 2,C:\Program Files\Common Files\Microsoft Shared\ink\zh-CN
32 2,C:\Program Files\Common Files\Microsoft Shared\ink\fi-FI
33 2,C:\Program Files\Common Files\Microsoft Shared
34 ..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  # Path to lib directory which contains pytan package
2  PYTAN_LIB_PATH = '../lib'
3
4  # connection info for Tanium Server
5  USERNAME = "Tanium User"
6  PASSWORD = "T@n!um"
7  HOST = "172.16.31.128"
8  PORT = "444"
9
10 # Logging conrols
11 LOGLEVEL = 2
12 DEBUGFORMAT = False
13
14 import sys, tempfile
15 sys.path.append(PYTAN_LIB_PATH)
16
17 import pytan
18 handler = pytan.Handler(
19     username=USERNAME,
20     password=PASSWORD,
21     host=HOST,
22     port=PORT,
23     loglevel=LOGLEVEL,
24     debugformat=DEBUGFORMAT,
25 )
26
27 print handler
28
29 # setup the arguments for the handler method
30 kwargs = {}
31 kwargs["sensors"] = u'Operating System, that contains:Windows, opt:max_data_age:3600, opt:value_type
32 kwargs["qtype"] = u'manual_human'
33
34 # call the handler with the ask method, passing in kwargs for arguments
35 response = handler.ask(**kwargs)
36 import pprint, io

```

```
37
38 print ""
39 print "Type of response: ", type(response)
40
41 print ""
42 print "Pretty print of response:"
43 print pprint.pformat(response)
44
45 print ""
46 print "Equivalent Question if it were to be asked in the Tanium Console: "
47 print response['question_object'].query_text
48
49 # create an IO stream to store CSV results to
50 out = io.BytesIO()
51
52 # call the write_csv() method to convert response to CSV and store it in out
53 response['question_results'].write_csv(out, response['question_results'])
54
55 print ""
56 print "CSV Results of response: "
57 out = out.getvalue()
58 if len(out.splitlines()) > 15:
59     out = out.splitlines()[0:15]
60     out.append('..trimmed for brevity..')
61     out = '\n'.join(out)
62 print out
```

### Output from Python Code

```
1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
2 2014-12-08 16:25:15,477 INFO      question_progress: Results 0% (Get Operating System contains "Windows")
3 2014-12-08 16:25:20,494 INFO      question_progress: Results 67% (Get Operating System contains "Windows")
4 2014-12-08 16:25:25,510 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 0x102b1fbd0>,
10  'question_results': <taniumpy.object_types.result_set.ResultSet object at 0x1022e0bd0>}
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 Count,Operating System
17 4,[no results]
18 2,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  # Path to lib directory which contains pytan package
2  PYTAN_LIB_PATH = '../lib'
3
4  # connection info for Tanium Server
5  USERNAME = "Tanium User"
6  PASSWORD = "T@n!um"
7  HOST = "172.16.31.128"
8  PORT = "444"
9
10 # Logging conrols
11 LOGLEVEL = 2
12 DEBUGFORMAT = False
13
14 import sys, tempfile
15 sys.path.append(PYTAN_LIB_PATH)
16
17 import pytan
18 handler = pytan.Handler(
19     username=USERNAME,
20     password=PASSWORD,
21     host=HOST,
22     port=PORT,
23     loglevel=LOGLEVEL,
24     debugformat=DEBUGFORMAT,
25 )
26
27 print handler
28
29 # setup the arguments for the handler method
30 kwargs = {}
31 kwargs["sensors"] = u'Operating System, that contains:Windows'
32 kwargs["qtype"] = u'manual_human'
33
34 # call the handler with the ask method, passing in kwargs for arguments
35 response = handler.ask(**kwargs)
36 import pprint, io
37
38 print ""
39 print "Type of response: ", type(response)
40
41 print ""
42 print "Pretty print of response:"
43 print pprint.pformat(response)
44
45 print ""
46 print "Equivalent Question if it were to be asked in the Tanium Console: "
47 print response['question_object'].query_text
48
49 # create an IO stream to store CSV results to
50 out = io.BytesIO()
51
52 # call the write_csv() method to convert response to CSV and store it in out
53 response['question_results'].write_csv(out, response['question_results'])
54

```

```
55 print ""
56 print "CSV Results of response: "
57 out = out.getvalue()
58 if len(out.splitlines()) > 15:
59     out = out.splitlines()[0:15]
60     out.append('..trimmed for brevity..')
61     out = '\n'.join(out)
62 print out
```

### Output from Python Code

```
1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
2 2014-12-08 16:25:25,602 INFO      question_progress: Results 0% (Get Operating System contains "Windows")
3 2014-12-08 16:25:30,621 INFO      question_progress: Results 50% (Get Operating System contains "Windows")
4 2014-12-08 16:25:35,641 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 0x102123ed0>,
10  'question_results': <taniumpy.object_types.result_set.ResultSet object at 0x10218fc50>}
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 Count,Operating System
17 4,[no results]
18 2,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 # Path to lib directory which contains pytan package
2 PYTAN_LIB_PATH = '../lib'
3
4 # connection info for Tanium Server
5 USERNAME = "Tanium User"
6 PASSWORD = "T@n!um"
7 HOST = "172.16.31.128"
8 PORT = "444"
9
```

```

10 # Logging controls
11 LOGLEVEL = 2
12 DEBUGFORMAT = False
13
14 import sys, tempfile
15 sys.path.append(PYTAN_LIB_PATH)
16
17 import pytan
18 handler = pytan.Handler(
19     username=USERNAME,
20     password=PASSWORD,
21     host=HOST,
22     port=PORT,
23     loglevel=LOGLEVEL,
24     debugformat=DEBUGFORMAT,
25 )
26
27 print handler
28
29 # setup the arguments for the handler method
30 kwargs = {}
31 kwargs["question_filters"] = [u'Operating System, that contains:Windows',
32     u'Operating System, that does not contain:Windows']
33 kwargs["sensors"] = [u'Computer Name',
34     u'Folder Name Search with RegEx Match{dirname=Program Files,regex=Microsoft.*, invalidparam=test}',
35     kwargs["question_options"] = [u'ignore_case', u'or']
36     kwargs["qtype"] = u'manual_human'
37
38 # call the handler with the ask method, passing in kwargs for arguments
39 response = handler.ask(**kwargs)
40 import pprint, io
41
42 print ""
43 print "Type of response: ", type(response)
44
45 print ""
46 print "Pretty print of response:"
47 print pprint.pformat(response)
48
49 print ""
50 print "Equivalent Question if it were to be asked in the Tanium Console: "
51 print response['question_object'].query_text
52
53 # create an IO stream to store CSV results to
54 out = io.BytesIO()
55
56 # call the write_csv() method to convert response to CSV and store it in out
57 response['question_results'].write_csv(out, response['question_results'])
58
59 print ""
60 print "CSV Results of response: "
61 out = out.getvalue()
62 if len(out.splitlines()) > 15:
63     out = out.splitlines()[0:15]
64     out.append('..trimmed for brevity..')
65     out = '\n'.join(out)
66 print out

```

## Output from Python Code

```

1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
2 2014-12-08 16:25:35,790 INFO      question_progress: Results 0% (Get Computer Name and Folder Name Se
3 2014-12-08 16:25:40,818 INFO      question_progress: Results 0% (Get Computer Name and Folder Name Se
4 2014-12-08 16:25:45,841 INFO      question_progress: Results 17% (Get Computer Name and Folder Name S
5 2014-12-08 16:25:50,868 INFO      question_progress: Results 50% (Get Computer Name and Folder Name S
6 2014-12-08 16:25:55,894 INFO      question_progress: Results 67% (Get Computer Name and Folder Name S
7 2014-12-08 16:26:00,919 INFO      question_progress: Results 100% (Get Computer Name and Folder Name
8
9 Type of response: <type 'dict'>
10
11 Pretty print of response:
12 {'question_object': <taniumpy.object_types.question.Question object at 0x102967a10>,
13  'question_results': <taniumpy.object_types.result_set.ResultSet object at 0x1022e0490>}
14
15 Equivalent Question if it were to be asked in the Tanium Console:
16 Get Computer Name and Folder Name Search with RegEx Match[No, Program Files, No, ] contains "Shared"
17
18 CSV Results of response:
19 Computer Name,"Folder Name Search with RegEx Match[No, Program Files, No, ]"
20 Casus-Belli.local,[no results]
21 ubuntu.(none),[no results]
22 localhost.(none),[no results]
23 Jims-Mac.local,[no results]
24 jtanium1.localdomain,"C:\Program Files\Common Files\Microsoft Shared\VS7Debug
25 C:\Program Files\Common Files\Microsoft Shared\ink\ar-SA
26 C:\Program Files\Common Files\Microsoft Shared\ink\ru-RU
27 C:\Program Files\Common Files\Microsoft Shared\ink\fsdefinitions\keypad
28 C:\Program Files\Common Files\Microsoft Shared\ink
29 C:\Program Files\Common Files\Microsoft Shared\ink\sv-SE
30 C:\Program Files\Common Files\Microsoft Shared\ink\uk-UA
31 C:\Program Files\Common Files\Microsoft Shared\ink\sl-SI
32 C:\Program Files\Common Files\Microsoft Shared\ink\hu-HU
33 C:\Program Files\Common Files\Microsoft Shared\ink\zh-TW
34 ..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 # Path to lib directory which contains pytan package
2 PYTAN_LIB_PATH = '../lib'
3
4 # connection info for Tanium Server
5 USERNAME = "Tanium User"
6 PASSWORD = "T@n!um"
7 HOST = "172.16.31.128"
8 PORT = "444"
9
10 # Logging conrols
11 LOGLEVEL = 2

```

```

12  DEBUGFORMAT = False
13
14  import sys, tempfile
15  sys.path.append(PYTAN_LIB_PATH)
16
17  import pytan
18  handler = pytan.Handler(
19      username=USERNAME,
20      password=PASSWORD,
21      host=HOST,
22      port=PORT,
23      loglevel=LOGLEVEL,
24      debugformat=DEBUGFORMAT,
25  )
26
27  print handler
28
29  # setup the arguments for the handler method
30  kwargs = {}
31  kwargs["question_filters"] = [u'Installed Applications, that contains:Google Search',
32      u'Installed Applications, that contains:Google Chrome']
33  kwargs["sensors"] = [u'Computer Name',
34      u'Last Logged In User',
35      u'Installed Applications, that contains:Google Search',
36      u'Installed Applications, that contains:Google Chrome']
37  kwargs["question_options"] = [u'ignore_case', u'and']
38  kwargs["qtype"] = u'manual_human'
39
40  # call the handler with the ask method, passing in kwargs for arguments
41  response = handler.ask(**kwargs)
42  import pprint, io
43
44  print ""
45  print "Type of response: ", type(response)
46
47  print ""
48  print "Pretty print of response:"
49  print pprint.pformat(response)
50
51  print ""
52  print "Equivalent Question if it were to be asked in the Tanium Console: "
53  print response['question_object'].query_text
54
55  # create an IO stream to store CSV results to
56  out = io.BytesIO()
57
58  # call the write_csv() method to convert response to CSV and store it in out
59  response['question_results'].write_csv(out, response['question_results'])
60
61  print ""
62  print "CSV Results of response: "
63  out = out.getvalue()
64  if len(out.splitlines()) > 15:
65      out = out.splitlines()[0:15]
66      out.append('..trimmed for brevity..')
67      out = '\n'.join(out)
68  print out

```

## Output from Python Code

```
1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
2 2014-12-08 16:26:01,111 INFO      question_progress: Results 0% (Get Computer Name and Last Logged In
3 2014-12-08 16:26:06,148 INFO      question_progress: Results 67% (Get Computer Name and Last Logged I
4 2014-12-08 16:26:11,188 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 0x102b0c790>,
10  'question_results': <taniumpy.object_types.result_set.ResultSet object at 0x1029cf510>}
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 # Path to lib directory which contains pytan package
2 PYTAN_LIB_PATH = '../lib'
3
4 # connection info for Tanium Server
5 USERNAME = "Tanium User"
6 PASSWORD = "T@n!um"
7 HOST = "172.16.31.128"
8 PORT = "444"
9
10 # Logging conrols
11 LOGLEVEL = 2
12 DEBUGFORMAT = False
13
14 import sys, tempfile
15 sys.path.append(PYTAN_LIB_PATH)
16
17 import pytan
18 handler = pytan.Handler(
19     username=USERNAME,
20     password=PASSWORD,
```

```

21     host=HOST,
22     port=PORT,
23     loglevel=LOGLEVEL,
24     debugformat=DEBUGFORMAT,
25 )
26
27 print handler
28
29 # setup the arguments for the handler method
30 kwargs = {}
31 kwargs["question_filter_defs"] = [{u'filter': {u'not_flag': 0,
32         u'operator': u'RegexMatch',
33         u'value': u'.*Windows.*'},
34     u'name': u'Operating System'}]
35 kwargs["sensor_defs"] = [u'Computer Name',
36     {u'filter': {u'not_flag': 0,
37         u'operator': u'RegexMatch',
38         u'value': u'.*Shared.*'},
39     u'name': u'Folder Name Search with RegEx Match',
40     u'options': {u'ignore_case_flag': 0,
41         u'max_age_seconds': 3600,
42         u'value_type': u'string'},
43     u'params': {u'dirname': u'Program Files'}}}]
44 kwargs["question_option_defs"] = {u'and_flag': 0, u'ignore_case_flag': 0, u'max_age_seconds': 3600}
45 kwargs["qtype"] = u'manual'
46
47 # call the handler with the ask method, passing in kwargs for arguments
48 response = handler.ask(**kwargs)
49 import pprint, io
50
51 print ""
52 print "Type of response: ", type(response)
53
54 print ""
55 print "Pretty print of response:"
56 print pprint.pformat(response)
57
58 print ""
59 print "Equivalent Question if it were to be asked in the Tanium Console: "
60 print response['question_object'].query_text
61
62 # create an IO stream to store CSV results to
63 out = io.BytesIO()
64
65 # call the write_csv() method to convert response to CSV and store it in out
66 response['question_results'].write_csv(out, response['question_results'])
67
68 print ""
69 print "CSV Results of response: "
70 out = out.getvalue()
71 if len(out.splitlines()) > 15:
72     out = out.splitlines()[0:15]
73     out.append('..trimmed for brevity..')
74     out = '\n'.join(out)
75 print out

```

## Output from Python Code

```
1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
2 2014-12-08 16:26:11,367 INFO      question_progress: Results 0% (Get Computer Name and Folder Name S
3 2014-12-08 16:26:16,396 INFO      question_progress: Results 33% (Get Computer Name and Folder Name S
4 2014-12-08 16:26:21,433 INFO      question_progress: Results 67% (Get Computer Name and Folder Name S
5 2014-12-08 16:26:26,459 INFO      question_progress: Results 67% (Get Computer Name and Folder Name S
6 2014-12-08 16:26:31,481 INFO      question_progress: Results 83% (Get Computer Name and Folder Name S
7 2014-12-08 16:26:36,503 INFO      question_progress: Results 100% (Get Computer Name and Folder Name S
8
9 Type of response: <type 'dict'>
10
11 Pretty print of response:
12 {'question_object': <taniumpy.object_types.question.Question object at 0x1029571d0>,
13  'question_results': <taniumpy.object_types.result_set.ResultSet object at 0x102116710>}
14
15 Equivalent Question if it were to be asked in the Tanium Console:
16 Get Computer Name and Folder Name Search with RegEx Match[No, Program Files, No, ] contains "Shared"
17
18 CSV Results of response:
19 Computer Name,"Folder Name Search with RegEx Match[No, Program Files, No, ]"
20 jtanium1.localdomain,"C:\Program Files\Common Files\Microsoft Shared\VS7Debug
21 C:\Program Files\Common Files\Microsoft Shared\ink\ar-SA
22 C:\Program Files\Common Files\Microsoft Shared\ink\ru-RU
23 C:\Program Files\Common Files\Microsoft Shared\ink\fsdefinitions\keypad
24 C:\Program Files\Common Files\Microsoft Shared\ink
25 C:\Program Files\Common Files\Microsoft Shared\ink\sv-SE
26 C:\Program Files\Common Files\Microsoft Shared\ink\uk-UA
27 C:\Program Files\Common Files\Microsoft Shared\ink\sl-SI
28 C:\Program Files\Common Files\Microsoft Shared\ink\hu-HU
29 C:\Program Files\Common Files\Microsoft Shared\ink\zh-TW
30 C:\Program Files\Common Files\Microsoft Shared\ink\zh-CN
31 C:\Program Files\Common Files\Microsoft Shared\ink\fi-FI
32 C:\Program Files\Common Files\Microsoft Shared
33 C:\Program Files\Common Files\Microsoft Shared\ink\da-DK
34 ..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 # Path to lib directory which contains pytan package
2 PYTAN_LIB_PATH = '../lib'
3
4 # connection info for Tanium Server
5 USERNAME = "Tanium User"
6 PASSWORD = "T@n!um"
7 HOST = "172.16.31.128"
8 PORT = "444"
9
10 # Logging conrols
11 LOGLEVEL = 2
12 DEBUGFORMAT = False
```



```

13
14 import sys, tempfile
15 sys.path.append(PYTAN_LIB_PATH)
16
17 import pytan
18 handler = pytan.Handler(
19     username=USERNAME,
20     password=PASSWORD,
21     host=HOST,
22     port=PORT,
23     loglevel=LOGLEVEL,
24     debugformat=DEBUGFORMAT,
25 )
26
27 print handler
28
29 # setup the arguments for the handler method
30 kwargs = {}
31 kwargs["filters_help"] = True
32 kwargs["qtype"] = u'manual_human'
33
34
35 # call the handler with the ask method, passing in kwargs for arguments
36 # this should throw an exception: pytan.utils.PytanHelp
37 import traceback
38 try:
39     handler.ask(**kwargs)
40 except Exception as e:
41     traceback.print_exc(file=sys.stdout)

```

### Output from Python Code

```

1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
2 Traceback (most recent call last):
3   File "<string>", line 39, 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
86     '>'
87         Help: Filter for greater than VALUE
88         Example: "Sensor1, that >:VALUE"
89
90     'greater'
91         Help: Filter for greater than VALUE
92         Example: "Sensor1, that greater:VALUE"
93
94     'gt'
95         Help: Filter for greater than VALUE
96         Example: "Sensor1, that gt:VALUE"
97
98     'greater than'
99         Help: Filter for greater than VALUE
100        Example: "Sensor1, that greater than:VALUE"
101
102     '!>'
103         Help: Filter for not greater than VALUE
104         Example: "Sensor1, that !>:VALUE"
105
106     'not greater'
107         Help: Filter for not greater than VALUE
108         Example: "Sensor1, that not greater:VALUE"
109
110     'notgreater'
111         Help: Filter for not greater than VALUE
112         Example: "Sensor1, that notgreater:VALUE"
113
114     'not greater than'
115         Help: Filter for not greater than VALUE
116         Example: "Sensor1, that not greater than:VALUE"
117
118     '=>'
119         Help: Filter for greater than or equal to VALUE
120         Example: "Sensor1, that =>:VALUE"
121
122     'greater equal'
123         Help: Filter for greater than or equal to VALUE
124         Example: "Sensor1, that greater equal:VALUE"
125
126     'greaterequal'
127         Help: Filter for greater than or equal to VALUE
128         Example: "Sensor1, that greaterequal:VALUE"
129
130     'ge'
131         Help: Filter for greater than or equal to VALUE
132         Example: "Sensor1, that ge:VALUE"
133
134     '!>='
135         Help: Filter for not greater than VALUE
136         Example: "Sensor1, that !>=:VALUE"
137
138     'not greater equal'
139         Help: Filter for not greater than VALUE
140         Example: "Sensor1, that not greater equal:VALUE"
141

```

```
142 'notgreaterequal'
143     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  # Path to lib directory which contains pytan package
2  PYTAN_LIB_PATH = '../lib'
3
4  # connection info for Tanium Server
5  USERNAME = "Tanium User"
6  PASSWORD = "T@n!um"
7  HOST = "172.16.31.128"
8  PORT = "444"
9
```

```

10 # Logging controls
11 LOGLEVEL = 2
12 DEBUGFORMAT = False
13
14 import sys, tempfile
15 sys.path.append(PYTAN_LIB_PATH)
16
17 import pytan
18 handler = pytan.Handler(
19     username=USERNAME,
20     password=PASSWORD,
21     host=HOST,
22     port=PORT,
23     loglevel=LOGLEVEL,
24     debugformat=DEBUGFORMAT,
25 )
26
27 print handler
28
29 # setup the arguments for the handler method
30 kwargs = {}
31 kwargs["options_help"] = True
32 kwargs["qtype"] = u'manual_human'
33
34
35 # call the handler with the ask method, passing in kwargs for arguments
36 # this should throw an exception: pytan.utils.PytanHelp
37 import traceback
38 try:
39     handler.ask(**kwargs)
40 except Exception as e:
41     traceback.print_exc(file=sys.stdout)

```

### Output from Python Code

```

1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
2 Traceback (most recent call last):
3   File "<string>", line 39, 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

```

supplied as 'OPTION' or 'OPTION:VALUE' for options that require a value.

Options can be used on 'filter' or 'group', where 'group' pertains to group filters or question filters. All 'filter' options are also applicable to 'group' for question options.

#### Valid Options

-----

##### 'ignore\_case'

Help: Make the filter do a case insensitive match

Usable on: filter

Example for sensor: "Sensor1, opt:ignore\_case"

Example for question: "ignore\_case"

##### 'match\_case'

Help: Make the filter do a case sensitive match

Usable on: filter

Example for sensor: "Sensor1, opt:match\_case"

Example for question: "match\_case"

##### 'match\_any\_value'

Help: Make the filter match any value

Usable on: filter

Example for sensor: "Sensor1, opt:match\_any\_value"

Example for question: "match\_any\_value"

##### 'match\_all\_values'

Help: Make the filter match all values

Usable on: filter

Example for sensor: "Sensor1, opt:match\_all\_values"

Example for question: "match\_all\_values"

##### 'max\_data\_age'

Help: Re-fetch cached values older than N seconds

Usable on: filter

VALUE description and type: seconds, <type 'int'>

Example for sensor: "Sensor1, opt:max\_data\_age:seconds"

Example for question: "max\_data\_age:seconds"

##### 'value\_type'

Help: Make the filter consider the value type as VALUE\_TYPE

Usable on: filter

VALUE description and type: value\_type, <type 'str'>

Example for sensor: "Sensor1, opt:value\_type:value\_type"

Example for question: "value\_type:value\_type"

##### 'and'

Help: Use 'and' for all of the filters supplied

Usable on: group

Example for sensor: "Sensor1, opt:and"

Example for question: "and"

##### 'or'

Help: Use 'or' for all of the filters supplied

Usable on: group

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  # Path to lib directory which contains pytan package
2  PYTAN_LIB_PATH = '../lib'
3
4  # connection info for Tanium Server
5  USERNAME = "Tanium User"
6  PASSWORD = "T@n!um"
7  HOST = "172.16.31.128"
8  PORT = "444"
9
10 # Logging conrols
11 LOGLEVEL = 2
12 DEBUGFORMAT = False
13
14 import sys, tempfile
15 sys.path.append(PYTAN_LIB_PATH)
16
17 import pytan
18 handler = pytan.Handler(
19     username=USERNAME,
20     password=PASSWORD,
21     host=HOST,
22     port=PORT,
23     loglevel=LOGLEVEL,
24     debugformat=DEBUGFORMAT,
25 )
26
27 print handler
28
29 # setup the arguments for the handler method
30 kwargs = {}
31 kwargs["qtype"] = u'manual_human'
32 kwargs["sensors_help"] = True
33
34
35 # call the handler with the ask method, passing in kwargs for arguments
36 # this should throw an exception: pytan.utils.PytanHelp
37 import traceback
38 try:
39     handler.ask(**kwargs)
40 except Exception as e:
41     traceback.print_exc(file=sys.stdout)
```

### Output from Python Code

```
1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
2 Traceback (most recent call last):
```

```
3 File "<string>", line 39, 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     [
35         'Computer Name',
36         'name:Computer Name',
37         'id:1',
38         'hash:123456789',
39     ]
40
41 Sensor Parameters
42 -----
43
44 Supplying parameters to a sensor can control the arguments that are
45 supplied to a sensor, if that sensor takes any arguments.
46
47 Sensor parameters must be surrounded with curly braces '{}',
48 and must have a key and value specified that is separated by
49 an equals '='. Multiple parameters must be seperated by
50 a comma ','. The key should match up to a valid parameter key
51 for the sensor in question.
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 (pulldowns, for
```

example), a default value is derived (for pulldowns,  
the first value available as a pulldown entry is used).

#### Examples for sensors with parameters

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

```
'Sensor With Params{dirname=Program Files}'
```

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

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

#### Sensor Filters

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

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

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

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

#### 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

```
119 that sensor works.
```

```
120
121 Sensor options can be supplied by adding ', opt:OPTION' or
122 ', opt:OPTION:VALUE' for those options that require values,
123 where OPTION is a valid option string, and VALUE is the
124 appropriate value required by accordant OPTION.
```

```
125
126 See options help for a list of options that can control how
127 the filter works.
```

```
128
129 Examples for sensors with options
```

```
130 -----
```

```
131
132 Supplying a sensor with an option that forces tanium to
133 re-fetch any cached column data that is older than 1 minute:
```

```
134
135     'Computer Name, opt:max_data_age:60'
```

```
136
137 Supplying a sensor with filter and an option that causes
138 Tanium to match case for the filter value:
```

```
139
140     'Computer Name, that contains:Windows, opt:match_case'
```

```
141
142 Supplying a sensor with a filter and an option that causes
143 Tanium to match all values supplied:
```

```
144
145     'Computer Name, that contains:Windows, opt:match_all_values'
```

```
146
147 Supplying a sensor with a filter and a set of options that
148 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  # Path to lib directory which contains pytan package
2  PYTAN_LIB_PATH = '../lib'
3
4  # connection info for Tanium Server
5  USERNAME = "Tanium User"
6  PASSWORD = "T@n!um"
7  HOST = "172.16.31.128"
8  PORT = "444"
9
10 # Logging conrols
11 LOGLEVEL = 2
12 DEBUGFORMAT = False
```

```

13
14 import sys, tempfile
15 sys.path.append(PYTAN_LIB_PATH)
16
17 import pytan
18 handler = pytan.Handler(
19     username=USERNAME,
20     password=PASSWORD,
21     host=HOST,
22     port=PORT,
23     loglevel=LOGLEVEL,
24     debugformat=DEBUGFORMAT,
25 )
26
27 print handler
28
29 # setup the arguments for the handler method
30 kwargs = {}
31 kwargs["sensors"] = u'Computer name, that does not meet:little'
32 kwargs["qtype"] = u'manual_human'
33
34
35 # call the handler with the ask method, passing in kwargs for arguments
36 # this should throw an exception: pytan.utils.HumanParserError
37 import traceback
38 try:
39     handler.ask(**kwargs)
40 except Exception as e:
41     traceback.print_exc(file=sys.stdout)

```

### Output from Python Code

```

1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
2 Traceback (most recent call last):
3   File "<string>", line 39, 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 1304, in dehumanize_sensors
9     s, parsed_filter = extract_filter(s)
10    File "/Users/jolsen/gh/pytan/lib/pytan/utils.py", line 1666, 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 # Path to lib directory which contains pytan package
2 PYTAN_LIB_PATH = '../lib'
3

```

```
4  # connection info for Tanium Server
5  USERNAME = "Tanium User"
6  PASSWORD = "T@n!um"
7  HOST = "172.16.31.128"
8  PORT = "444"
9
10 # Logging conrols
11 LOGLEVEL = 2
12 DEBUGFORMAT = False
13
14 import sys, tempfile
15 sys.path.append(PYTAN_LIB_PATH)
16
17 import pytan
18 handler = pytan.Handler(
19     username=USERNAME,
20     password=PASSWORD,
21     host=HOST,
22     port=PORT,
23     loglevel=LOGLEVEL,
24     debugformat=DEBUGFORMAT,
25 )
26
27 print handler
28
29 # setup the arguments for the handler method
30 kwargs = {}
31 kwargs["sensor_defs"] = u'Dweedle Dee and Dum'
32 kwargs["qtype"] = u'manual'
33
34
35 # call the handler with the ask method, passing in kwargs for arguments
36 # this should throw an exception: pytan.utils.HandlerError
37 import traceback
38 try:
39     handler.ask(**kwargs)
40 except Exception as e:
41     traceback.print_exc(file=sys.stdout)
```

### Output from Python Code

```
1  Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
2  Traceback (most recent call last):
3    File "<string>", line 39, 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 1784, 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 1567, in get
11     return self._get_multi(obj_map, **kwargs)
12   File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 1729, in _get_multi
13     found = self._find(api_obj_multi, **kwargs)
14   File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 1694, 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 | # Path to lib directory which contains pytan package
2 | PYTAN_LIB_PATH = '../lib'
3 |
4 | # connection info for Tanium Server
5 | USERNAME = "Tanium User"
6 | PASSWORD = "T@n!um"
7 | HOST = "172.16.31.128"
8 | PORT = "444"
9 |
10 | # Logging conrols
11 | LOGLEVEL = 2
12 | DEBUGFORMAT = False
13 |
14 | import sys, tempfile
15 | sys.path.append(PYTAN_LIB_PATH)
16 |
17 | import pytan
18 | handler = pytan.Handler(
19 |     username=USERNAME,
20 |     password=PASSWORD,
21 |     host=HOST,
22 |     port=PORT,
23 |     loglevel=LOGLEVEL,
24 |     debugformat=DEBUGFORMAT,
25 | )
26 |
27 | print handler
28 |
29 | # setup the arguments for the handler method
30 | kwargs = {}
31 | kwargs["sensors"] = u'Folder Name Search with RegEx Match(dirname=Program Files,regex=.*){}'
32 | kwargs["qtype"] = u'manual_human'
33 |
34 |
35 | # call the handler with the ask method, passing in kwargs for arguments
36 | # this should throw an exception: pytan.utils.HumanParserError
37 | import traceback
38 | try:
39 |     handler.ask(**kwargs)
40 | except Exception as e:
41 |     traceback.print_exc(file=sys.stdout)
```

### Output from Python Code

```
1 | Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
2 | Traceback (most recent call last):
```

```
3 File "<string>", line 39, 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_params = extract_params(s)
10 File "/Users/jolsen/gh/pytan/lib/pytan/utils.py", line 1472, 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 # Path to lib directory which contains pytan package
2 PYTAN_LIB_PATH = '../lib'
3
4 # connection info for Tanium Server
5 USERNAME = "Tanium User"
6 PASSWORD = "T@n!um"
7 HOST = "172.16.31.128"
8 PORT = "444"
9
10 # Logging conrols
11 LOGLEVEL = 2
12 DEBUGFORMAT = False
13
14 import sys, tempfile
15 sys.path.append(PYTAN_LIB_PATH)
16
17 import pytan
18 handler = pytan.Handler(
19     username=USERNAME,
20     password=PASSWORD,
21     host=HOST,
22     port=PORT,
23     loglevel=LOGLEVEL,
24     debugformat=DEBUGFORMAT,
25 )
26
27 print handler
28
29 # setup the arguments for the handler method
30 kwargs = {}
31 kwargs["sensors"] = u'Operating system, opt:bad'
32 kwargs["qtype"] = u'manual_human'
33
34
35 # call the handler with the ask method, passing in kwargs for arguments
36 # this should throw an exception: pytan.utils.HumanParserError
37 import traceback
38 try:
```



```

39     handler.ask(**kwargs)
40 except Exception as e:
41     traceback.print_exc(file=sys.stdout)

```

### Output from Python Code

```

1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
2 Traceback (most recent call last):
3   File "<string>", line 39, 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_options = extract_options(s)
10  File "/Users/jolsen/gh/pytan/lib/pytan/utils.py", line 1541, in extract_options
11    parsed_options = map_options(parsed_options, ['filter'])
12  File "/Users/jolsen/gh/pytan/lib/pytan/utils.py", line 1571, 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  # Path to lib directory which contains pytan package
2  PYTAN_LIB_PATH = '../lib'
3
4  # connection info for Tanium Server
5  USERNAME = "Tanium User"
6  PASSWORD = "T@n!um"
7  HOST = "172.16.31.128"
8  PORT = "444"
9
10 # Logging conrols
11 LOGLEVEL = 2
12 DEBUGFORMAT = False
13
14 import sys, tempfile
15 sys.path.append(PYTAN_LIB_PATH)
16
17 import pytan
18 handler = pytan.Handler(
19     username=USERNAME,
20     password=PASSWORD,
21     host=HOST,
22     port=PORT,
23     loglevel=LOGLEVEL,
24     debugformat=DEBUGFORMAT,
25 )
26
27 print handler

```

```
28
29 # setup the arguments for the handler method
30 kwargs = {}
31 kwargs["sensors"] = u'Computer Name{Dweedle}'
32 kwargs["qtype"] = u'manual_human'
33
34
35 # call the handler with the ask method, passing in kwargs for arguments
36 # this should throw an exception: pytan.utils.HumanParserError
37 import traceback
38 try:
39     handler.ask(**kwargs)
40 except Exception as e:
41     traceback.print_exc(file=sys.stdout)
```

### Output from Python Code

```
1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
2 Traceback (most recent call last):
3   File "<string>", line 39, 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_params = extract_params(s)
10  File "/Users/jolsen/gh/pytan/lib/pytan/utils.py", line 1490, 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 # Path to lib directory which contains pytan package
2 PYTAN_LIB_PATH = '../lib'
3
4 # connection info for Tanium Server
5 USERNAME = "Tanium User"
6 PASSWORD = "T@n!um"
7 HOST = "172.16.31.128"
8 PORT = "444"
9
10 # Logging conrols
11 LOGLEVEL = 2
12 DEBUGFORMAT = False
13
14 import sys, tempfile
15 sys.path.append(PYTAN_LIB_PATH)
16
```

```

17 import pytan
18 handler = pytan.Handler(
19     username=USERNAME,
20     password=PASSWORD,
21     host=HOST,
22     port=PORT,
23     loglevel=LOGLEVEL,
24     debugformat=DEBUGFORMAT,
25 )
26
27 print handler
28
29 # setup the arguments for the handler method
30 kwargs = {}
31 kwargs["objtype"] = u'action'
32 kwargs["id"] = 1
33
34 # call the handler with the get method, passing in kwargs for arguments
35 response = handler.get(**kwargs)
36
37 print ""
38 print "Type of response: ", type(response)
39
40 print ""
41 print "print of response:"
42 print response
43
44 print ""
45 print "length of response (number of objects returned): "
46 print len(response)
47
48 print ""
49 print "print the first object returned in JSON format:"
50 out = response.to_json(response[0])
51 if len(out.splitlines()) > 15:
52     out = out.splitlines()[0:15]
53     out.append('..trimmed for brevity..')
54     out = '\n'.join(out)
55
56 print out

```

### Output from Python Code

```

1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
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": "2014-12-08T19:22:33",
21 "distribute_seconds": 600,
22 "expire_seconds": 1800,
23 "history_saved_question": {
24     "_type": "saved_question",
25     "id": 173
26 },
27 ..trimmed for brevity..
```

### Get question by id

Get a question by id

### Example Python Code

```
1  # Path to lib directory which contains pytan package
2  PYTAN_LIB_PATH = '../lib'
3
4  # connection info for Tanium Server
5  USERNAME = "Tanium User"
6  PASSWORD = "T@n!um"
7  HOST = "172.16.31.128"
8  PORT = "444"
9
10 # Logging conrols
11 LOGLEVEL = 2
12 DEBUGFORMAT = False
13
14 import sys, tempfile
15 sys.path.append(PYTAN_LIB_PATH)
16
17 import pytan
18 handler = pytan.Handler(
19     username=USERNAME,
20     password=PASSWORD,
21     host=HOST,
22     port=PORT,
23     loglevel=LOGLEVEL,
24     debugformat=DEBUGFORMAT,
25 )
26
27 print handler
28
29 # setup the arguments for the handler method
30 kwargs = {}
31 kwargs["objtype"] = u'question'
32 kwargs["id"] = 1
33
34 # call the handler with the get method, passing in kwargs for arguments
35 response = handler.get(**kwargs)
```

```

36
37 print ""
38 print "Type of response: ", type(response)
39
40 print ""
41 print "print of response:"
42 print response
43
44 print ""
45 print "length of response (number of objects returned): "
46 print len(response)
47
48 print ""
49 print "print the first object returned in JSON format:"
50 out = response.to_json(response[0])
51 if len(out.splitlines()) > 15:
52     out = out.splitlines()[0:15]
53     out.append('..trimmed for brevity..')
54     out = '\n'.join(out)
55
56 print out

```

### Output from Python Code

```

1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
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": "2014-12-08T19:30:12",
20     "expire_seconds": 0,
21     "force_computer_id_flag": 1,
22     "hidden_flag": 0,
23     "id": 1,
24     "management_rights_group": {
25         "_type": "group",
26         "id": 0
27     }
28 }
29 ..trimmed for brevity..

```

### Get saved question by names

Get two saved questions by name

## Example Python Code

```
1  # Path to lib directory which contains pytan package
2  PYTAN_LIB_PATH = '../lib'
3
4  # connection info for Tanium Server
5  USERNAME = "Tanium User"
6  PASSWORD = "T@n!um"
7  HOST = "172.16.31.128"
8  PORT = "444"
9
10 # Logging conrols
11 LOGLEVEL = 2
12 DEBUGFORMAT = False
13
14 import sys, tempfile
15 sys.path.append(PYTAN_LIB_PATH)
16
17 import pytan
18 handler = pytan.Handler(
19     username=USERNAME,
20     password=PASSWORD,
21     host=HOST,
22     port=PORT,
23     loglevel=LOGLEVEL,
24     debugformat=DEBUGFORMAT,
25 )
26
27 print handler
28
29 # setup the arguments for the handler method
30 kwargs = {}
31 kwargs["objtype"] = u'saved_question'
32 kwargs["name"] = [u'Installed Applications', u'Computer Name']
33
34 # call the handler with the get method, passing in kwargs for arguments
35 response = handler.get(**kwargs)
36
37 print ""
38 print "Type of response: ", type(response)
39
40 print ""
41 print "print of response:"
42 print response
43
44 print ""
45 print "length of response (number of objects returned): "
46 print len(response)
47
48 print ""
49 print "print the first object returned in JSON format:"
50 out = response.to_json(response[0])
51 if len(out.splitlines()) > 15:
52     out = out.splitlines()[0:15]
53     out.append('..trimmed for brevity..')
54     out = '\n'.join(out)
55
56 print out
```

## Output from Python Code

```

1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
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 # Path to lib directory which contains pytan package
2 PYTAN_LIB_PATH = '../lib'
3
4 # connection info for Tanium Server
5 USERNAME = "Tanium User"
6 PASSWORD = "T@n!um"
7 HOST = "172.16.31.128"
8 PORT = "444"
9
10 # Logging conrols
11 LOGLEVEL = 2
12 DEBUGFORMAT = False
13
14 import sys, tempfile
15 sys.path.append(PYTAN_LIB_PATH)
16
17 import pytan
18 handler = pytan.Handler(
19     username=USERNAME,
20     password=PASSWORD,

```

```
21     host=HOST,
22     port=PORT,
23     loglevel=LOGLEVEL,
24     debugformat=DEBUGFORMAT,
25 )
26
27 print handler
28
29 # setup the arguments for the handler method
30 kwargs = {}
31 kwargs["objtype"] = u'userrole'
32 kwargs["id"] = 1
33
34 # call the handler with the get method, passing in kwargs for arguments
35 response = handler.get(**kwargs)
36
37 print ""
38 print "Type of response: ", type(response)
39
40 print ""
41 print "print of response:"
42 print response
43
44 print ""
45 print "length of response (number of objects returned): "
46 print len(response)
47
48 print ""
49 print "print the first object returned in JSON format:"
50 out = response.to_json(response[0])
51 if len(out.splitlines()) > 15:
52     out = out.splitlines()[0:15]
53     out.append('..trimmed for brevity..')
54     out = '\n'.join(out)
55
56 print out
```

### Output from Python Code

```
1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
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  # Path to lib directory which contains pytan package
2  PYTAN_LIB_PATH = '../lib'
3
4  # connection info for Tanium Server
5  USERNAME = "Tanium User"
6  PASSWORD = "T@n!um"
7  HOST = "172.16.31.128"
8  PORT = "444"
9
10 # Logging conrols
11 LOGLEVEL = 2
12 DEBUGFORMAT = False
13
14 import sys, tempfile
15 sys.path.append(PYTAN_LIB_PATH)
16
17 import pytan
18 handler = pytan.Handler(
19     username=USERNAME,
20     password=PASSWORD,
21     host=HOST,
22     port=PORT,
23     loglevel=LOGLEVEL,
24     debugformat=DEBUGFORMAT,
25 )
26
27 print handler
28
29 # setup the arguments for the handler method
30 kwargs = {}
31 kwargs["objtype"] = u'client'
32 kwargs["status"] = u'Leader'
33
34 # call the handler with the get method, passing in kwargs for arguments
35 response = handler.get(**kwargs)
36
37 print ""
38 print "Type of response: ", type(response)
39
40 print ""
41 print "print of response:"
42 print response
43
44 print ""
45 print "length of response (number of objects returned): "

```

```
46 print len(response)
47
48 print ""
49 print "print the first object returned in JSON format:"
50 out = response.to_json(response[0])
51 if len(out.splitlines()) > 15:
52     out = out.splitlines()[0:15]
53     out.append('..trimmed for brevity..')
54     out = '\n'.join(out)
55
56 print out
```

### Output from Python Code

```
1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
2
3 Type of response: <class 'taniumpy.object_types.system_status_list.SystemStatusList'>
4
5 print of response:
6 SystemStatusList, 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": "client_status",
14     "cache_row_id": 2,
15     "computer_id": "3508795802",
16     "full_version": "6.0.314.1190",
17     "host_name": "WIN-A12SC6N6T7Q",
18     "ipaddress_client": "172.16.31.145",
19     "ipaddress_server": "172.16.31.145",
20     "last_registration": "2014-12-08T21:26:21",
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 # Path to lib directory which contains pytan package
2 PYTAN_LIB_PATH = '../lib'
3
4 # connection info for Tanium Server
5 USERNAME = "Tanium User"
6 PASSWORD = "T@n!um"
7 HOST = "172.16.31.128"
```

```

8  PORT = "444"
9
10 # Logging controls
11 LOGLEVEL = 2
12 DEBUGFORMAT = False
13
14 import sys, tempfile
15 sys.path.append(PYTAN_LIB_PATH)
16
17 import pytan
18 handler = pytan.Handler(
19     username=USERNAME,
20     password=PASSWORD,
21     host=HOST,
22     port=PORT,
23     loglevel=LOGLEVEL,
24     debugformat=DEBUGFORMAT,
25 )
26
27 print handler
28
29 # setup the arguments for the handler method
30 kwargs = {}
31 kwargs["objtype"] = u'setting'
32 kwargs["name"] = u'control_address'
33
34 # call the handler with the get method, passing in kwargs for arguments
35 response = handler.get(**kwargs)
36
37 print ""
38 print "Type of response: ", type(response)
39
40 print ""
41 print "print of response:"
42 print response
43
44 print ""
45 print "length of response (number of objects returned): "
46 print len(response)
47
48 print ""
49 print "print the first object returned in JSON format:"
50 out = response.to_json(response[0])
51 if len(out.splitlines()) > 15:
52     out = out.splitlines()[0:15]
53     out.append('..trimmed for brevity..')
54     out = '\n'.join(out)
55
56 print out

```

### Output from Python Code

```

1  Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
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  # Path to lib directory which contains pytan package
2  PYTAN_LIB_PATH = '../lib'
3
4  # connection info for Tanium Server
5  USERNAME = "Tanium User"
6  PASSWORD = "T@n!um"
7  HOST = "172.16.31.128"
8  PORT = "444"
9
10 # Logging conrols
11 LOGLEVEL = 2
12 DEBUGFORMAT = False
13
14 import sys, tempfile
15 sys.path.append(PYTAN_LIB_PATH)
16
17 import pytan
18 handler = pytan.Handler(
19     username=USERNAME,
20     password=PASSWORD,
21     host=HOST,
22     port=PORT,
23     loglevel=LOGLEVEL,
24     debugformat=DEBUGFORMAT,
25 )
26
27 print handler
28
29 # setup the arguments for the handler method
30 kwargs = {}
31 kwargs["objtype"] = u'user'
```

```

32 kwargs["name"] = u'Tanium User'
33
34 # call the handler with the get method, passing in kwargs for arguments
35 response = handler.get(**kwargs)
36
37 print ""
38 print "Type of response: ", type(response)
39
40 print ""
41 print "print of response:"
42 print response
43
44 print ""
45 print "length of response (number of objects returned): "
46 print len(response)
47
48 print ""
49 print "print the first object returned in JSON format:"
50 out = response.to_json(response[0])
51 if len(out.splitlines()) > 15:
52     out = out.splitlines()[0:15]
53     out.append('..trimmed for brevity..')
54     out = '\n'.join(out)
55
56 print out

```

### Output from Python Code

```

1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
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": "2014-12-08T21:26:37",
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  # Path to lib directory which contains pytan package
2  PYTAN_LIB_PATH = '../lib'
3
4  # connection info for Tanium Server
5  USERNAME = "Tanium User"
6  PASSWORD = "T@n!um"
7  HOST = "172.16.31.128"
8  PORT = "444"
9
10 # Logging controls
11 LOGLEVEL = 2
12 DEBUGFORMAT = False
13
14 import sys, tempfile
15 sys.path.append(PYTAN_LIB_PATH)
16
17 import pytan
18 handler = pytan.Handler(
19     username=USERNAME,
20     password=PASSWORD,
21     host=HOST,
22     port=PORT,
23     loglevel=LOGLEVEL,
24     debugformat=DEBUGFORMAT,
25 )
26
27 print handler
28
29 # setup the arguments for the handler method
30 kwargs = {}
31 kwargs["objtype"] = u'sensor'
32 kwargs["id"] = 1
33
34 # call the handler with the get method, passing in kwargs for arguments
35 response = handler.get(**kwargs)
36
37 print ""
38 print "Type of response: ", type(response)
39
40 print ""
41 print "print of response:"
42 print response
43
44 print ""
45 print "length of response (number of objects returned): "
46 print len(response)
47
48 print ""
49 print "print the first object returned in JSON format:"
50 out = response.to_json(response[0])
51 if len(out.splitlines()) > 15:
52     out = out.splitlines()[0:15]
```

```

53     out.append('..trimmed for brevity..')
54     out = '\n'.join(out)
55
56 print out

```

### Output from Python Code

```

1  Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
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  # Path to lib directory which contains pytan package
2  PYTAN_LIB_PATH = '../lib'
3
4  # connection info for Tanium Server
5  USERNAME = "Tanium User"
6  PASSWORD = "T@n!um"
7  HOST = "172.16.31.128"
8  PORT = "444"
9
10 # Logging conrols
11 LOGLEVEL = 2
12 DEBUGFORMAT = False
13

```

```
14 import sys, tempfile
15 sys.path.append(PYTAN_LIB_PATH)
16
17 import pytan
18 handler = pytan.Handler(
19     username=USERNAME,
20     password=PASSWORD,
21     host=HOST,
22     port=PORT,
23     loglevel=LOGLEVEL,
24     debugformat=DEBUGFORMAT,
25 )
26
27 print handler
28
29 # setup the arguments for the handler method
30 kwargs = {}
31 kwargs["objtype"] = u'sensor'
32 kwargs["hash"] = [u'322086833']
33 kwargs["name"] = [u'Computer Name']
34 kwargs["id"] = [1, 2]
35
36 # call the handler with the get method, passing in kwargs for arguments
37 response = handler.get(**kwargs)
38
39 print ""
40 print "Type of response: ", type(response)
41
42 print ""
43 print "print of response:"
44 print response
45
46 print ""
47 print "length of response (number of objects returned): "
48 print len(response)
49
50 print ""
51 print "print the first object returned in JSON format:"
52 out = response.to_json(response[0])
53 if len(out.splitlines()) > 15:
54     out = out.splitlines()[0:15]
55     out.append('..trimmed for brevity..')
56     out = '\n'.join(out)
57
58 print out
```

### Output from Python Code

```
1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
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  # Path to lib directory which contains pytan package
2  PYTAN_LIB_PATH = '../lib'
3
4  # connection info for Tanium Server
5  USERNAME = "Tanium User"
6  PASSWORD = "T@n!um"
7  HOST = "172.16.31.128"
8  PORT = "444"
9
10 # Logging conrols
11 LOGLEVEL = 2
12 DEBUGFORMAT = False
13
14 import sys, tempfile
15 sys.path.append(PYTAN_LIB_PATH)
16
17 import pytan
18 handler = pytan.Handler(
19     username=USERNAME,
20     password=PASSWORD,
21     host=HOST,
22     port=PORT,
23     loglevel=LOGLEVEL,
24     debugformat=DEBUGFORMAT,
25 )
26
27 print handler
28
29 # setup the arguments for the handler method
30 kwargs = {}

```

```
31 kwargs["objtype"] = u'whitelisted_url'
32 kwargs["id"] = 1
33
34 # call the handler with the get method, passing in kwargs for arguments
35 response = handler.get(**kwargs)
36
37 print ""
38 print "Type of response: ", type(response)
39
40 print ""
41 print "print of response:"
42 print response
43
44 print ""
45 print "length of response (number of objects returned): "
46 print len(response)
47
48 print ""
49 print "print the first object returned in JSON format:"
50 out = response.to_json(response[0])
51 if len(out.splitlines()) > 15:
52     out = out.splitlines()[0:15]
53     out.append('..trimmed for brevity..')
54     out = '\n'.join(out)
55
56 print out
```

### Output from Python Code

```
1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
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  # Path to lib directory which contains pytan package
2  PYTAN_LIB_PATH = '../lib'
3
4  # connection info for Tanium Server
5  USERNAME = "Tanium User"
6  PASSWORD = "T@n!um"
7  HOST = "172.16.31.128"
8  PORT = "444"
9
10 # Logging conrols
11 LOGLEVEL = 2
12 DEBUGFORMAT = False
13
14 import sys, tempfile
15 sys.path.append(PYTAN_LIB_PATH)
16
17 import pytan
18 handler = pytan.Handler(
19     username=USERNAME,
20     password=PASSWORD,
21     host=HOST,
22     port=PORT,
23     loglevel=LOGLEVEL,
24     debugformat=DEBUGFORMAT,
25 )
26
27 print handler
28
29 # setup the arguments for the handler method
30 kwargs = {}
31 kwargs["objtype"] = u'group'
32 kwargs["name"] = u'All Computers'
33
34 # call the handler with the get method, passing in kwargs for arguments
35 response = handler.get(**kwargs)
36
37 print ""
38 print "Type of response: ", type(response)
39
40 print ""
41 print "print of response:"
42 print response
43
44 print ""
45 print "length of response (number of objects returned): "
46 print len(response)
47
48 print ""
49 print "print the first object returned in JSON format:"
50 out = response.to_json(response[0])
51 if len(out.splitlines()) > 15:
52     out = out.splitlines()[0:15]
53     out.append('..trimmed for brevity..')
54     out = '\n'.join(out)
55
56 print out

```

## Output from Python Code

```
1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
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": 1,
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 # Path to lib directory which contains pytan package
2 PYTAN_LIB_PATH = '../lib'
3
4 # connection info for Tanium Server
5 USERNAME = "Tanium User"
6 PASSWORD = "T@n!um"
7 HOST = "172.16.31.128"
8 PORT = "444"
9
10 # Logging conrols
11 LOGLEVEL = 2
12 DEBUGFORMAT = False
13
14 import sys, tempfile
15 sys.path.append(PYTAN_LIB_PATH)
16
17 import pytan
18 handler = pytan.Handler(
19     username=USERNAME,
20     password=PASSWORD,
```

```

21     host=HOST,
22     port=PORT,
23     loglevel=LOGLEVEL,
24     debugformat=DEBUGFORMAT,
25 )
26
27 print handler
28
29 # setup the arguments for the handler method
30 kwargs = {}
31 kwargs["objtype"] = u'sensor'
32 kwargs["hash"] = u'322086833'
33
34 # call the handler with the get method, passing in kwargs for arguments
35 response = handler.get(**kwargs)
36
37 print ""
38 print "Type of response: ", type(response)
39
40 print ""
41 print "print of response:"
42 print response
43
44 print ""
45 print "length of response (number of objects returned): "
46 print len(response)
47
48 print ""
49 print "print the first object returned in JSON format:"
50 out = response.to_json(response[0])
51 if len(out.splitlines()) > 15:
52     out = out.splitlines()[0:15]
53     out.append('..trimmed for brevity..')
54     out = '\n'.join(out)
55
56 print out

```

### Output from Python Code

```

1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
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  # Path to lib directory which contains pytan package
2  PYTAN_LIB_PATH = '../lib'
3
4  # connection info for Tanium Server
5  USERNAME = "Tanium User"
6  PASSWORD = "T@n!um"
7  HOST = "172.16.31.128"
8  PORT = "444"
9
10 # Logging conrols
11 LOGLEVEL = 2
12 DEBUGFORMAT = False
13
14 import sys, tempfile
15 sys.path.append(PYTAN_LIB_PATH)
16
17 import pytan
18 handler = pytan.Handler(
19     username=USERNAME,
20     password=PASSWORD,
21     host=HOST,
22     port=PORT,
23     loglevel=LOGLEVEL,
24     debugformat=DEBUGFORMAT,
25 )
26
27 print handler
28
29 # setup the arguments for the handler method
30 kwargs = {}
31 kwargs["objtype"] = u'package'
32 kwargs["name"] = u'Distribute Patch Tools'
33
34 # call the handler with the get method, passing in kwargs for arguments
35 response = handler.get(**kwargs)
36
37 print ""
38 print "Type of response: ", type(response)
39
```

```

40 print ""
41 print "print of response:"
42 print response
43
44 print ""
45 print "length of response (number of objects returned): "
46 print len(response)
47
48 print ""
49 print "print the first object returned in JSON format:"
50 out = response.to_json(response[0])
51 if len(out.splitlines()) > 15:
52     out = out.splitlines()[0:15]
53     out.append('..trimmed for brevity..')
54     out = '\n'.join(out)
55
56 print out

```

### Output from Python Code

```

1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
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": "2014-12-08T19:25:53",
15     "command": "cmd /c cscript //T:1800 copy-to-tanium-dir.vbs \"Tools\\",
16     "command_timeout": 1800,
17     "creation_time": "2014-12-08T19:21:06",
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  # Path to lib directory which contains pytan package
2  PYTAN_LIB_PATH = '../lib'
3
4  # connection info for Tanium Server
5  USERNAME = "Tanium User"
6  PASSWORD = "T@n!um"
7  HOST = "172.16.31.128"
8  PORT = "444"
9
10 # Logging conrols
11 LOGLEVEL = 2
12 DEBUGFORMAT = False
13
14 import sys, tempfile
15 sys.path.append(PYTAN_LIB_PATH)
16
17 import pytan
18 handler = pytan.Handler(
19     username=USERNAME,
20     password=PASSWORD,
21     host=HOST,
22     port=PORT,
23     loglevel=LOGLEVEL,
24     debugformat=DEBUGFORMAT,
25 )
26
27 print handler
28
29 # setup the arguments for the handler method
30 kwargs = {}
31 kwargs["objtype"] = u'sensor'
32 kwargs["name"] = [u'Computer Name', u'Action Statuses']
33
34 # call the handler with the get method, passing in kwargs for arguments
35 response = handler.get(**kwargs)
36
37 print ""
38 print "Type of response: ", type(response)
39
40 print ""
41 print "print of response:"
42 print response
43
44 print ""
45 print "length of response (number of objects returned): "
46 print len(response)
47
48 print ""
49 print "print the first object returned in JSON format:"
50 out = response.to_json(response[0])
51 if len(out.splitlines()) > 15:
52     out = out.splitlines()[0:15]
53     out.append('..trimmed for brevity..')
54     out = '\n'.join(out)
55
56 print out
```



## Output from Python Code

```

1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
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 # Path to lib directory which contains pytan package
2 PYTAN_LIB_PATH = '../lib'
3
4 # connection info for Tanium Server
5 USERNAME = "Tanium User"
6 PASSWORD = "T@n!um"
7 HOST = "172.16.31.128"
8 PORT = "444"
9
10 # Logging conrols
11 LOGLEVEL = 2
12 DEBUGFORMAT = False
13
14 import sys, tempfile
15 sys.path.append(PYTAN_LIB_PATH)
16
17 import pytan
18 handler = pytan.Handler(
19     username=USERNAME,
20     password=PASSWORD,

```

```
21     host=HOST,
22     port=PORT,
23     loglevel=LOGLEVEL,
24     debugformat=DEBUGFORMAT,
25 )
26
27 print handler
28
29 # setup the arguments for the handler method
30 kwargs = {}
31 kwargs["objtype"] = u'saved_question'
32 kwargs["name"] = u'Installed Applications'
33
34 # call the handler with the get method, passing in kwargs for arguments
35 response = handler.get(**kwargs)
36
37 print ""
38 print "Type of response: ", type(response)
39
40 print ""
41 print "print of response:"
42 print response
43
44 print ""
45 print "length of response (number of objects returned): "
46 print len(response)
47
48 print ""
49 print "print the first object returned in JSON format:"
50 out = response.to_json(response[0])
51 if len(out.splitlines()) > 15:
52     out = out.splitlines()[0:15]
53     out.append('..trimmed for brevity..')
54     out = '\n'.join(out)
55
56 print out
```

### Output from Python Code

```
1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
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  # Path to lib directory which contains pytan package
2  PYTAN_LIB_PATH = '../lib'
3
4  # connection info for Tanium Server
5  USERNAME = "Tanium User"
6  PASSWORD = "T@n!um"
7  HOST = "172.16.31.128"
8  PORT = "444"
9
10 # Logging conrols
11 LOGLEVEL = 2
12 DEBUGFORMAT = False
13
14 import sys, tempfile
15 sys.path.append(PYTAN_LIB_PATH)
16
17 import pytan
18 handler = pytan.Handler(
19     username=USERNAME,
20     password=PASSWORD,
21     host=HOST,
22     port=PORT,
23     loglevel=LOGLEVEL,
24     debugformat=DEBUGFORMAT,
25 )
26
27 print handler
28
29 # setup the arguments for the handler method
30 kwargs = {}
31 kwargs["objtype"] = u'user'
32 kwargs["id"] = 1
33
34 # call the handler with the get method, passing in kwargs for arguments
35 response = handler.get(**kwargs)
36
37 print ""
38 print "Type of response: ", type(response)
39

```

```
40 print ""
41 print "print of response:"
42 print response
43
44 print ""
45 print "length of response (number of objects returned): "
46 print len(response)
47
48 print ""
49 print "print the first object returned in JSON format:"
50 out = response.to_json(response[0])
51 if len(out.splitlines()) > 15:
52     out = out.splitlines()[0:15]
53     out.append('..trimmed for brevity..')
54     out = '\n'.join(out)
55
56 print out
```

### Output from Python Code

```
1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
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": "2014-12-08T19:28:09",
18     "name": "Jim Olsen",
19     "permissions": {
20         "_type": "permissions",
21         "permission": "admin"
22     },
23     "roles": {
24         "_type": "roles",
25         "role": [
26             {
27                 ..trimmed for brevity..
```

### Get sensor by name

Get a sensor by name

### Example Python Code

```

1  # Path to lib directory which contains pytan package
2  PYTAN_LIB_PATH = '../lib'
3
4  # connection info for Tanium Server
5  USERNAME = "Tanium User"
6  PASSWORD = "T@n!um"
7  HOST = "172.16.31.128"
8  PORT = "444"
9
10 # Logging controls
11 LOGLEVEL = 2
12 DEBUGFORMAT = False
13
14 import sys, tempfile
15 sys.path.append(PYTAN_LIB_PATH)
16
17 import pytan
18 handler = pytan.Handler(
19     username=USERNAME,
20     password=PASSWORD,
21     host=HOST,
22     port=PORT,
23     loglevel=LOGLEVEL,
24     debugformat=DEBUGFORMAT,
25 )
26
27 print handler
28
29 # setup the arguments for the handler method
30 kwargs = {}
31 kwargs["objtype"] = u'sensor'
32 kwargs["name"] = u'Computer Name'
33
34 # call the handler with the get method, passing in kwargs for arguments
35 response = handler.get(**kwargs)
36
37 print ""
38 print "Type of response: ", type(response)
39
40 print ""
41 print "print of response:"
42 print response
43
44 print ""
45 print "length of response (number of objects returned): "
46 print len(response)
47
48 print ""
49 print "print the first object returned in JSON format:"
50 out = response.to_json(response[0])
51 if len(out.splitlines()) > 15:
52     out = out.splitlines()[0:15]
53     out.append('..trimmed for brevity..')
54     out = '\n'.join(out)
55
56 print out

```

## Output from Python Code

```
1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
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 # Path to lib directory which contains pytan package
2 PYTAN_LIB_PATH = '../lib'
3
4 # connection info for Tanium Server
5 USERNAME = "Tanium User"
6 PASSWORD = "T@n!um"
7 HOST = "172.16.31.128"
8 PORT = "444"
9
10 # Logging conrols
11 LOGLEVEL = 2
12 DEBUGFORMAT = False
13
14 import sys, tempfile
15 sys.path.append(PYTAN_LIB_PATH)
16
17 import pytan
18 handler = pytan.Handler(
19     username=USERNAME,
20     password=PASSWORD,
```

```

21     host=HOST,
22     port=PORT,
23     loglevel=LOGLEVEL,
24     debugformat=DEBUGFORMAT,
25 )
26
27 print handler
28
29 # setup the arguments for the handler method
30 kwargs = {}
31 kwargs["objtype"] = u'saved_action'
32 kwargs["name"] = u'Distribute Tanium Standard Utilities'
33
34 # call the handler with the get method, passing in kwargs for arguments
35 response = handler.get(**kwargs)
36
37 print ""
38 print "Type of response: ", type(response)
39
40 print ""
41 print "print of response:"
42 print response
43
44 print ""
45 print "length of response (number of objects returned): "
46 print len(response)
47
48 print ""
49 print "print the first object returned in JSON format:"
50 out = response.to_json(response[0])
51 if len(out.splitlines()) > 15:
52     out = out.splitlines()[0:15]
53     out.append('..trimmed for brevity..')
54     out = '\n'.join(out)
55
56 print out

```

### Output from Python Code

```

1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
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": "2014-12-08T19:22:36",
17     "distribute_seconds": 0,
18     "end_time": "Never",

```

```
19     "expire_seconds": 660,
20     "id": 14,
21     "issue_count": 0,
22     "issue_seconds": 86400,
23     "last_action": {
24         "_type": "action",
25         "id": 4294967295,
26         "start_time": "Never"
27     } ..trimmed for brevity..
```

## Get all users

Get all users

## Example Python Code

```
1  # Path to lib directory which contains pytan package
2  PYTAN_LIB_PATH = '../lib'
3
4  # connection info for Tanium Server
5  USERNAME = "Tanium User"
6  PASSWORD = "T@n!um"
7  HOST = "172.16.31.128"
8  PORT = "444"
9
10 # Logging conrols
11 LOGLEVEL = 2
12 DEBUGFORMAT = False
13
14 import sys, tempfile
15 sys.path.append(PYTAN_LIB_PATH)
16
17 import pytan
18 handler = pytan.Handler(
19     username=USERNAME,
20     password=PASSWORD,
21     host=HOST,
22     port=PORT,
23     loglevel=LOGLEVEL,
24     debugformat=DEBUGFORMAT,
25 )
26
27 print handler
28
29 # setup the arguments for the handler method
30 kwargs = {}
31 kwargs["objtype"] = u'user'
32
33 # call the handler with the get_all method, passing in kwargs for arguments
34 response = handler.get_all(**kwargs)
35
36 print ""
37 print "Type of response: ", type(response)
38
39 print ""
```



```

40 print "print of response:"
41 print response
42
43 print ""
44 print "length of response (number of objects returned): "
45 print len(response)
46
47 print ""
48 print "print the first object returned in JSON format:"
49 out = response.to_json(response[0])
50 if len(out.splitlines()) > 15:
51     out = out.splitlines()[0:15]
52     out.append('..trimmed for brevity..')
53     out = '\n'.join(out)
54
55 print out

```

### Output from Python Code

```

1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
2
3 Type of response: <class 'taniumpy.object_types.user_list.UserList'>
4
5 print of response:
6 UserList, len: 6
7
8 length of response (number of objects returned):
9 6
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": "2014-12-08T19:28:09",
18     "name": "Jim Olsen",
19     "permissions": {
20         "_type": "permissions",
21         "permission": "admin"
22     },
23     "roles": {
24         "_type": "roles",
25         "role": [
26             {
27 ..trimmed for brevity..

```

### Get all saved actions

Get all saved actions

### Example Python Code

```
1  # Path to lib directory which contains pytan package
2  PYTAN_LIB_PATH = '../lib'
3
4  # connection info for Tanium Server
5  USERNAME = "Tanium User"
6  PASSWORD = "T@n!um"
7  HOST = "172.16.31.128"
8  PORT = "444"
9
10 # Logging conrols
11 LOGLEVEL = 2
12 DEBUGFORMAT = False
13
14 import sys, tempfile
15 sys.path.append(PYTAN_LIB_PATH)
16
17 import pytan
18 handler = pytan.Handler(
19     username=USERNAME,
20     password=PASSWORD,
21     host=HOST,
22     port=PORT,
23     loglevel=LOGLEVEL,
24     debugformat=DEBUGFORMAT,
25 )
26
27 print handler
28
29 # setup the arguments for the handler method
30 kwargs = {}
31 kwargs["objtype"] = u'saved_action'
32
33 # call the handler with the get_all method, passing in kwargs for arguments
34 response = handler.get_all(**kwargs)
35
36 print ""
37 print "Type of response: ", type(response)
38
39 print ""
40 print "print of response:"
41 print response
42
43 print ""
44 print "length of response (number of objects returned): "
45 print len(response)
46
47 print ""
48 print "print the first object returned in JSON format:"
49 out = response.to_json(response[0])
50 if len(out.splitlines()) > 15:
51     out = out.splitlines()[0:15]
52     out.append('...trimmed for brevity..')
53     out = '\n'.join(out)
54
55 print out
```

## Output from Python Code

```

1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
2
3 Type of response: <class 'taniumpy.object_types.saved_action_list.SavedActionList'>
4
5 print of response:
6 SavedActionList, len: 38
7
8 length of response (number of objects returned):
9 38
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": "2014-12-08T19:22:33",
18     "distribute_seconds": 600,
19     "end_time": "Never",
20     "expire_seconds": 1800,
21     "id": 1,
22     "issue_count": 3,
23     "issue_seconds": 3600,
24     "last_action": {
25         "_type": "action",
26         "id": 45,
27     ..trimmed for brevity..

```

## Get all settings

Get all system settings

## Example Python Code

```

1 # Path to lib directory which contains pytan package
2 PYTAN_LIB_PATH = '../lib'
3
4 # connection info for Tanium Server
5 USERNAME = "Tanium User"
6 PASSWORD = "T@n!um"
7 HOST = "172.16.31.128"
8 PORT = "444"
9
10 # Logging conrols
11 LOGLEVEL = 2
12 DEBUGFORMAT = False
13
14 import sys, tempfile
15 sys.path.append(PYTAN_LIB_PATH)
16
17 import pytan
18 handler = pytan.Handler(
19     username=USERNAME,
20     password=PASSWORD,
21     host=HOST,

```

```
22     port=PORT,
23     loglevel=LOGLEVEL,
24     debugformat=DEBUGFORMAT,
25 )
26
27 print handler
28
29 # setup the arguments for the handler method
30 kwargs = {}
31 kwargs["objtype"] = u'setting'
32
33 # call the handler with the get_all method, passing in kwargs for arguments
34 response = handler.get_all(**kwargs)
35
36 print ""
37 print "Type of response: ", type(response)
38
39 print ""
40 print "print of response:"
41 print response
42
43 print ""
44 print "length of response (number of objects returned): "
45 print len(response)
46
47 print ""
48 print "print the first object returned in JSON format:"
49 out = response.to_json(response[0])
50 if len(out.splitlines()) > 15:
51     out = out.splitlines()[0:15]
52     out.append('..trimmed for brevity..')
53     out = '\n'.join(out)
54
55 print out
```

### Output from Python Code

```
1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
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     "cache_row_id": 0,
15     "default_value": "0",
16     "hidden_flag": 0,
17     "id": 1,
18     "name": "load_initial_content",
19     "read_only_flag": 0,
20     "setting_type": "Server",
```

```

21     "value": "0",
22     "value_type": "Numeric"
23 }

```

### Get all saved questions

Get all saved questions

### Example Python Code

```

1  # Path to lib directory which contains pytan package
2  PYTAN_LIB_PATH = '../lib'
3
4  # connection info for Tanium Server
5  USERNAME = "Tanium User"
6  PASSWORD = "T@n!um"
7  HOST = "172.16.31.128"
8  PORT = "444"
9
10 # Logging conrols
11 LOGLEVEL = 2
12 DEBUGFORMAT = False
13
14 import sys, tempfile
15 sys.path.append(PYTAN_LIB_PATH)
16
17 import pytan
18 handler = pytan.Handler(
19     username=USERNAME,
20     password=PASSWORD,
21     host=HOST,
22     port=PORT,
23     loglevel=LOGLEVEL,
24     debugformat=DEBUGFORMAT,
25 )
26
27 print handler
28
29 # setup the arguments for the handler method
30 kwargs = {}
31 kwargs["objtype"] = u'saved_question'
32
33 # call the handler with the get_all method, passing in kwargs for arguments
34 response = handler.get_all(**kwargs)
35
36 print ""
37 print "Type of response: ", type(response)
38
39 print ""
40 print "print of response:"
41 print response
42
43 print ""
44 print "length of response (number of objects returned): "
45 print len(response)

```

```
46
47 print ""
48 print "print the first object returned in JSON format:"
49 out = response.to_json(response[0])
50 if len(out.splitlines()) > 15:
51     out = out.splitlines()[0:15]
52     out.append('..trimmed for brevity..')
53     out = '\n'.join(out)
54
55 print out
```

### Output from Python Code

```
1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
2
3 Type of response: <class 'taniumpy.object_types.saved_question_list.SavedQuestionList'>
4
5 print of response:
6 SavedQuestionList, len: 176
7
8 length of response (number of objects returned):
9 176
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 # Path to lib directory which contains pytan package
2 PYTAN_LIB_PATH = '../lib'
3
4 # connection info for Tanium Server
5 USERNAME = "Tanium User"
6 PASSWORD = "T@n!um"
7 HOST = "172.16.31.128"
```

```

8  PORT = "444"
9
10 # Logging controls
11 LOGLEVEL = 2
12 DEBUGFORMAT = False
13
14 import sys, tempfile
15 sys.path.append(PYTAN_LIB_PATH)
16
17 import pytan
18 handler = pytan.Handler(
19     username=USERNAME,
20     password=PASSWORD,
21     host=HOST,
22     port=PORT,
23     loglevel=LOGLEVEL,
24     debugformat=DEBUGFORMAT,
25 )
26
27 print handler
28
29 # setup the arguments for the handler method
30 kwargs = {}
31 kwargs["objtype"] = u'userrole'
32
33 # call the handler with the get_all method, passing in kwargs for arguments
34 response = handler.get_all(**kwargs)
35
36 print ""
37 print "Type of response: ", type(response)
38
39 print ""
40 print "print of response:"
41 print response
42
43 print ""
44 print "length of response (number of objects returned): "
45 print len(response)
46
47 print ""
48 print "print the first object returned in JSON format:"
49 out = response.to_json(response[0])
50 if len(out.splitlines()) > 15:
51     out = out.splitlines()[0:15]
52     out.append('..trimmed for brevity..')
53     out = '\n'.join(out)
54
55 print out

```

### Output from Python Code

```

1  Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
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
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  # Path to lib directory which contains pytan package
2  PYTAN_LIB_PATH = '../lib'
3
4  # connection info for Tanium Server
5  USERNAME = "Tanium User"
6  PASSWORD = "T@n!um"
7  HOST = "172.16.31.128"
8  PORT = "444"
9
10 # Logging conrols
11 LOGLEVEL = 2
12 DEBUGFORMAT = False
13
14 import sys, tempfile
15 sys.path.append(PYTAN_LIB_PATH)
16
17 import pytan
18 handler = pytan.Handler(
19     username=USERNAME,
20     password=PASSWORD,
21     host=HOST,
22     port=PORT,
23     loglevel=LOGLEVEL,
24     debugformat=DEBUGFORMAT,
25 )
26
27 print handler
28
29 # setup the arguments for the handler method
30 kwargs = {}
31 kwargs["objtype"] = u'question'
32
33 # call the handler with the get_all method, passing in kwargs for arguments
```



```

34 response = handler.get_all(**kwargs)
35
36 print ""
37 print "Type of response: ", type(response)
38
39 print ""
40 print "print of response:"
41 print response
42
43 print ""
44 print "length of response (number of objects returned): "
45 print len(response)
46
47 print ""
48 print "print the first object returned in JSON format:"
49 out = response.to_json(response[0])
50 if len(out.splitlines()) > 15:
51     out = out.splitlines()[0:15]
52     out.append('..trimmed for brevity..')
53     out = '\n'.join(out)
54
55 print out

```

### Output from Python Code

```

1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
2
3 Type of response:  <class 'taniumpy.object_types.question_list.QuestionList'>
4
5 print of response:
6 QuestionList, len: 255
7
8 length of response (number of objects returned):
9 255
10
11 print the first object returned in JSON format:
12 {
13     "_type": "question",
14     "action_tracking_flag": 0,
15     "cache_row_id": 0,
16     "context_group": {
17         "_type": "group",
18         "id": 0
19     },
20     "expiration": "2014-12-08T19:30:12",
21     "expire_seconds": 600,
22     "hidden_flag": 0,
23     "id": 1,
24     "management_rights_group": {
25         "_type": "group",
26         "id": 0
27     }
28 }
29 ..trimmed for brevity..

```

## Get all groups

Get all groups

### Example Python Code

```
1  # Path to lib directory which contains pytan package
2  PYTAN_LIB_PATH = '../lib'
3
4  # connection info for Tanium Server
5  USERNAME = "Tanium User"
6  PASSWORD = "T@n!um"
7  HOST = "172.16.31.128"
8  PORT = "444"
9
10 # Logging controls
11 LOGLEVEL = 2
12 DEBUGFORMAT = False
13
14 import sys, tempfile
15 sys.path.append(PYTAN_LIB_PATH)
16
17 import pytan
18 handler = pytan.Handler(
19     username=USERNAME,
20     password=PASSWORD,
21     host=HOST,
22     port=PORT,
23     loglevel=LOGLEVEL,
24     debugformat=DEBUGFORMAT,
25 )
26
27 print handler
28
29 # setup the arguments for the handler method
30 kwargs = {}
31 kwargs["objtype"] = u'group'
32
33 # call the handler with the get_all method, passing in kwargs for arguments
34 response = handler.get_all(**kwargs)
35
36 print ""
37 print "Type of response: ", type(response)
38
39 print ""
40 print "print of response:"
41 print response
42
43 print ""
44 print "length of response (number of objects returned): "
45 print len(response)
46
47 print ""
48 print "print the first object returned in JSON format:"
49 out = response.to_json(response[0])
50 if len(out.splitlines()) > 15:
51     out = out.splitlines()[0:15]
52     out.append('..trimmed for brevity..')
```

```

53     out = '\n'.join(out)
54
55 print out

```

### Output from Python Code

```

1  Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
2
3  Type of response: <class 'taniumpy.object_types.group_list.GroupList'>
4
5  print of response:
6  GroupList, 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": "group",
14     "and_flag": 0,
15     "deleted_flag": 0,
16     "filters": {
17         "_type": "filters",
18         "filter": []
19     },
20     "id": 1,
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  # Path to lib directory which contains pytan package
2  PYTAN_LIB_PATH = '../lib'
3
4  # connection info for Tanium Server
5  USERNAME = "Tanium User"
6  PASSWORD = "T@n!um"
7  HOST = "172.16.31.128"
8  PORT = "444"
9
10 # Logging conrols
11 LOGLEVEL = 2
12 DEBUGFORMAT = False
13
14 import sys, tempfile

```

```
15 sys.path.append(PYTAN_LIB_PATH)
16
17 import pytan
18 handler = pytan.Handler(
19     username=USERNAME,
20     password=PASSWORD,
21     host=HOST,
22     port=PORT,
23     loglevel=LOGLEVEL,
24     debugformat=DEBUGFORMAT,
25 )
26
27 print handler
28
29 # setup the arguments for the handler method
30 kwargs = {}
31 kwargs["objtype"] = u'sensor'
32
33 # call the handler with the get_all method, passing in kwargs for arguments
34 response = handler.get_all(**kwargs)
35
36 print ""
37 print "Type of response: ", type(response)
38
39 print ""
40 print "print of response:"
41 print response
42
43 print ""
44 print "length of response (number of objects returned): "
45 print len(response)
46
47 print ""
48 print "print the first object returned in JSON format:"
49 out = response.to_json(response[0])
50 if len(out.splitlines()) > 15:
51     out = out.splitlines()[0:15]
52     out.append('..trimmed for brevity..')
53     out = '\n'.join(out)
54
55 print out
```

### Output from Python Code

```
1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
2
3 Type of response: <class 'taniumpy.object_types.sensor_list.SensorList'>
4
5 print of response:
6 SensorList, len: 536
7
8 length of response (number of objects returned):
9 536
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  # Path to lib directory which contains pytan package
2  PYTAN_LIB_PATH = '../lib'
3
4  # connection info for Tanium Server
5  USERNAME = "Tanium User"
6  PASSWORD = "T@n!um"
7  HOST = "172.16.31.128"
8  PORT = "444"
9
10 # Logging conrols
11 LOGLEVEL = 2
12 DEBUGFORMAT = False
13
14 import sys, tempfile
15 sys.path.append(PYTAN_LIB_PATH)
16
17 import pytan
18 handler = pytan.Handler(
19     username=USERNAME,
20     password=PASSWORD,
21     host=HOST,
22     port=PORT,
23     loglevel=LOGLEVEL,
24     debugformat=DEBUGFORMAT,
25 )
26
27 print handler
28
29 # setup the arguments for the handler method
30 kwargs = {}
31 kwargs["objtype"] = u'whitelisted_url'
32
33 # call the handler with the get_all method, passing in kwargs for arguments
34 response = handler.get_all(**kwargs)

```

```
35
36 print ""
37 print "Type of response: ", type(response)
38
39 print ""
40 print "print of response:"
41 print response
42
43 print ""
44 print "length of response (number of objects returned): "
45 print len(response)
46
47 print ""
48 print "print the first object returned in JSON format:"
49 out = response.to_json(response[0])
50 if len(out.splitlines()) > 15:
51     out = out.splitlines()[0:15]
52     out.append('..trimmed for brevity..')
53     out = '\n'.join(out)
54
55 print out
```

### Output from Python Code

```
1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
2
3 Type of response: <class 'taniumpy.object_types.white_listed_url_list.WhiteListedUrlList'>
4
5 print of response:
6 WhiteListedUrlList, len: 46
7
8 length of response (number of objects returned):
9 46
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 # Path to lib directory which contains pytan package
2 PYTAN_LIB_PATH = '../lib'
3
4 # connection info for Tanium Server
5 USERNAME = "Tanium User"
6 PASSWORD = "T@n!um"
```

```

7  HOST = "172.16.31.128"
8  PORT = "444"
9
10 # Logging conrols
11 LOGLEVEL = 2
12 DEBUGFORMAT = False
13
14 import sys, tempfile
15 sys.path.append(PYTAN_LIB_PATH)
16
17 import pytan
18 handler = pytan.Handler(
19     username=USERNAME,
20     password=PASSWORD,
21     host=HOST,
22     port=PORT,
23     loglevel=LOGLEVEL,
24     debugformat=DEBUGFORMAT,
25 )
26
27 print handler
28
29 # setup the arguments for the handler method
30 kwargs = {}
31 kwargs["objtype"] = u'client'
32
33 # call the handler with the get_all method, passing in kwargs for arguments
34 response = handler.get_all(**kwargs)
35
36 print ""
37 print "Type of response: ", type(response)
38
39 print ""
40 print "print of response:"
41 print response
42
43 print ""
44 print "length of response (number of objects returned): "
45 print len(response)
46
47 print ""
48 print "print the first object returned in JSON format:"
49 out = response.to_json(response[0])
50 if len(out.splitlines()) > 15:
51     out = out.splitlines()[0:15]
52     out.append('..trimmed for brevity..')
53     out = '\n'.join(out)
54
55 print out

```

### Output from Python Code

```

1  Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
2
3  Type of response:  <class 'taniumpy.object_types.system_status_list.SystemStatusList'>
4
5  print of response:

```

```
6 SystemStatusList, len: 6
7
8 length of response (number of objects returned):
9 6
10
11 print the first object returned in JSON format:
12 {
13     "_type": "client_status",
14     "cache_row_id": 0,
15     "computer_id": "660621737",
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": "2014-12-08T21:26:16",
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 # Path to lib directory which contains pytan package
2 PYTAN_LIB_PATH = '../lib'
3
4 # connection info for Tanium Server
5 USERNAME = "Tanium User"
6 PASSWORD = "T@n!um"
7 HOST = "172.16.31.128"
8 PORT = "444"
9
10 # Logging conrols
11 LOGLEVEL = 2
12 DEBUGFORMAT = False
13
14 import sys, tempfile
15 sys.path.append(PYTAN_LIB_PATH)
16
17 import pytan
18 handler = pytan.Handler(
19     username=USERNAME,
20     password=PASSWORD,
21     host=HOST,
22     port=PORT,
23     loglevel=LOGLEVEL,
24     debugformat=DEBUGFORMAT,
25 )
26
27 print handler
28
```



```

29 # setup the arguments for the handler method
30 kwargs = {}
31 kwargs["objtype"] = u'package'
32
33 # call the handler with the get_all method, passing in kwargs for arguments
34 response = handler.get_all(**kwargs)
35
36 print ""
37 print "Type of response: ", type(response)
38
39 print ""
40 print "print of response:"
41 print response
42
43 print ""
44 print "length of response (number of objects returned): "
45 print len(response)
46
47 print ""
48 print "print the first object returned in JSON format:"
49 out = response.to_json(response[0])
50 if len(out.splitlines()) > 15:
51     out = out.splitlines()[0:15]
52     out.append('..trimmed for brevity..')
53     out = '\n'.join(out)
54
55 print out

```

### Output from Python Code

```

1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
2
3 Type of response: <class 'taniumpy.object_types.package_spec_list.PackageSpecList'>
4
5 print of response:
6 PackageSpecList, len: 92
7
8 length of response (number of objects returned):
9 92
10
11 print the first object returned in JSON format:
12 {
13     "_type": "package_spec",
14     "available_time": "2014-12-08T19:21:15",
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": "2014-12-08T19:20:46",
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 | # Path to lib directory which contains pytan package
2 | PYTAN_LIB_PATH = '../lib'
3 |
4 | # connection info for Tanium Server
5 | USERNAME = "Tanium User"
6 | PASSWORD = "T@n!um"
7 | HOST = "172.16.31.128"
8 | PORT = "444"
9 |
10 | # Logging conrols
11 | LOGLEVEL = 2
12 | DEBUGFORMAT = False
13 |
14 | import sys, tempfile
15 | sys.path.append(PYTAN_LIB_PATH)
16 |
17 | import pytan
18 | handler = pytan.Handler(
19 |     username=USERNAME,
20 |     password=PASSWORD,
21 |     host=HOST,
22 |     port=PORT,
23 |     loglevel=LOGLEVEL,
24 |     debugformat=DEBUGFORMAT,
25 | )
26 |
27 | print handler
28 |
29 | # setup the arguments for the handler method
30 | kwargs = {}
31 | kwargs["objtype"] = u'action'
32 |
33 | # call the handler with the get_all method, passing in kwargs for arguments
34 | response = handler.get_all(**kwargs)
35 |
36 | print ""
37 | print "Type of response: ", type(response)
38 |
39 | print ""
40 | print "print of response:"
41 | print response
42 |
43 | print ""
44 | print "length of response (number of objects returned): "
45 | print len(response)
46 |
47 | print ""
```

```

48 print "print the first object returned in JSON format:"
49 out = response.to_json(response[0])
50 if len(out.splitlines()) > 15:
51     out = out.splitlines()[0:15]
52     out.append('..trimmed for brevity..')
53     out = '\n'.join(out)
54
55 print out

```

### Output from Python Code

```

1  Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
2
3  Type of response: <class 'taniumpy.object_types.action_list.ActionList'>
4
5  print of response:
6  ActionList, len: 30
7
8  length of response (number of objects returned):
9  30
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": "2014-12-08T19:26:36",
22     "distribute_seconds": 600,
23     "expiration_time": "2014-12-08T20:16:36",
24     "expire_seconds": 3000,
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  # Path to lib directory which contains pytan package
2  PYTAN_LIB_PATH = '../lib'
3
4  # connection info for Tanium Server
5  USERNAME = "Tanium User"
6  PASSWORD = "T@n!um"
7  HOST = "172.16.31.128"

```

```
8  PORT = "444"
9
10 # Logging conrols
11 LOGLEVEL = 2
12 DEBUGFORMAT = False
13
14 import sys, tempfile
15 sys.path.append(PYTAN_LIB_PATH)
16
17 import pytan
18 handler = pytan.Handler(
19     username=USERNAME,
20     password=PASSWORD,
21     host=HOST,
22     port=PORT,
23     loglevel=LOGLEVEL,
24     debugformat=DEBUGFORMAT,
25 )
26
27 print handler
28
29 # setup the arguments for the handler method
30 kwargs = {}
31 kwargs["objtype"] = u'action'
32 kwargs["name"] = u'Distribute Tanium Standard Utilities'
33
34
35 # call the handler with the get method, passing in kwargs for arguments
36 # this should throw an exception: pytan.utils.HandlerError
37 import traceback
38 try:
39     handler.get(**kwargs)
40 except Exception as e:
41     traceback.print_exc(file=sys.stdout)
```

### Output from Python Code

```
1  Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
2  Traceback (most recent call last):
3    File "<string>", line 39, in <module>
4    File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 1562, 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  # Path to lib directory which contains pytan package
2  PYTAN_LIB_PATH = '../lib'
3
4  # connection info for Tanium Server
```

```

5  USERNAME = "Tanium User"
6  PASSWORD = "T@n!um"
7  HOST = "172.16.31.128"
8  PORT = "444"
9
10 # Logging conrols
11 LOGLEVEL = 2
12 DEBUGFORMAT = False
13
14 import sys, tempfile
15 sys.path.append(PYTAN_LIB_PATH)
16
17 import pytan
18 handler = pytan.Handler(
19     username=USERNAME,
20     password=PASSWORD,
21     host=HOST,
22     port=PORT,
23     loglevel=LOGLEVEL,
24     debugformat=DEBUGFORMAT,
25 )
26
27 print handler
28
29 # setup the arguments for the handler method
30 kwargs = {}
31 kwargs["objtype"] = u'question'
32 kwargs["name"] = u'dweedle'
33
34
35 # call the handler with the get method, passing in kwargs for arguments
36 # this should throw an exception: pytan.utils.HandlerError
37 import traceback
38 try:
39     handler.get(**kwargs)
40 except Exception as e:
41     traceback.print_exc(file=sys.stdout)

```

### Output from Python Code

```

1  Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
2  Traceback (most recent call last):
3    File "<string>", line 39, in <module>
4    File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 1562, 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  # Path to lib directory which contains pytan package
2  PYTAN_LIB_PATH = '../lib'
3
4  # connection info for Tanium Server
5  USERNAME = "Tanium User"
6  PASSWORD = "T@n!um"
7  HOST = "172.16.31.128"
8  PORT = "444"
9
10 # Logging controls
11 LOGLEVEL = 2
12 DEBUGFORMAT = False
13
14 import sys, tempfile
15 sys.path.append(PYTAN_LIB_PATH)
16
17 import pytan
18 handler = pytan.Handler(
19     username=USERNAME,
20     password=PASSWORD,
21     host=HOST,
22     port=PORT,
23     loglevel=LOGLEVEL,
24     debugformat=DEBUGFORMAT,
25 )
26
27 print handler
28
29 # setup the arguments for the handler method
30 kwargs = {}
31 kwargs["run"] = True
32 kwargs["package"] = u'Distribute Tanium Standard Utilities'
33
34 # call the handler with the deploy_action_human method, passing in kwargs for arguments
35 response = handler.deploy_action_human(**kwargs)
36 import pprint, io
37
38 print ""
39 print "Type of response: ", type(response)
40
41 print ""
42 print "Pretty print of response:"
43 print pprint.pformat(response)
44
45 print ""
46 print "Print of action object: "
47 print response['action_object']
48
49 # create an IO stream to store CSV results to
50 out = io.BytesIO()
51
52 # if results were returned (i.e. get_results=True was one of the kwargs passed in):
53 if response['action_results']:
54     # call the write_csv() method to convert response to CSV and store it in out
55     response['action_results'].write_csv(out, response['action_results'])
56
57     print ""
58     print "CSV Results of response: "
```

```
59 print out.getvalue()
```

### Output from Python Code

```
1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
2 2014-12-08 16:26:39,899 INFO question_progress: Results 0% (Get Online = "True" from all machine
3 2014-12-08 16:26:44,920 INFO question_progress: Results 50% (Get Online = "True" from all machine
4 2014-12-08 16:26:49,936 INFO question_progress: Results 100% (Get Online = "True" from all machine
5 2014-12-08 16:26:50,011 INFO action_progress: Action Results Passed: 0% (API Deploy Distribute T
6 2014-12-08 16:26:51,045 INFO action_progress: Action Results Passed: 0% (API Deploy Distribute T
7 2014-12-08 16:26:52,079 INFO action_progress: Action Results Passed: 0% (API Deploy Distribute T
8 2014-12-08 16:26:53,112 INFO action_progress: Action Results Passed: 0% (API Deploy Distribute T
9 2014-12-08 16:26:54,152 INFO action_progress: Action Results Passed: 0% (API Deploy Distribute T
10 2014-12-08 16:26:55,185 INFO action_progress: Action Results Passed: 0% (API Deploy Distribute T
11 2014-12-08 16:26:56,219 INFO action_progress: Action Results Passed: 0% (API Deploy Distribute T
12 2014-12-08 16:26:57,251 INFO action_progress: Action Results Passed: 0% (API Deploy Distribute T
13 2014-12-08 16:26:58,286 INFO action_progress: Action Results Passed: 0% (API Deploy Distribute T
14 2014-12-08 16:26:59,329 INFO action_progress: Action Results Passed: 0% (API Deploy Distribute T
15 2014-12-08 16:27:00,372 INFO action_progress: Action Results Passed: 0% (API Deploy Distribute T
16 2014-12-08 16:27:01,482 INFO action_progress: Action Results Passed: 0% (API Deploy Distribute T
17 2014-12-08 16:27:02,523 INFO action_progress: Action Results Passed: 0% (API Deploy Distribute T
18 2014-12-08 16:27:03,559 INFO action_progress: Action Results Passed: 0% (API Deploy Distribute T
19 2014-12-08 16:27:04,603 INFO action_progress: Action Results Passed: 0% (API Deploy Distribute T
20 2014-12-08 16:27:05,647 INFO action_progress: Action Results Passed: 0% (API Deploy Distribute T
21 2014-12-08 16:27:06,688 INFO action_progress: Action Results Passed: 0% (API Deploy Distribute T
22 2014-12-08 16:27:07,731 INFO action_progress: Action Results Passed: 0% (API Deploy Distribute T
23 2014-12-08 16:27:08,770 INFO action_progress: Action Results Passed: 0% (API Deploy Distribute T
24 2014-12-08 16:27:09,813 INFO action_progress: Action Results Passed: 33% (API Deploy Distribute
25 2014-12-08 16:27:10,851 INFO action_progress: Action Results Passed: 33% (API Deploy Distribute
26 2014-12-08 16:27:11,889 INFO action_progress: Action Results Passed: 50% (API Deploy Distribute
27 2014-12-08 16:27:12,925 INFO action_progress: Action Results Passed: 50% (API Deploy Distribute
28 2014-12-08 16:27:13,968 INFO action_progress: Action Results Passed: 50% (API Deploy Distribute
29 2014-12-08 16:27:15,005 INFO action_progress: Action Results Passed: 83% (API Deploy Distribute
30 2014-12-08 16:27:16,041 INFO action_progress: Action Results Passed: 100% (API Deploy Distribute
31 2014-12-08 16:27:16,077 INFO action_progress: Action Results Completed: 100% (API Deploy Distribute
32 2014-12-08 16:27:16,077 INFO action_progress: API Deploy Distribute Tanium Standard Utilities Re
33 Running Count: 0
34 Success Count: 6
35 Failed Count: 0
36 Unknown Count: 0
37 Finished Count: 6
38 Total Count: 6
39 Finished Count must equal: 6
40
41 Type of response: <type 'dict'>
42
43 Pretty print of response:
44 {'action_object': <taniumpy.object_types.action.Action object at 0x102967d10>,
45 'action_progress_human': 'API Deploy Distribute Tanium Standard Utilities Result Counts:\n\tRunning
46 'action_progress_map': {'Completed.': ['Casus-Belli.local',
47 'jtanium1.localdomain',
48 'ubuntu.(none)',
49 'localhost.(none)',
50 'Jims-Mac.local',
51 'WIN-A12SC6N6T7Q']},
52 'action_results': <taniumpy.object_types.result_set.ResultSet object at 0x1025a2050>,
53 'pre_action_question_results': {'question_object': <taniumpy.object_types.question.Question object
```

```
54         'question_results': <taniumpy.object_types.result_set.ResultSet object>
55
56     Print of action object:
57     Action, name: 'API Deploy Distribute Tanium Standard Utilities'
58
59     CSV Results of response:
60     Action Statuses,Computer Name
61     46:Completed.,Casus-Belli.local
62     46:Completed.,jtanium1.localdomain
63     46:Completed.,ubuntu.(none)
64     46:Completed.,localhost.(none)
65     46:Completed.,Jims-Mac.local
66     46:Completed.,WIN-A12SC6N6T7Q
```

### 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  # Path to lib directory which contains pytan package
2  PYTAN_LIB_PATH = '../lib'
3
4  # connection info for Tanium Server
5  USERNAME = "Tanium User"
6  PASSWORD = "T@n!um"
7  HOST = "172.16.31.128"
8  PORT = "444"
9
10 # Logging conrols
11 LOGLEVEL = 2
12 DEBUGFORMAT = False
13
14 import sys, tempfile
15 sys.path.append(PYTAN_LIB_PATH)
16
17 import pytan
18 handler = pytan.Handler(
19     username=USERNAME,
20     password=PASSWORD,
21     host=HOST,
22     port=PORT,
23     loglevel=LOGLEVEL,
24     debugformat=DEBUGFORMAT,
25 )
26
27 print handler
28
29 # setup the arguments for the handler method
30 kwargs = {}
31 kwargs["get_results"] = False
32 kwargs["run"] = True
33 kwargs["package"] = u'Distribute Tanium Standard Utilities'
34
```



```

35 # call the handler with the deploy_action_human method, passing in kwargs for arguments
36 response = handler.deploy_action_human(**kwargs)
37 import pprint, io
38
39 print ""
40 print "Type of response: ", type(response)
41
42 print ""
43 print "Pretty print of response:"
44 print pprint.pformat(response)
45
46 print ""
47 print "Print of action object: "
48 print response['action_object']
49
50 # create an IO stream to store CSV results to
51 out = io.BytesIO()
52
53 # if results were returned (i.e. get_results=True was one of the kwargs passed in):
54 if response['action_results']:
55     # call the write_csv() method to convert response to CSV and store it in out
56     response['action_results'].write_csv(out, response['action_results'])
57
58     print ""
59     print "CSV Results of response: "
60     print out.getvalue()

```

### Output from Python Code

```

1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
2 2014-12-08 16:27:16,199 INFO      question_progress: Results 0% (Get Online = "True" from all machine
3 2014-12-08 16:27:21,218 INFO      question_progress: Results 100% (Get Online = "True" from all machi
4
5 Type of response:  <type 'dict'>
6
7 Pretty print of response:
8 {'action_object': <taniumpy.object_types.action.Action object at 0x10211d590>,
9  'action_progress_human': None,
10 'action_progress_map': None,
11 'action_results': None,
12 'pre_action_question_results': {'question_object': <taniumpy.object_types.question.Question object
13                                'question_results': <taniumpy.object_types.result_set.ResultSet obj
14
15 Print of action object:
16 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 # Path to lib directory which contains pytan package
2 PYTAN_LIB_PATH = '../lib'

```

```
3
4 # connection info for Tanium Server
5 USERNAME = "Tanium User"
6 PASSWORD = "T@n!um"
7 HOST = "172.16.31.128"
8 PORT = "444"
9
10 # Logging conrols
11 LOGLEVEL = 2
12 DEBUGFORMAT = False
13
14 import sys, tempfile
15 sys.path.append(PYTAN_LIB_PATH)
16
17 import pytan
18 handler = pytan.Handler(
19     username=USERNAME,
20     password=PASSWORD,
21     host=HOST,
22     port=PORT,
23     loglevel=LOGLEVEL,
24     debugformat=DEBUGFORMAT,
25 )
26
27 print handler
28
29 # setup the arguments for the handler method
30 kwargs = {}
31 kwargs["run"] = True
32 kwargs["action_filters"] = u'Operating System, that contains:Windows'
33 kwargs["package"] = u'Distribute Tanium Standard Utilities'
34
35 # call the handler with the deploy_action_human method, passing in kwargs for arguments
36 response = handler.deploy_action_human(**kwargs)
37 import pprint, io
38
39 print ""
40 print "Type of response: ", type(response)
41
42 print ""
43 print "Pretty print of response:"
44 print pprint.pformat(response)
45
46 print ""
47 print "Print of action object: "
48 print response['action_object']
49
50 # create an IO stream to store CSV results to
51 out = io.BytesIO()
52
53 # if results were returned (i.e. get_results=True was one of the kwargs passed in):
54 if response['action_results']:
55     # call the write_csv() method to convert response to CSV and store it in out
56     response['action_results'].write_csv(out, response['action_results'])
57
58     print ""
59     print "CSV Results of response: "
```

```
60 print out.getvalue()
```

### Output from Python Code

```
1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
2 2014-12-08 16:27:21,364 INFO question_progress: Results 0% (Get Online = "True" from all machine
3 2014-12-08 16:27:26,389 INFO question_progress: Results 0% (Get Online = "True" from all machine
4 2014-12-08 16:27:31,409 INFO question_progress: Results 100% (Get Online = "True" from all machi
5 2014-12-08 16:27:31,496 INFO action_progress: Action Results Passed: 0% (API Deploy Distribute T
6 2014-12-08 16:27:32,528 INFO action_progress: Action Results Passed: 0% (API Deploy Distribute T
7 2014-12-08 16:27:33,561 INFO action_progress: Action Results Passed: 0% (API Deploy Distribute T
8 2014-12-08 16:27:34,607 INFO action_progress: Action Results Passed: 0% (API Deploy Distribute T
9 2014-12-08 16:27:35,649 INFO action_progress: Action Results Passed: 0% (API Deploy Distribute T
10 2014-12-08 16:27:36,707 INFO action_progress: Action Results Passed: 0% (API Deploy Distribute T
11 2014-12-08 16:27:37,764 INFO action_progress: Action Results Passed: 0% (API Deploy Distribute T
12 2014-12-08 16:27:38,800 INFO action_progress: Action Results Passed: 0% (API Deploy Distribute T
13 2014-12-08 16:27:39,830 INFO action_progress: Action Results Passed: 0% (API Deploy Distribute T
14 2014-12-08 16:27:40,867 INFO action_progress: Action Results Passed: 0% (API Deploy Distribute T
15 2014-12-08 16:27:41,904 INFO action_progress: Action Results Passed: 0% (API Deploy Distribute T
16 2014-12-08 16:27:42,942 INFO action_progress: Action Results Passed: 0% (API Deploy Distribute T
17 2014-12-08 16:27:43,986 INFO action_progress: Action Results Passed: 0% (API Deploy Distribute T
18 2014-12-08 16:27:45,091 INFO action_progress: Action Results Passed: 0% (API Deploy Distribute T
19 2014-12-08 16:27:46,143 INFO action_progress: Action Results Passed: 0% (API Deploy Distribute T
20 2014-12-08 16:27:47,186 INFO action_progress: Action Results Passed: 0% (API Deploy Distribute T
21 2014-12-08 16:27:48,222 INFO action_progress: Action Results Passed: 0% (API Deploy Distribute T
22 2014-12-08 16:27:49,267 INFO action_progress: Action Results Passed: 0% (API Deploy Distribute T
23 2014-12-08 16:27:50,316 INFO action_progress: Action Results Passed: 0% (API Deploy Distribute T
24 2014-12-08 16:27:51,363 INFO action_progress: Action Results Passed: 0% (API Deploy Distribute T
25 2014-12-08 16:27:52,475 INFO action_progress: Action Results Passed: 0% (API Deploy Distribute T
26 2014-12-08 16:27:53,522 INFO action_progress: Action Results Passed: 100% (API Deploy Distribute
27 2014-12-08 16:27:53,563 INFO action_progress: Action Results Completed: 50% (API Deploy Distribu
28 2014-12-08 16:27:54,610 INFO action_progress: Action Results Completed: 50% (API Deploy Distribu
29 2014-12-08 16:27:55,648 INFO action_progress: Action Results Completed: 50% (API Deploy Distribu
30 2014-12-08 16:27:56,697 INFO action_progress: Action Results Completed: 50% (API Deploy Distribu
31 2014-12-08 16:27:58,074 INFO action_progress: Action Results Completed: 100% (API Deploy Distrib
32 2014-12-08 16:27:58,074 INFO action_progress: API Deploy Distribute Tanium Standard Utilities Re
33 Running Count: 0
34 Success Count: 2
35 Failed Count: 0
36 Unknown Count: 0
37 Finished Count: 2
38 Total Count: 2
39 Finished Count must equal: 2
40
41 Type of response: <type 'dict'>
42
43 Pretty print of response:
44 {'action_object': <taniumpy.object_types.action.Action object at 0x102120410>,
45 'action_progress_human': 'API Deploy Distribute Tanium Standard Utilities Result Counts:\n\tRunning
46 'action_progress_map': {'Completed.': ['jtanium1.localdomain',
47 'WIN-A12SC6N6T7Q']},
48 'action_results': <taniumpy.object_types.result_set.ResultSet object at 0x102a047d0>,
49 'pre_action_question_results': {'question_object': <taniumpy.object_types.question.Question object
50 'question_results': <taniumpy.object_types.result_set.ResultSet obj
51
52 Print of action object:
53 Action, name: 'API Deploy Distribute Tanium Standard Utilities'
```

```
54
55 CSV Results of response:
56 Action Statuses,Computer Name
57 48:Completed.,jtanium1.localdomain
58 48:Completed.,WIN-A12SC6N6T7Q
```

## 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  # Path to lib directory which contains pytan package
2  PYTAN_LIB_PATH = '../lib'
3
4  # connection info for Tanium Server
5  USERNAME = "Tanium User"
6  PASSWORD = "T@n!um"
7  HOST = "172.16.31.128"
8  PORT = "444"
9
10 # Logging conrols
11 LOGLEVEL = 2
12 DEBUGFORMAT = False
13
14 import sys, tempfile
15 sys.path.append(PYTAN_LIB_PATH)
16
17 import pytan
18 handler = pytan.Handler(
19     username=USERNAME,
20     password=PASSWORD,
21     host=HOST,
22     port=PORT,
23     loglevel=LOGLEVEL,
24     debugformat=DEBUGFORMAT,
25 )
26
27 print handler
28
29 # setup the arguments for the handler method
30 kwargs = {}
31 kwargs["run"] = True
32 kwargs["action_filters"] = u'Operating System, that contains:Windows'
33 kwargs["package"] = u'Custom Tagging - Add Tags($1=tag_should_be_added,$2=tag_should_be_ignore)'
34
35 # call the handler with the deploy_action_human method, passing in kwargs for arguments
36 response = handler.deploy_action_human(**kwargs)
37 import pprint, io
38
39 print ""
40 print "Type of response: ", type(response)
41
```

```

42 print ""
43 print "Pretty print of response:"
44 print pprint.pformat(response)
45
46 print ""
47 print "Print of action object: "
48 print response['action_object']
49
50 # create an IO stream to store CSV results to
51 out = io.BytesIO()
52
53 # if results were returned (i.e. get_results=True was one of the kwargs passed in):
54 if response['action_results']:
55     # call the write_csv() method to convert response to CSV and store it in out
56     response['action_results'].write_csv(out, response['action_results'])
57
58 print ""
59 print "CSV Results of response: "
60 print out.getvalue()

```

### Output from Python Code

```

1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
2 2014-12-08 16:27:58,284 INFO question_progress: Results 0% (Get Online = "True" from all machine
3 2014-12-08 16:28:03,310 INFO question_progress: Results 100% (Get Online = "True" from all machi
4 2014-12-08 16:28:03,416 INFO action_progress: Action Results Passed: 0% (API Deploy Custom Taggi
5 2014-12-08 16:28:04,454 INFO action_progress: Action Results Passed: 0% (API Deploy Custom Taggi
6 2014-12-08 16:28:05,489 INFO action_progress: Action Results Passed: 0% (API Deploy Custom Taggi
7 2014-12-08 16:28:06,521 INFO action_progress: Action Results Passed: 0% (API Deploy Custom Taggi
8 2014-12-08 16:28:07,561 INFO action_progress: Action Results Passed: 0% (API Deploy Custom Taggi
9 2014-12-08 16:28:08,602 INFO action_progress: Action Results Passed: 0% (API Deploy Custom Taggi
10 2014-12-08 16:28:09,633 INFO action_progress: Action Results Passed: 0% (API Deploy Custom Taggi
11 2014-12-08 16:28:10,667 INFO action_progress: Action Results Passed: 0% (API Deploy Custom Taggi
12 2014-12-08 16:28:11,704 INFO action_progress: Action Results Passed: 0% (API Deploy Custom Taggi
13 2014-12-08 16:28:12,738 INFO action_progress: Action Results Passed: 0% (API Deploy Custom Taggi
14 2014-12-08 16:28:13,773 INFO action_progress: Action Results Passed: 100% (API Deploy Custom Tag
15 2014-12-08 16:28:13,805 INFO action_progress: Action Results Completed: 0% (API Deploy Custom Ta
16 2014-12-08 16:28:14,848 INFO action_progress: Action Results Completed: 0% (API Deploy Custom Ta
17 2014-12-08 16:28:15,890 INFO action_progress: Action Results Completed: 0% (API Deploy Custom Ta
18 2014-12-08 16:28:16,935 INFO action_progress: Action Results Completed: 0% (API Deploy Custom Ta
19 2014-12-08 16:28:17,970 INFO action_progress: Action Results Completed: 0% (API Deploy Custom Ta
20 2014-12-08 16:28:19,006 INFO action_progress: Action Results Completed: 100% (API Deploy Custom
21 2014-12-08 16:28:19,006 INFO action_progress: API Deploy Custom Tagging - Add Tags Result Counts
22 Running Count: 0
23 Success Count: 2
24 Failed Count: 0
25 Unknown Count: 0
26 Finished Count: 2
27 Total Count: 2
28 Finished Count must equal: 2
29
30 Type of response: <type 'dict'>
31
32 Pretty print of response:
33 {'action_object': <taniumpy.object_types.action.Action object at 0x1029798d0>,
34 'action_progress_human': 'API Deploy Custom Tagging - Add Tags Result Counts:\n\tRunning Count: 0\n\t
35 'action_progress_map': {'Completed.': ['jtanium1.localdomain',

```

```
36         'WIN-A12SC6N6T7Q']]],
37     'action_results': <taniumpy.object_types.result_set.ResultSet object at 0x102a05fd0>,
38     'pre_action_question_results': {'question_object': <taniumpy.object_types.question.Question object
39                                     'question_results': <taniumpy.object_types.result_set.ResultSet obj
40
41 Print of action object:
42 Action, name: 'API Deploy Custom Tagging - Add Tags'
43
44 CSV Results of response:
45 Action Statuses,Computer Name
46 49:Completed.,jtanium1.localdomain
47 49:Completed.,WIN-A12SC6N6T7Q
```

## 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  # Path to lib directory which contains pytan package
2  PYTAN_LIB_PATH = '../lib'
3
4  # connection info for Tanium Server
5  USERNAME = "Tanium User"
6  PASSWORD = "T@n!um"
7  HOST = "172.16.31.128"
8  PORT = "444"
9
10 # Logging conrols
11 LOGLEVEL = 2
12 DEBUGFORMAT = False
13
14 import sys, tempfile
15 sys.path.append(PYTAN_LIB_PATH)
16
17 import pytan
18 handler = pytan.Handler(
19     username=USERNAME,
20     password=PASSWORD,
21     host=HOST,
22     port=PORT,
23     loglevel=LOGLEVEL,
24     debugformat=DEBUGFORMAT,
25 )
26
27 print handler
28
29 # setup the arguments for the handler method
30 kwargs = {}
31 kwargs['report_dir'] = tempfile.gettempdir()
32 kwargs["package"] = u'Distribute Tanium Standard Utilities'
33
```

```

34 # call the handler with the deploy_action_human method, passing in kwargs for arguments
35 # this should throw an exception: pytan.utils.RunFalse
36 import traceback
37 try:
38     handler.deploy_action_human(**kwargs)
39 except Exception as e:
40     traceback.print_exc(file=sys.stdout)
41

```

### Output from Python Code

```

1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
2 2014-12-08 16:28:19,131 INFO      question_progress: Results 0% (Get Computer Name and Online = "True
3 2014-12-08 16:28:24,148 INFO      question_progress: Results 67% (Get Computer Name and Online = "True
4 2014-12-08 16:28:29,164 INFO      question_progress: Results 100% (Get Computer Name and Online = "True
5 2014-12-08 16:28:29,185 INFO      handler: Report file '/var/folders/dk/vjrlr_c53yx6k6gzp2bbt_c40000gn/T/VERIFY_BEFORE_DEPLO
6 Traceback (most recent call last):
7   File "<string>", line 39, in <module>
8     File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 1172, in deploy_action_human
9         **kwargs
10    File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 1003, 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 # Path to lib directory which contains pytan package
2 PYTAN_LIB_PATH = '../lib'
3
4 # connection info for Tanium Server
5 USERNAME = "Tanium User"
6 PASSWORD = "T@n!um"
7 HOST = "172.16.31.128"
8 PORT = "444"
9
10 # Logging conrols
11 LOGLEVEL = 2
12 DEBUGFORMAT = False
13
14 import sys, tempfile
15 sys.path.append(PYTAN_LIB_PATH)
16
17 import pytan
18 handler = pytan.Handler(
19     username=USERNAME,
20     password=PASSWORD,
21     host=HOST,
22     port=PORT,

```

```
23     loglevel=LOGLEVEL,
24     debugformat=DEBUGFORMAT,
25 )
26
27 print handler
28
29 # setup the arguments for the handler method
30 kwargs = {}
31 kwargs['report_dir'] = tempfile.gettempdir()
32 kwargs["package_help"] = True
33
34
35 # call the handler with the deploy_action_human method, passing in kwargs for arguments
36 # this should throw an exception: pytan.utils.PytanHelp
37 import traceback
38 try:
39     handler.deploy_action_human(**kwargs)
40 except Exception as e:
41     traceback.print_exc(file=sys.stdout)
```

### Output from Python Code

```
1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
2 Traceback (most recent call last):
3   File "<string>", line 39, in <module>
4   File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 1136, 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 separated 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  # Path to lib directory which contains pytan package
2  PYTAN_LIB_PATH = '../lib'
3
4  # connection info for Tanium Server
5  USERNAME = "Tanium User"
6  PASSWORD = "T@n!um"
7  HOST = "172.16.31.128"
8  PORT = "444"
9
10 # Logging controls
11 LOGLEVEL = 2
12 DEBUGFORMAT = False
13
14 import sys, tempfile
15 sys.path.append(PYTAN_LIB_PATH)
16
17 import pytan
18 handler = pytan.Handler(
19     username=USERNAME,
20     password=PASSWORD,

```

```
21     host=HOST,
22     port=PORT,
23     loglevel=LOGLEVEL,
24     debugformat=DEBUGFORMAT,
25 )
26
27 print handler
28
29 # setup the arguments for the handler method
30 kwargs = {}
31 kwargs['report_dir'] = tempfile.gettempdir()
32 kwargs["run"] = True
33 kwargs["package"] = u'Invalid Package'
34
35
36 # call the handler with the deploy_action_human method, passing in kwargs for arguments
37 # this should throw an exception: pytan.utils.HandlerError
38 import traceback
39 try:
40     handler.deploy_action_human(**kwargs)
41 except Exception as e:
42     traceback.print_exc(file=sys.stdout)
```

### Output from Python Code

```
1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
2 Traceback (most recent call last):
3   File "<string>", line 40, in <module>
4   File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 1172, in deploy_action_human
5     **kwargs
6   File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 963, in deploy_action
7     package_def = self._get_package_def(package_def)
8   File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 1796, 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 1572, in get
11    return self._get_single(obj_map, **kwargs)
12  File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 1756, 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 1766, in _single_find
15    obj_ret = self._find(api_obj_single, **kwargs)
16  File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 1694, 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 # Path to lib directory which contains pytan package
2 PYTAN_LIB_PATH = '../lib'
3
4 # connection info for Tanium Server
```

```

5  USERNAME = "Tanium User"
6  PASSWORD = "T@n!um"
7  HOST = "172.16.31.128"
8  PORT = "444"
9
10 # Logging conrols
11 LOGLEVEL = 2
12 DEBUGFORMAT = False
13
14 import sys, tempfile
15 sys.path.append(PYTAN_LIB_PATH)
16
17 import pytan
18 handler = pytan.Handler(
19     username=USERNAME,
20     password=PASSWORD,
21     host=HOST,
22     port=PORT,
23     loglevel=LOGLEVEL,
24     debugformat=DEBUGFORMAT,
25 )
26
27 print handler
28
29 # setup the arguments for the handler method
30 kwargs = {}
31 kwargs['report_dir'] = tempfile.gettempdir()
32 kwargs["options_help"] = True
33
34
35 # call the handler with the deploy_action_human method, passing in kwargs for arguments
36 # this should throw an exception: pytan.utils.PytanHelp
37 import traceback
38 try:
39     handler.deploy_action_human(**kwargs)
40 except Exception as e:
41     traceback.print_exc(file=sys.stdout)

```

### Output from Python Code

```

1  Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
2  Traceback (most recent call last):
3      File "<string>", line 39, in <module>
4      File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 1142, 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

```

that require a value.

When options are supplied for question options, they must be supplied as 'OPTION' or 'OPTION:VALUE' for options that require a value.

Options can be used on 'filter' or 'group', where 'group' pertains to group filters or question filters. All 'filter' options are also applicable to 'group' for question options.

#### Valid Options

-----

##### 'ignore\_case'

Help: Make the filter do a case insensitive match

Usable on: filter

Example for sensor: "Sensor1, opt:ignore\_case"

Example for question: "ignore\_case"

##### 'match\_case'

Help: Make the filter do a case sensitive match

Usable on: filter

Example for sensor: "Sensor1, opt:match\_case"

Example for question: "match\_case"

##### 'match\_any\_value'

Help: Make the filter match any value

Usable on: filter

Example for sensor: "Sensor1, opt:match\_any\_value"

Example for question: "match\_any\_value"

##### 'match\_all\_values'

Help: Make the filter match all values

Usable on: filter

Example for sensor: "Sensor1, opt:match\_all\_values"

Example for question: "match\_all\_values"

##### 'max\_data\_age'

Help: Re-fetch cached values older than N seconds

Usable on: filter

VALUE description and type: seconds, <type 'int'>

Example for sensor: "Sensor1, opt:max\_data\_age:seconds"

Example for question: "max\_data\_age:seconds"

##### 'value\_type'

Help: Make the filter consider the value type as VALUE\_TYPE

Usable on: filter

VALUE description and type: value\_type, <type 'str'>

Example for sensor: "Sensor1, opt:value\_type:value\_type"

Example for question: "value\_type:value\_type"

##### 'and'

Help: Use 'and' for all of the filters supplied

Usable on: group

Example for sensor: "Sensor1, opt:and"

Example for question: "and"

##### '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  # Path to lib directory which contains pytan package
2  PYTAN_LIB_PATH = '../lib'
3
4  # connection info for Tanium Server
5  USERNAME = "Tanium User"
6  PASSWORD = "T@n!um"
7  HOST = "172.16.31.128"
8  PORT = "444"
9
10 # Logging conrols
11 LOGLEVEL = 2
12 DEBUGFORMAT = False
13
14 import sys, tempfile
15 sys.path.append(PYTAN_LIB_PATH)
16
17 import pytan
18 handler = pytan.Handler(
19     username=USERNAME,
20     password=PASSWORD,
21     host=HOST,
22     port=PORT,
23     loglevel=LOGLEVEL,
24     debugformat=DEBUGFORMAT,
25 )
26
27 print handler
28
29 # setup the arguments for the handler method
30 kwargs = {}
31 kwargs['report_dir'] = tempfile.gettempdir()
32 kwargs["run"] = True
33 kwargs["package"] = u''
34
35
36 # call the handler with the deploy_action_human method, passing in kwargs for arguments
37 # this should throw an exception: pytan.utils.HumanParserError
38 import traceback
39 try:
40     handler.deploy_action_human(**kwargs)
41 except Exception as e:
42     traceback.print_exc(file=sys.stdout)
```

### Output from Python Code

```
1  Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
2  Traceback (most recent call last):
3      File "<string>", line 40, in <module>
4      File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 1166, in deploy_action_human
5          package_def = utils.dehumanize_package(package)
```

```

6   File "/Users/jolsen/gh/pytan/lib/pytan/utils.py", line 1334, 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  # Path to lib directory which contains pytan package
2  PYTAN_LIB_PATH = '../lib'
3
4  # connection info for Tanium Server
5  USERNAME = "Tanium User"
6  PASSWORD = "T@n!um"
7  HOST = "172.16.31.128"
8  PORT = "444"
9
10 # Logging conrols
11 LOGLEVEL = 2
12 DEBUGFORMAT = False
13
14 import sys, tempfile
15 sys.path.append(PYTAN_LIB_PATH)
16
17 import pytan
18 handler = pytan.Handler(
19     username=USERNAME,
20     password=PASSWORD,
21     host=HOST,
22     port=PORT,
23     loglevel=LOGLEVEL,
24     debugformat=DEBUGFORMAT,
25 )
26
27 print handler
28
29 # setup the arguments for the handler method
30 kwargs = {}
31 kwargs['report_dir'] = tempfile.gettempdir()
32 kwargs["filters_help"] = True
33
34
35 # call the handler with the deploy_action_human method, passing in kwargs for arguments
36 # this should throw an exception: pytan.utils.PytanHelp
37 import traceback
38 try:
39     handler.deploy_action_human(**kwargs)
40 except Exception as e:
41     traceback.print_exc(file=sys.stdout)

```

### Output from Python Code

```
1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
2 Traceback (most recent call last):
3   File "<string>", line 39, in <module>
4   File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 1139, 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
12 the sensor returns. When filters are used for whole question filters,
13 they control what rows will be returned. They are used by Groups to
14 define group membership, deploy actions to determine which machines
15 should have the action deployed to it, and more.
16
17 A filter string is a human string that describes, a sensor followed
18 by ', that FILTER:VALUE', where FILTER is a valid filter string,
19 and VALUE is the string that you want FILTER to match on.
20
21 Valid Filters
22 -----
23
24 '<'
25     Help: Filter for less than VALUE
26     Example: "Sensor1, that <:VALUE"
27
28 'less'
29     Help: Filter for less than VALUE
30     Example: "Sensor1, that less:VALUE"
31
32 'lt'
33     Help: Filter for less than VALUE
34     Example: "Sensor1, that lt:VALUE"
35
36 'less than'
37     Help: Filter for less than VALUE
38     Example: "Sensor1, that less than:VALUE"
39
40 '!<'
41     Help: Filter for not less than VALUE
42     Example: "Sensor1, that !<:VALUE"
43
44 'notless'
45     Help: Filter for not less than VALUE
46     Example: "Sensor1, that notless:VALUE"
47
48 'not less'
49     Help: Filter for not less than VALUE
50     Example: "Sensor1, that not less:VALUE"
51
52 'not less than'
53     Help: Filter for not less than VALUE
54     Example: "Sensor1, that not less than:VALUE"
55
56 '<='
57     Help: Filter for less than or equal to VALUE
58     Example: "Sensor1, that <=:VALUE"
```



```

59 'less equal'
60     Help: Filter for less than or equal to VALUE
61     Example: "Sensor1, that less equal:VALUE"
62
63
64 'lessequal'
65     Help: Filter for less than or equal to VALUE
66     Example: "Sensor1, that lessequal:VALUE"
67
68 'le'
69     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 'notequals'
176     Help: Filter for not equals to VALUE
177     Example: "Sensor1, that notequals:VALUE"
178
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 're'
292     Help: Filter for regular expression match for VALUE
293     Example: "Sensor1, that re:VALUE"
294
295 '<'
296     Help: Filter for less than VALUE
297     Example: "Sensor1, that <:VALUE"
298
299 'less'
300     Help: Filter for less than VALUE
301     Example: "Sensor1, that less:VALUE"
302
303 'lt'
304     Help: Filter for less than VALUE
305     Example: "Sensor1, that lt:VALUE"
306
307 'less than'
308     Help: Filter for less than VALUE
309     Example: "Sensor1, that less than:VALUE"
310
311 '!<'
312     Help: Filter for not less than VALUE
313     Example: "Sensor1, that !<:VALUE"
314
315 'notless'
316     Help: Filter for not less than VALUE
317     Example: "Sensor1, that notless:VALUE"
318
319 'not less'
320     Help: Filter for not less than VALUE
321     Example: "Sensor1, that not less:VALUE"
322
323 'not less than'
324     Help: Filter for not less than VALUE
325     Example: "Sensor1, that not less than:VALUE"
326
327 '<='
328     Help: Filter for less than or equal to VALUE
329     Example: "Sensor1, that <=:VALUE"
330
331 'less equal'
332     Help: Filter for less than or equal to VALUE
333     Example: "Sensor1, that less equal:VALUE"
334
335 'lessequal'
336     Help: Filter for less than or equal to VALUE
337     Example: "Sensor1, that lessequal:VALUE"
338
339 'le'
340     Help: Filter for less than or equal to VALUE
341     Example: "Sensor1, that le:VALUE"
342
343 '!<='
344     Help: Filter for not less than or equal to VALUE
345     Example: "Sensor1, that !<=:VALUE"
346
347 'not less equal'
348

```

```
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 'not greater equal'
408     Help: Filter for not greater than VALUE
409     Example: "Sensor1, that not greater equal:VALUE"
410
411 'notgreaterequal'
412     Help: Filter for not greater than VALUE
413     Example: "Sensor1, that notgreaterequal:VALUE"
414
415 '='
416     Help: Filter for equals to VALUE
417     Example: "Sensor1, that =:VALUE"
418
419 'equal'
420     Help: Filter for equals to VALUE
421     Example: "Sensor1, that equal:VALUE"
422
423 'equals'
424     Help: Filter for equals to VALUE
425     Example: "Sensor1, that equals:VALUE"
426
427 'eq'
428     Help: Filter for equals to VALUE
429     Example: "Sensor1, that eq:VALUE"
430
431 '!='
432     Help: Filter for not equals to VALUE
433     Example: "Sensor1, that !=:VALUE"
434
435 'not equal'
436     Help: Filter for not equals to VALUE
437     Example: "Sensor1, that not equal:VALUE"
438
439 'notequal'
440     Help: Filter for not equals to VALUE
441     Example: "Sensor1, that notequal:VALUE"
442
443 'not equals'
444     Help: Filter for not equals to VALUE
445     Example: "Sensor1, that not equals:VALUE"
446
447 'notequals'
448     Help: Filter for not equals to VALUE
449     Example: "Sensor1, that notequals:VALUE"
450
451 'ne'
452     Help: Filter for not equals to VALUE
453     Example: "Sensor1, that ne:VALUE"
454
455 'contains'
456     Help: Filter for contains VALUE (adds .* before and after VALUE)
457     Example: "Sensor1, that contains:VALUE"
458
459 'does not contain'
460     Help: Filter for does not contain VALUE (adds .* before and after VALUE)
461     Example: "Sensor1, that does not contain:VALUE"
462
463 'doesnotcontain'
464

```

```
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     'is not'
524         Help: Filter for non regular expression match for VALUE
525         Example: "Sensor1, that is not:VALUE"
526
527
528     'not regex'
529         Help: Filter for non regular expression match for VALUE
530         Example: "Sensor1, that not regex:VALUE"
531
532
533     'notregex'
534         Help: Filter for non regular expression match for VALUE
535         Example: "Sensor1, that notregex:VALUE"
536
537
538     'not regex match'
539         Help: Filter for non regular expression match for VALUE
540         Example: "Sensor1, that not regex match:VALUE"
541
542
543     'notregexmatch'
544         Help: Filter for non regular expression match for VALUE
545         Example: "Sensor1, that notregexmatch:VALUE"
546
547
548     'nre'
549         Help: Filter for non regular expression match for VALUE
550         Example: "Sensor1, that nre:VALUE"
551
552
553     'is'
554         Help: Filter for regular expression match for VALUE
555         Example: "Sensor1, that is:VALUE"
556
557
558     'regex'
559         Help: Filter for regular expression match for VALUE
560         Example: "Sensor1, that regex:VALUE"
561
562
563     'regex match'
564         Help: Filter for regular expression match for VALUE
565         Example: "Sensor1, that regex match:VALUE"
566
567
568     'regexmatch'
569         Help: Filter for regular expression match for VALUE
570         Example: "Sensor1, that regexmatch:VALUE"
571
572
573     're'
574         Help: Filter for regular expression match for VALUE
575         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  # Path to lib directory which contains pytan package
2  PYTAN_LIB_PATH = '../lib'
3
4  # connection info for Tanium Server

```

```
5  USERNAME = "Tanium User"
6  PASSWORD = "T@n!um"
7  HOST = "172.16.31.128"
8  PORT = "444"
9
10 # Logging conrols
11 LOGLEVEL = 2
12 DEBUGFORMAT = False
13
14 import sys, tempfile
15 sys.path.append(PYTAN_LIB_PATH)
16
17 import pytan
18 handler = pytan.Handler(
19     username=USERNAME,
20     password=PASSWORD,
21     host=HOST,
22     port=PORT,
23     loglevel=LOGLEVEL,
24     debugformat=DEBUGFORMAT,
25 )
26
27 print handler
28
29 # setup the arguments for the handler method
30 kwargs = {}
31 kwargs['report_dir'] = tempfile.gettempdir()
32 kwargs["run"] = True
33 kwargs["package"] = u'Custom Tagging - Add Tags'
34
35
36 # call the handler with the deploy_action_human method, passing in kwargs for arguments
37 # this should throw an exception: pytan.utils.HandlerError
38 import traceback
39 try:
40     handler.deploy_action_human(**kwargs)
41 except Exception as e:
42     traceback.print_exc(file=sys.stdout)
```

### Output from Python Code

```
1  Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
2  2014-12-08 16:28:29,332 INFO      question_progress: Results 0% (Get Online = "True" from all machine
3  2014-12-08 16:28:34,353 INFO      question_progress: Results 50% (Get Online = "True" from all machine
4  2014-12-08 16:28:39,370 INFO      question_progress: Results 83% (Get Online = "True" from all machine
5  2014-12-08 16:28:44,391 INFO      question_progress: Results 100% (Get Online = "True" from all machi
6  Traceback (most recent call last):
7    File "<string>", line 40, in <module>
8    File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 1172, in deploy_action_human
9      **kwargs
10   File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 1026, in deploy_action
11     empty_ok=False,
12   File "/Users/jolsen/gh/pytan/lib/pytan/utils.py", line 2147, in build_param_objlist
13     raise HandlerError(err(obj_name, p_key, jsonify(obj_param)))
14 HandlerError: PackageSpec, name: 'Custom Tagging - Add Tags' parameter key '$1' requires a value, pa
15 {
16   "defaultValue": "",
```

```

17 "helpString": "Enter tags space-delimited.",
18 "key": "$1",
19 "label": "Add tags (space-delimited)",
20 "maxChars": 0,
21 "model": "com.tanium.components.parameters::TextInputParameter",
22 "parameterType": "com.tanium.components.parameters::TextInputParameter",
23 "promptText": "e.g. PCI DMZ Decomm",
24 "restrict": null,
25 "validationExpressions": [
26     {
27         "expression": "\\S",
28         "flags": "",
29         "helpString": "You must enter a value",
30         "model": "com.tanium.models::ValidationExpression",
31         "parameterType": "com.tanium.models::ValidationExpression"
32     }
33 ],
34 "value": ""
35 }

```

## pytan API Valid Create Object Examples

### Create user

Create a user called API Test User

### Example Python Code

```

1  # Path to lib directory which contains pytan package
2  PYTAN_LIB_PATH = '../lib'
3
4  # connection info for Tanium Server
5  USERNAME = "Tanium User"
6  PASSWORD = "T@n!um"
7  HOST = "172.16.31.128"
8  PORT = "444"
9
10 # Logging conrols
11 LOGLEVEL = 2
12 DEBUGFORMAT = False
13
14 import sys, tempfile
15 sys.path.append(PYTAN_LIB_PATH)
16
17 import pytan
18 handler = pytan.Handler(
19     username=USERNAME,
20     password=PASSWORD,
21     host=HOST,
22     port=PORT,
23     loglevel=LOGLEVEL,
24     debugformat=DEBUGFORMAT,
25 )
26
27 print handler

```

```
28
29 # setup the arguments for the delete method (to remove the package in case it exists)
30 delete_kwargs = {}
31 delete_kwargs["objtype"] = 'user'
32 delete_kwargs["name"] = 'API Test User'
33
34
35 # setup the arguments for the handler method
36 kwargs = {}
37 kwargs["username"] = u'API Test User'
38 kwargs["rolename"] = u'Administrator'
39 kwargs["properties"] = [[u'property1', u'value1']]
40
41 # delete the object in case it already exists
42 try:
43     handler.delete(**delete_kwargs)
44 except Exception as e:
45     print e
46
47 # call the handler with the create_user method, passing in kwargs for arguments
48 response = handler.create_user(**kwargs)
49
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 "print the object returned in JSON format:"
60 print response.to_json(response)
61
62 # delete the object, we are done with it now
63 try:
64     handler.delete(**delete_kwargs)
65 except Exception as e:
66     print e
```

### Output from Python Code

```
1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
2 No results found searching for user with {'name': 'API Test User'}!!
3 2014-12-08 16:28:44,451 INFO handler: New user 'API Test User' created with ID 16, roles: ['Admi
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": 16,
```

```

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 c
39             "id": 1,
40             "name": "Administrator",
41             "permissions": {
42                 "_type": "permissions",
43                 "permission": "admin"
44             }
45         }
46     ]
47 }
48 }
49 2014-12-08 16:28:44,469 INFO      handler: Deleted "User, name: 'API Test User'"

```

## Create package

Create a package called package49

## Example Python Code

```

1  # Path to lib directory which contains pytan package
2  PYTAN_LIB_PATH = '../lib'
3
4  # connection info for Tanium Server
5  USERNAME = "Tanium User"
6  PASSWORD = "T@n!um"
7  HOST = "172.16.31.128"
8  PORT = "444"
9
10 # Logging conrols
11 LOGLEVEL = 2
12 DEBUGFORMAT = False
13
14 import sys, tempfile

```

```
15 sys.path.append(PYTAN_LIB_PATH)
16
17 import pytan
18 handler = pytan.Handler(
19     username=USERNAME,
20     password=PASSWORD,
21     host=HOST,
22     port=PORT,
23     loglevel=LOGLEVEL,
24     debugformat=DEBUGFORMAT,
25 )
26
27 print handler
28
29 # setup the arguments for the delete method (to remove the package in case it exists)
30 delete_kwargs = {}
31 delete_kwargs["objtype"] = 'package'
32 delete_kwargs["name"] = 'package49'
33
34
35 # setup the arguments for the handler method
36 kwargs = {}
37 kwargs["expire_seconds"] = 1500
38 kwargs["display_name"] = u'package49 API test'
39 kwargs["name"] = u'package49'
40 kwargs["parameters_json_file"] = u'../doc/example_of_all_package_parameters.json'
41 kwargs["verify_expire_seconds"] = 3600
42 kwargs["command"] = u'package49 $1 $2 $3 $4 $5 $6 $7 $8'
43 kwargs["file_urls"] = [u'3600::testing.vbs|https://content.tanium.com/files/initialcontent/bundles/
44 kwargs["verify_filter_options"] = [u'and']
45 kwargs["verify_filters"] = [u'Custom Tags, that contains:tag']
46 kwargs["command_timeout_seconds"] = 9999
47
48 # delete the object in case it already exists
49 try:
50     handler.delete(**delete_kwargs)
51 except Exception as e:
52     print e
53
54 # call the handler with the create_package method, passing in kwargs for arguments
55 response = handler.create_package(**kwargs)
56
57
58 print ""
59 print "Type of response: ", type(response)
60
61 print ""
62 print "print of response:"
63 print response
64
65 print ""
66 print "print the object returned in JSON format:"
67 print response.to_json(response)
68
69 # delete the object, we are done with it now
70 try:
71     handler.delete(**delete_kwargs)
72 except Exception as e:
```

```
73 print e
```

## Output from Python Code

```

1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
2 No results found searching for PackageSpec, name: 'package49'!!
3 2014-12-08 16:28:44,534 INFO handler: New package 'package49' created with ID 107, command: 'pac
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": "2014-12-08T21:28:44",
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": 235,
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": 107,
39     "last_modified_by": "Tanium User",
40     "last_update": "2014-12-08T21:28:44",
41     "modification_time": "2014-12-08T21:28:44",
42     "name": "package49",
43     "parameter_definition": "{\"parameterType\": \"com.tanium.components.parameters::ParametersArray\"
44     "source_id": 0,
45     "verify_group_id": 396
46 }
47 2014-12-08 16:28:44,554 INFO handler: Deleted 'PackageSpec, id: 107'

```

## Create group

Create a group called All Windows Computers API Test

## Example Python Code

```
1  # Path to lib directory which contains pytan package
2  PYTAN_LIB_PATH = '../lib'
3
4  # connection info for Tanium Server
5  USERNAME = "Tanium User"
6  PASSWORD = "T@n!um"
7  HOST = "172.16.31.128"
8  PORT = "444"
9
10 # Logging conrols
11 LOGLEVEL = 2
12 DEBUGFORMAT = False
13
14 import sys, tempfile
15 sys.path.append(PYTAN_LIB_PATH)
16
17 import pytan
18 handler = pytan.Handler(
19     username=USERNAME,
20     password=PASSWORD,
21     host=HOST,
22     port=PORT,
23     loglevel=LOGLEVEL,
24     debugformat=DEBUGFORMAT,
25 )
26
27 print handler
28
29 # setup the arguments for the delete method (to remove the package in case it exists)
30 delete_kwargs = {}
31 delete_kwargs["objtype"] = 'group'
32 delete_kwargs["name"] = 'All Windows Computers API Test'
33
34
35 # setup the arguments for the handler method
36 kwargs = {}
37 kwargs["groupname"] = u'All Windows Computers API Test'
38 kwargs["filters"] = [u'Operating System, that contains:Windows']
39 kwargs["filter_options"] = [u'and']
40
41 # delete the object in case it already exists
42 try:
43     handler.delete(**delete_kwargs)
44 except Exception as e:
45     print e
46
47 # call the handler with the create_group method, passing in kwargs for arguments
48 response = handler.create_group(**kwargs)
49
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 "print the object returned in JSON format:"
60 print response.to_json(response)
61
62 # delete the object, we are done with it now
63 try:
64     handler.delete(**delete_kwargs)
65 except Exception as e:
66     print e

```

## Output from Python Code

```

1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
2 No results found searching for Group, name: 'All Windows Computers API Test'!!
3 2014-12-08 16:28:44,605 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": 1,
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": 397,
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 2014-12-08 16:28:44,619 INFO      handler: Deleted 'Group, id: 397'
```

## Create whitelisted url

Create a whitelisted url

## Example Python Code

```
1  # Path to lib directory which contains pytan package
2  PYTAN_LIB_PATH = '../lib'
3
4  # connection info for Tanium Server
5  USERNAME = "Tanium User"
6  PASSWORD = "T@n!um"
7  HOST = "172.16.31.128"
8  PORT = "444"
9
10 # Logging conrols
11 LOGLEVEL = 2
12 DEBUGFORMAT = False
13
14 import sys, tempfile
15 sys.path.append(PYTAN_LIB_PATH)
16
17 import pytan
18 handler = pytan.Handler(
19     username=USERNAME,
20     password=PASSWORD,
21     host=HOST,
22     port=PORT,
23     loglevel=LOGLEVEL,
24     debugformat=DEBUGFORMAT,
25 )
26
27 print handler
28
29 # setup the arguments for the delete method (to remove the package in case it exists)
30 delete_kwargs = {}
31 delete_kwargs["objtype"] = 'whitelisted_url'
32 delete_kwargs["url_regex"] = 'regex:http://test.com/*.API_Test.*URL'
33
34
35 # setup the arguments for the handler method
36 kwargs = {}
37 kwargs["url"] = u'http://test.com/*.API_Test.*URL'
38 kwargs["regex"] = True
39 kwargs["properties"] = [[u'property1', u'value1']]
40 kwargs["download_seconds"] = 3600
41
42 # delete the object in case it already exists
43 try:
```

```

44     handler.delete(**delete_kwargs)
45 except Exception as e:
46     print e
47
48 # call the handler with the create_whitelisted_url method, passing in kwargs for arguments
49 response = handler.create_whitelisted_url(**kwargs)
50
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 "print the object returned in JSON format:"
61 print response.to_json(response)
62
63 # delete the object, we are done with it now
64 try:
65     handler.delete(**delete_kwargs)
66 except Exception as e:
67     print e

```

### Output from Python Code

```

1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
2 No results found searching for whitelisted_url with {'url_regex': 'regex:http://test.com/.API_Test.'}
3 2014-12-08 16:28:44,663 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: 52
9
10 print the object returned in JSON format:
11 {
12     "_type": "white_listed_url",
13     "download_seconds": 3600,
14     "id": 52,
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 2014-12-08 16:28:44,685 INFO handler: Deleted 'WhiteListedUrl, id: 52'

```

## pytan API Invalid Create Object Examples

### Invalid create sensor

Create a sensor (Unsupported!)

### Example Python Code

```
1  # Path to lib directory which contains pytan package
2  PYTAN_LIB_PATH = '../lib'
3
4  # connection info for Tanium Server
5  USERNAME = "Tanium User"
6  PASSWORD = "T@n!um"
7  HOST = "172.16.31.128"
8  PORT = "444"
9
10 # Logging conrols
11 LOGLEVEL = 2
12 DEBUGFORMAT = False
13
14 import sys, tempfile
15 sys.path.append(PYTAN_LIB_PATH)
16
17 import pytan
18 handler = pytan.Handler(
19     username=USERNAME,
20     password=PASSWORD,
21     host=HOST,
22     port=PORT,
23     loglevel=LOGLEVEL,
24     debugformat=DEBUGFORMAT,
25 )
26
27 print handler
28
29 # setup the arguments for the handler method
30 kwargs = {}
31
32
33 # call the handler with the create_sensor method, passing in kwargs for arguments
34 # this should throw an exception: pytan.utils.HandlerError
35 import traceback
36 try:
37     handler.create_sensor(**kwargs)
38 except Exception as e:
39     traceback.print_exc(file=sys.stdout)
```

### Output from Python Code

```
1  Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
2  Traceback (most recent call last):
3      File "<string>", line 37, 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  # Path to lib directory which contains pytan package
2  PYTAN_LIB_PATH = '../lib'
3
4  # connection info for Tanium Server
5  USERNAME = "Tanium User"
6  PASSWORD = "T@n!um"
7  HOST = "172.16.31.128"
8  PORT = "444"
9
10 # Logging conrols
11 LOGLEVEL = 2
12 DEBUGFORMAT = False
13
14 import sys, tempfile
15 sys.path.append(PYTAN_LIB_PATH)
16
17 import pytan
18 handler = pytan.Handler(
19     username=USERNAME,
20     password=PASSWORD,
21     host=HOST,
22     port=PORT,
23     loglevel=LOGLEVEL,
24     debugformat=DEBUGFORMAT,
25 )
26
27 print handler
28
29 # set the attribute name and value we want to add to the original object (if any)
30 attr_name = "name"
31 attr_add = " API TEST"
32
33 # delete object before creating it?
34 delete = True
35
36 # setup the arguments for getting an object to export as json file
37 get_kwargs = {}
38 get_kwargs["objtype"] = u'package'
39 get_kwargs["id"] = 31
40
41
42 # get objects to use as an export to JSON file

```

```
43 orig_objs = handler.get(**get_kwargs)
44
45 # if attr_name and attr_add exists, modify the orig_objs to add attr_add to the attribute
46 # attr_name
47 if attr_name:
48     for x in orig_objs:
49         new_attr = getattr(x, attr_name)
50         new_attr += attr_add
51         setattr(x, attr_name, new_attr)
52     if delete:
53         # delete the object in case it already exists
54         del_kwargs = {}
55         del_kwargs[attr_name] = new_attr
56         del_kwargs['objtype'] = u'package'
57         try:
58             handler.delete(**del_kwargs)
59         except Exception as e:
60             print e
61
62 # export orig_objs to a json file
63 json_file, results = handler.export_to_report_file(
64     obj=orig_objs,
65     export_format='json',
66     report_dir=tempfile.gettempdir(),
67 )
68
69 # create the object from the exported JSON file
70 create_kwargs = {'objtype': u'package', 'json_file': json_file}
71 response = handler.create_from_json(**create_kwargs)
72
73
74 print ""
75 print "Type of response: ", type(response)
76
77 print ""
78 print "print of response:"
79 print response
80
81 print ""
82 print "print the object returned in JSON format:"
83 print response.to_json(response)
```

### Output from Python Code

```
1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
2 2014-12-08 16:28:44,729 INFO handler: Deleted 'PackageSpec, id: 101'
3 2014-12-08 16:28:44,730 INFO handler: Report file '/var/folders/dk/vjr1r_c53yx6k6gzzp2bbt_c40000g
4 2014-12-08 16:28:44,761 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": "2014-12-08T21:28:44",
21         "deleted_flag": 0,
22         "display_name": "Custom Tagging - Add Tags",
23         "expire_seconds": 660,
24         "files": {
25             "_type": "package_files",
26             "file": [
27                 {
28                     "_type": "file",
29                     "bytes_downloaded": 1972,
30                     "bytes_total": 1972,
31                     "cache_status": "CACHED",
32                     "download_seconds": 0,
33                     "download_start_time": "2014-12-08T19:23:55",
34                     "hash": "55aa6c54d82282ad2d41390e49f7b9939c582e14fa5cfca1b7b7fb9264261182",
35                     "id": 71,
36                     "last_download_progress_time": "2014-12-08T19:24:06",
37                     "name": "add-tags.vbs",
38                     "size": 1972,
39                     "source": "https://content.tanium.com/files/initialcontent/bundles/2014-11-05_12-56-07-8",
40                     "status": 200
41                 }
42             ]
43         },
44         "hidden_flag": 0,
45         "id": 108,
46         "last_modified_by": "Tanium User",
47         "last_update": "2014-12-08T21:28:44",
48         "metadata": {
49             "_type": "metadata",
50             "item": [
51                 {
52                     "_type": "item",
53                     "admin_flag": 0,
54                     "name": "defined",
55                     "value": "Tanium"
56                 },
57                 {
58                     "_type": "item",
59                     "admin_flag": 0,
60                     "name": "category",
61                     "value": "Tanium"
62                 }
63             ]
64         },
65         "modification_time": "2014-12-08T21:28:44",
66         "name": "Custom Tagging - Add Tags API TEST",
67         "parameter_definition": "{\"parameters\": [{\"restrict\": null, \"validationExpressions\": [{\"hel",
68         "source_id": 0,
69         "verify_group_id": 0
70     }
71 ]

```

72     }

### 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  # Path to lib directory which contains pytan package
2  PYTAN_LIB_PATH = '../lib'
3
4  # connection info for Tanium Server
5  USERNAME = "Tanium User"
6  PASSWORD = "T@n!um"
7  HOST = "172.16.31.128"
8  PORT = "444"
9
10 # Logging conrols
11 LOGLEVEL = 2
12 DEBUGFORMAT = False
13
14 import sys, tempfile
15 sys.path.append(PYTAN_LIB_PATH)
16
17 import pytan
18 handler = pytan.Handler(
19     username=USERNAME,
20     password=PASSWORD,
21     host=HOST,
22     port=PORT,
23     loglevel=LOGLEVEL,
24     debugformat=DEBUGFORMAT,
25 )
26
27 print handler
28
29 # set the attribute name and value we want to add to the original object (if any)
30 attr_name = "name"
31 attr_add = " API TEST"
32
33 # delete object before creating it?
34 delete = True
35
36 # setup the arguments for getting an object to export as json file
37 get_kwargs = {}
38 get_kwargs["objtype"] = u'user'
39 get_kwargs["id"] = 1
40
41
42 # get objects to use as an export to JSON file
43 orig_objs = handler.get(**get_kwargs)
44
45 # if attr_name and attr_add exists, modify the orig_objs to add attr_add to the attribute
46 # attr_name
```



```

47 if attr_name:
48     for x in orig_objs:
49         new_attr = getattr(x, attr_name)
50         new_attr += attr_add
51         setattr(x, attr_name, new_attr)
52     if delete:
53         # delete the object in case it already exists
54         del_kwargs = {}
55         del_kwargs[attr_name] = new_attr
56         del_kwargs['objtype'] = u'user'
57         try:
58             handler.delete(**del_kwargs)
59         except Exception as e:
60             print e
61
62 # export orig_objs to a json file
63 json_file, results = handler.export_to_report_file(
64     obj=orig_objs,
65     export_format='json',
66     report_dir=tempfile.gettempdir(),
67 )
68
69 # create the object from the exported JSON file
70 create_kwargs = {'objtype': u'user', 'json_file': json_file}
71 response = handler.create_from_json(**create_kwargs)
72
73
74 print ""
75 print "Type of response: ", type(response)
76
77 print ""
78 print "print of response:"
79 print response
80
81 print ""
82 print "print the object returned in JSON format:"
83 print response.to_json(response)

```

### Output from Python Code

```

1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
2 2014-12-08 16:28:44,799 INFO handler: Deleted "User, name: 'Jim Olsen API TEST'"
3 2014-12-08 16:28:44,800 INFO handler: Report file '/var/folders/dk/vjrlr_c53yx6k6gzzp2bbt_c40000g
4 2014-12-08 16:28:44,816 INFO handler: New User, name: 'Jim Olsen API TEST' (ID: 17) created succ
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": 17,
20     "last_login": "2001-01-01T00:00:00",
21     "name": "Jim Olsen API TEST",
22     "permissions": {
23         "_type": "permissions",
24         "permission": "admin"
25     },
26     "roles": {
27         "_type": "roles",
28         "role": [
29             {
30                 "_type": "role",
31                 "description": "Administrators can perform all functions in the system, including creati
32                 "id": 1,
33                 "name": "Administrator",
34                 "permissions": {
35                     "_type": "permissions",
36                     "permission": "admin"
37                 }
38             }
39         ]
40     }
41 }
42 ]
43 }
```

### 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  # Path to lib directory which contains pytan package
2  PYTAN_LIB_PATH = '../lib'
3
4  # connection info for Tanium Server
5  USERNAME = "Tanium User"
6  PASSWORD = "T@n!um"
7  HOST = "172.16.31.128"
8  PORT = "444"
9
10 # Logging conrols
11 LOGLEVEL = 2
12 DEBUGFORMAT = False
13
14 import sys, tempfile
15 sys.path.append(PYTAN_LIB_PATH)
16
17 import pytan
18 handler = pytan.Handler(
19     username=USERNAME,
20     password=PASSWORD,
```

```

21     host=HOST,
22     port=PORT,
23     loglevel=LOGLEVEL,
24     debugformat=DEBUGFORMAT,
25 )
26
27 print handler
28
29 # set the attribute name and value we want to add to the original object (if any)
30 attr_name = "name"
31 attr_add = " API TEST"
32
33 # delete object before creating it?
34 delete = True
35
36 # setup the arguments for getting an object to export as json file
37 get_kwargs = {}
38 get_kwargs["objtype"] = u'saved_question'
39 get_kwargs["id"] = 1
40
41
42 # get objects to use as an export to JSON file
43 orig_objs = handler.get(**get_kwargs)
44
45 # if attr_name and attr_add exists, modify the orig_objs to add attr_add to the attribute
46 # attr_name
47 if attr_name:
48     for x in orig_objs:
49         new_attr = getattr(x, attr_name)
50         new_attr += attr_add
51         setattr(x, attr_name, new_attr)
52         if delete:
53             # delete the object in case it already exists
54             del_kwargs = {}
55             del_kwargs[attr_name] = new_attr
56             del_kwargs['objtype'] = u'saved_question'
57             try:
58                 handler.delete(**del_kwargs)
59             except Exception as e:
60                 print e
61
62 # export orig_objs to a json file
63 json_file, results = handler.export_to_report_file(
64     obj=orig_objs,
65     export_format='json',
66     report_dir=tempfile.gettempdir(),
67 )
68
69 # create the object from the exported JSON file
70 create_kwargs = {'objtype': u'saved_question', 'json_file': json_file}
71 response = handler.create_from_json(**create_kwargs)
72
73
74 print ""
75 print "Type of response: ", type(response)
76
77 print ""
78 print "print of response:"

```

```
79 print response
80
81 print ""
82 print "print the object returned in JSON format:"
83 print response.to_json(response)
```

### Output from Python Code

```
1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
2 2014-12-08 16:28:44,876 INFO handler: Deleted 'SavedQuestion, id: 178'
3 2014-12-08 16:28:44,877 INFO handler: Report file '/var/folders/dk/vjrlr_c53yx6k6gzzp2bbt_c40000g
4 2014-12-08 16:28:44,909 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": 180,
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": 315,
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": "2014-12-08T20:32:42",
41                 "expire_seconds": 0,
42                 "force_computer_id_flag": 0,
43                 "hidden_flag": 0,
44                 "id": 315,
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": 1
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": "2014-12-08T19:20:40",
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": "2014-12-08T19:20:40",
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\\necho 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  # Path to lib directory which contains pytan package
2  PYTAN_LIB_PATH = '../lib'
3
4  # connection info for Tanium Server
5  USERNAME = "Tanium User"
6  PASSWORD = "T@n!um"
```

```

7  HOST = "172.16.31.128"
8  PORT = "444"
9
10 # Logging conrols
11 LOGLEVEL = 2
12 DEBUGFORMAT = False
13
14 import sys, tempfile
15 sys.path.append(PYTAN_LIB_PATH)
16
17 import pytan
18 handler = pytan.Handler(
19     username=USERNAME,
20     password=PASSWORD,
21     host=HOST,
22     port=PORT,
23     loglevel=LOGLEVEL,
24     debugformat=DEBUGFORMAT,
25 )
26
27 print handler
28
29 # set the attribute name and value we want to add to the original object (if any)
30 attr_name = ""
31 attr_add = ""
32
33 # delete object before creating it?
34 delete = False
35
36 # setup the arguments for getting an object to export as json file
37 get_kwargs = {}
38 get_kwargs["objtype"] = u'action'
39 get_kwargs["id"] = 1
40
41
42 # get objects to use as an export to JSON file
43 orig_objs = handler.get(**get_kwargs)
44
45 # if attr_name and attr_add exists, modify the orig_objs to add attr_add to the attribute
46 # attr_name
47 if attr_name:
48     for x in orig_objs:
49         new_attr = getattr(x, attr_name)
50         new_attr += attr_add
51         setattr(x, attr_name, new_attr)
52         if delete:
53             # delete the object in case it already exists
54             del_kwargs = {}
55             del_kwargs[attr_name] = new_attr
56             del_kwargs['objtype'] = u'action'
57             try:
58                 handler.delete(**del_kwargs)
59             except Exception as e:
60                 print e
61
62 # export orig_objs to a json file
63 json_file, results = handler.export_to_report_file(
64     obj=orig_objs,

```

```
65     export_format='json',
66     report_dir=tempfile.gettempdir(),
67 )
68
69 # create the object from the exported JSON file
70 create_kwargs = {'objtype': u'action', 'json_file': json_file}
71 response = handler.create_from_json(**create_kwargs)
72
73
74 print ""
75 print "Type of response: ", type(response)
76
77 print ""
78 print "print of response:"
79 print response
80
81 print ""
82 print "print the object returned in JSON format:"
83 print response.to_json(response)
```

### Output from Python Code

```
1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
2 2014-12-08 16:28:44,933 INFO      handler: Report file '/var/folders/dk/vjrlr_c53yx6k6gzp2bbt_c40000g
3 2014-12-08 16:28:45,017 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": "2014-12-08T21:28:45",
23             "distribute_seconds": 600,
24             "expiration_time": "2014-12-08T22:18:45",
25             "expire_seconds": 3000,
26             "history_saved_question": {
27                 "_type": "saved_question",
28                 "id": 173
29             },
30             "id": 50,
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": 43
41     },
42     "skip_lock_flag": 0,
43     "start_time": "2014-12-08T21:28:45",
44     "status": "Active",
45     "stopped_flag": 0,
46     "target_group": {
47         "_type": "group",
48         "id": 65,
49         "name": "Default"
50     },
51     "user": {
52         "_type": "user",
53         "group_id": 0,
54         "id": 2,
55         "last_login": "2014-12-08T21:28:45",
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  # Path to lib directory which contains pytan package
2  PYTAN_LIB_PATH = '../lib'
3
4  # connection info for Tanium Server
5  USERNAME = "Tanium User"
6  PASSWORD = "T@n!um"
7  HOST = "172.16.31.128"
8  PORT = "444"
9
10 # Logging conrols
11 LOGLEVEL = 2
12 DEBUGFORMAT = False
13
14 import sys, tempfile
15 sys.path.append(PYTAN_LIB_PATH)
16
17 import pytan
18 handler = pytan.Handler(
19     username=USERNAME,
20     password=PASSWORD,
21     host=HOST,

```

```
22     port=PORT,
23     loglevel=LOGLEVEL,
24     debugformat=DEBUGFORMAT,
25 )
26
27 print handler
28
29 # set the attribute name and value we want to add to the original object (if any)
30 attr_name = "name"
31 attr_add = " API TEST"
32
33 # delete object before creating it?
34 delete = True
35
36 # setup the arguments for getting an object to export as json file
37 get_kwargs = {}
38 get_kwargs["objtype"] = u'sensor'
39 get_kwargs["id"] = 381
40
41
42 # get objects to use as an export to JSON file
43 orig_objs = handler.get(**get_kwargs)
44
45 # if attr_name and attr_add exists, modify the orig_objs to add attr_add to the attribute
46 # attr_name
47 if attr_name:
48     for x in orig_objs:
49         new_attr = getattr(x, attr_name)
50         new_attr += attr_add
51         setattr(x, attr_name, new_attr)
52         if delete:
53             # delete the object in case it already exists
54             del_kwargs = {}
55             del_kwargs[attr_name] = new_attr
56             del_kwargs['objtype'] = u'sensor'
57             try:
58                 handler.delete(**del_kwargs)
59             except Exception as e:
60                 print e
61
62 # export orig_objs to a json file
63 json_file, results = handler.export_to_report_file(
64     obj=orig_objs,
65     export_format='json',
66     report_dir=tempfile.gettempdir(),
67 )
68
69 # create the object from the exported JSON file
70 create_kwargs = {'objtype': u'sensor', 'json_file': json_file}
71 response = handler.create_from_json(**create_kwargs)
72
73
74 print ""
75 print "Type of response: ", type(response)
76
77 print ""
78 print "print of response:"
79 print response
```

```

80
81 print ""
82 print "print the object returned in JSON format:"
83 print response.to_json(response)

```

### Output from Python Code

```

1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
2 2014-12-08 16:28:45,059 INFO handler: Deleted 'Sensor, id: 827'
3 2014-12-08 16:28:45,060 INFO handler: Report file '/var/folders/dk/vjrlr_c53yx6k6gzzp2bbt_c40000g
4 2014-12-08 16:28:45,102 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": "2014-12-08T21:28:45",
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": 830,
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": "2014-12-08T21:28:45",
40             "name": "Folder Name Search with RegEx Match API TEST",
41             "parameter_definition": "{ \"parameters\": [{ \"restrict\": null, \"validationExpressions\": [{ \"hel
42             "queries": {
43                 "_type": "queries",
44                 "query": [
45                     {
46                         "_type": "query",
47                         "platform": "Windows",
48                         "script": "&#039;=====\n&#039; Folder Name Se
49                         "script_type": "VBScript"
50                     },

```

```
51         {
52             "_type": "query",
53             "platform": "Linux",
54             "script": "#!/bin/bash\n#||dirname|||regex|||casesensitive|||global||\necho Windows
55             "script_type": "UnixShell"
56         },
57         {
58             "_type": "query",
59             "platform": "Mac",
60             "script": "#!/bin/bash\n#||dirname|||regex|||casesensitive|||global||\necho 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  # Path to lib directory which contains pytan package
2  PYTAN_LIB_PATH = '../lib'
3
4  # connection info for Tanium Server
5  USERNAME = "Tanium User"
6  PASSWORD = "T@n!um"
7  HOST = "172.16.31.128"
8  PORT = "444"
9
10 # Logging conrols
11 LOGLEVEL = 2
12 DEBUGFORMAT = False
13
14 import sys, tempfile
15 sys.path.append(PYTAN_LIB_PATH)
16
17 import pytan
18 handler = pytan.Handler(
19     username=USERNAME,
20     password=PASSWORD,
21     host=HOST,
22     port=PORT,
23     loglevel=LOGLEVEL,
24     debugformat=DEBUGFORMAT,
25 )
26
27 print handler
```

```

28
29 # set the attribute name and value we want to add to the original object (if any)
30 attr_name = ""
31 attr_add = ""
32
33 # delete object before creating it?
34 delete = False
35
36 # setup the arguments for getting an object to export as json file
37 get_kwargs = {}
38 get_kwargs["objtype"] = u'question'
39 get_kwargs["id"] = 1
40
41
42 # get objects to use as an export to JSON file
43 orig_objs = handler.get(**get_kwargs)
44
45 # if attr_name and attr_add exists, modify the orig_objs to add attr_add to the attribute
46 # attr_name
47 if attr_name:
48     for x in orig_objs:
49         new_attr = getattr(x, attr_name)
50         new_attr += attr_add
51         setattr(x, attr_name, new_attr)
52         if delete:
53             # delete the object in case it already exists
54             del_kwargs = {}
55             del_kwargs[attr_name] = new_attr
56             del_kwargs['objtype'] = u'question'
57             try:
58                 handler.delete(**del_kwargs)
59             except Exception as e:
60                 print e
61
62 # export orig_objs to a json file
63 json_file, results = handler.export_to_report_file(
64     obj=orig_objs,
65     export_format='json',
66     report_dir=tempfile.gettempdir(),
67 )
68
69 # create the object from the exported JSON file
70 create_kwargs = {'objtype': u'question', 'json_file': json_file}
71 response = handler.create_from_json(**create_kwargs)
72
73
74 print ""
75 print "Type of response: ", type(response)
76
77 print ""
78 print "print of response:"
79 print response
80
81 print ""
82 print "print the object returned in JSON format:"
83 print response.to_json(response)

```

## Output from Python Code

```
1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
2 2014-12-08 16:28:45,123 INFO handler: Report file '/var/folders/dk/vjr1r_c53yx6k6gzzp2bbt_c40000g
3 2014-12-08 16:28:45,159 INFO handler: New Question, id: 437 (ID: 437) 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": "2014-12-08T21:38:45",
22             "expire_seconds": 0,
23             "force_computer_id_flag": 1,
24             "hidden_flag": 0,
25             "id": 437,
26             "management_rights_group": {
27                 "_type": "group",
28                 "id": 0
29             },
30             "query_text": "Get Action Statuses matches \"Nil\" from all machines",
31             "saved_question": {
32                 "_type": "saved_question",
33                 "id": 0
34             },
35             "selects": {
36                 "_type": "selects",
37                 "select": [
38                     {
39                         "_type": "select",
40                         "filter": {
41                             "_type": "filter",
42                             "all_times_flag": 0,
43                             "all_values_flag": 1,
44                             "delimiter_index": 0,
45                             "end_time": "2001-01-01T00:00:00",
46                             "ignore_case_flag": 1,
47                             "max_age_seconds": 0,
48                             "not_flag": 0,
49                             "operator": "RegexMatch",
50                             "start_time": "2001-01-01T00:00:00",
51                             "substring_flag": 0,
52                             "substring_length": 0,
53                             "substring_start": 0,
54                             "utf8_flag": 0,
55                             "value": "Nil",
56                             "value_type": "String"
57                         },
```

```

58         "sensor": {
59             "_type": "sensor",
60             "category": "Reserved",
61             "description": "The recorded state of each action a client has taken recently in the f
62             "exclude_from_parse_flag": 1,
63             "hash": 1792443391,
64             "hidden_flag": 0,
65             "id": 1,
66             "ignore_case_flag": 1,
67             "max_age_seconds": 3600,
68             "name": "Action Statuses",
69             "queries": {
70                 "_type": "queries",
71                 "query": [
72                     {
73                         "_type": "query",
74                         "platform": "Windows",
75                         "script": "Reserved",
76                         "script_type": "WMIQuery"
77                     }
78                 ]
79             },
80             "source_id": 0,
81             "string_count": 3540,
82             "value_type": "String"
83         }
84     }
85 ]
86 },
87 "skip_lock_flag": 0,
88 "user": {
89     "_type": "user",
90     "id": 2,
91     "name": "Tanium User"
92 }
93 ]
94 ]
95 }

```

### 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  # Path to lib directory which contains pytan package
2  PYTAN_LIB_PATH = '../lib'
3
4  # connection info for Tanium Server
5  USERNAME = "Tanium User"
6  PASSWORD = "T@n!um"
7  HOST = "172.16.31.128"
8  PORT = "444"

```

```
9
10 # Logging controls
11 LOGLEVEL = 2
12 DEBUGFORMAT = False
13
14 import sys, tempfile
15 sys.path.append(PYTAN_LIB_PATH)
16
17 import pytan
18 handler = pytan.Handler(
19     username=USERNAME,
20     password=PASSWORD,
21     host=HOST,
22     port=PORT,
23     loglevel=LOGLEVEL,
24     debugformat=DEBUGFORMAT,
25 )
26
27 print handler
28
29 # set the attribute name and value we want to add to the original object (if any)
30 attr_name = "url_regex"
31 attr_add = " API TEST"
32
33 # delete object before creating it?
34 delete = True
35
36 # setup the arguments for getting an object to export as json file
37 get_kwargs = {}
38 get_kwargs["objtype"] = u'whitelisted_url'
39 get_kwargs["url_regex"] = u'test1'
40
41
42 # get objects to use as an export to JSON file
43 orig_objs = handler.get(**get_kwargs)
44
45 # if attr_name and attr_add exists, modify the orig_objs to add attr_add to the attribute
46 # attr_name
47 if attr_name:
48     for x in orig_objs:
49         new_attr = getattr(x, attr_name)
50         new_attr += attr_add
51         setattr(x, attr_name, new_attr)
52         if delete:
53             # delete the object in case it already exists
54             del_kwargs = {}
55             del_kwargs[attr_name] = new_attr
56             del_kwargs['objtype'] = u'whitelisted_url'
57             try:
58                 handler.delete(**del_kwargs)
59             except Exception as e:
60                 print e
61
62 # export orig_objs to a json file
63 json_file, results = handler.export_to_report_file(
64     obj=orig_objs,
65     export_format='json',
66     report_dir=tempfile.gettempdir(),
```



```

67 )
68
69 # create the object from the exported JSON file
70 create_kwargs = {'objtype': u'whitelisted_url', 'json_file': json_file}
71 response = handler.create_from_json(**create_kwargs)
72
73
74 print ""
75 print "Type of response: ", type(response)
76
77 print ""
78 print "print of response:"
79 print response
80
81 print ""
82 print "print the object returned in JSON format:"
83 print response.to_json(response)

```

### Output from Python Code

```

1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
2 2014-12-08 16:28:45,212 INFO      handler: Deleted 'WhiteListedUrl, id: 27'
3 2014-12-08 16:28:45,213 INFO      handler: Report file '/var/folders/dk/vjrlr_c53yx6k6gxp2bbt_c40000g
4 2014-12-08 16:28:45,226 INFO      handler: New WhiteListedUrl, id: 53 (ID: 53) created successfully!
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": 53,
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 # Path to lib directory which contains pytan package
2 PYTAN_LIB_PATH = '../lib'
3

```

```
4  # connection info for Tanium Server
5  USERNAME = "Tanium User"
6  PASSWORD = "T@n!um"
7  HOST = "172.16.31.128"
8  PORT = "444"
9
10 # Logging conrols
11 LOGLEVEL = 2
12 DEBUGFORMAT = False
13
14 import sys, tempfile
15 sys.path.append(PYTAN_LIB_PATH)
16
17 import pytan
18 handler = pytan.Handler(
19     username=USERNAME,
20     password=PASSWORD,
21     host=HOST,
22     port=PORT,
23     loglevel=LOGLEVEL,
24     debugformat=DEBUGFORMAT,
25 )
26
27 print handler
28
29 # set the attribute name and value we want to add to the original object (if any)
30 attr_name = "name"
31 attr_add = " API TEST"
32
33 # delete object before creating it?
34 delete = True
35
36 # setup the arguments for getting an object to export as json file
37 get_kwargs = {}
38 get_kwargs["objtype"] = u'group'
39 get_kwargs["name"] = u'All Computers'
40
41
42 # get objects to use as an export to JSON file
43 orig_objs = handler.get(**get_kwargs)
44
45 # if attr_name and attr_add exists, modify the orig_objs to add attr_add to the attribute
46 # attr_name
47 if attr_name:
48     for x in orig_objs:
49         new_attr = getattr(x, attr_name)
50         new_attr += attr_add
51         setattr(x, attr_name, new_attr)
52         if delete:
53             # delete the object in case it already exists
54             del_kwargs = {}
55             del_kwargs[attr_name] = new_attr
56             del_kwargs['objtype'] = u'group'
57             try:
58                 handler.delete(**del_kwargs)
59             except Exception as e:
60                 print e
61
```

```

62 # export orig_objs to a json file
63 json_file, results = handler.export_to_report_file(
64     obj=orig_objs,
65     export_format='json',
66     report_dir=tempfile.gettempdir(),
67 )
68
69 # create the object from the exported JSON file
70 create_kwargs = {'objtype': u'group', 'json_file': json_file}
71 response = handler.create_from_json(**create_kwargs)
72
73
74 print ""
75 print "Type of response: ", type(response)
76
77 print ""
78 print "print of response:"
79 print response
80
81 print ""
82 print "print the object returned in JSON format:"
83 print response.to_json(response)

```

### Output from Python Code

```

1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
2 2014-12-08 16:28:45,264 INFO handler: Deleted 'Group, id: 311'
3 2014-12-08 16:28:45,265 INFO handler: Report file '/var/folders/dk/vjrlr_c53yx6k6gzp2bbt_c40000g
4 2014-12-08 16:28:45,289 INFO handler: New Group, name: 'All Computers API TEST' (ID: 398) create
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": 1,
19             "filters": {
20                 "_type": "filters",
21                 "filter": []
22             },
23             "id": 398,
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  # Path to lib directory which contains pytan package
2  PYTAN_LIB_PATH = '../lib'
3
4  # connection info for Tanium Server
5  USERNAME = "Tanium User"
6  PASSWORD = "T@n!um"
7  HOST = "172.16.31.128"
8  PORT = "444"
9
10 # Logging conrols
11 LOGLEVEL = 2
12 DEBUGFORMAT = False
13
14 import sys, tempfile
15 sys.path.append(PYTAN_LIB_PATH)
16
17 import pytan
18 handler = pytan.Handler(
19     username=USERNAME,
20     password=PASSWORD,
21     host=HOST,
22     port=PORT,
23     loglevel=LOGLEVEL,
24     debugformat=DEBUGFORMAT,
25 )
26
27 print handler
28
29 # setup the arguments for getting an object to export as json file
30 get_kwargs = {}
31 get_kwargs["objtype"] = u'saved_action'
32 get_kwargs["name"] = u'Distribute Tanium Standard Utilities'
33
34 # get objects to use as an export to JSON file
35 orig_objs = handler.get(**get_kwargs)
36
37 # export orig_objs to a json file
38 json_file, results = handler.export_to_report_file(
39     obj=orig_objs,
40     export_format='json',
41     report_dir=tempfile.gettempdir(),
42 )
43
44 # call the handler with the create_from_json method, passing in kwargs for arguments
45 # this should throw an exception: pytan.utils.HandlerError
```

```

46 import traceback
47
48 # create the object from the exported JSON file
49 create_kwargs = {'objtype': u'saved_action', 'json_file': json_file}
50 try:
51     response = handler.create_from_json(**create_kwargs)
52 except Exception as e:
53     traceback.print_exc(file=sys.stdout)

```

## Output from Python Code

```

1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
2 2014-12-08 16:28:45,310 INFO handler: Report file '/var/folders/dk/vjrlr_c53yx6k6g2p2bbt_c40000g
3 Traceback (most recent call last):
4   File "<string>", line 51, 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 # Path to lib directory which contains pytan package
2 PYTAN_LIB_PATH = '../lib'
3
4 # connection info for Tanium Server
5 USERNAME = "Tanium User"
6 PASSWORD = "T@n!um"
7 HOST = "172.16.31.128"
8 PORT = "444"
9
10 # Logging conrols
11 LOGLEVEL = 2
12 DEBUGFORMAT = False
13
14 import sys, tempfile
15 sys.path.append(PYTAN_LIB_PATH)
16
17 import pytan
18 handler = pytan.Handler(
19     username=USERNAME,
20     password=PASSWORD,
21     host=HOST,
22     port=PORT,
23     loglevel=LOGLEVEL,
24     debugformat=DEBUGFORMAT,
25 )
26
27 print handler
28
29 # setup the arguments for getting an object to export as json file

```

```
30 get_kwargs = {}
31 get_kwargs["objtype"] = u'client'
32 get_kwargs["status"] = u'Leader'
33
34 # get objects to use as an export to JSON file
35 orig_objs = handler.get(**get_kwargs)
36
37 # export orig_objs to a json file
38 json_file, results = handler.export_to_report_file(
39     obj=orig_objs,
40     export_format='json',
41     report_dir=tempfile.gettempdir(),
42 )
43
44 # call the handler with the create_from_json method, passing in kwargs for arguments
45 # this should throw an exception: pytan.utils.HandlerError
46 import traceback
47
48 # create the object from the exported JSON file
49 create_kwargs = {'objtype': u'client', 'json_file': json_file}
50 try:
51     response = handler.create_from_json(**create_kwargs)
52 except Exception as e:
53     traceback.print_exc(file=sys.stdout)
```

### Output from Python Code

```
1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
2 2014-12-08 16:28:45,334 INFO      handler: Report file '/var/folders/dk/vjrlr_c53yx6k6g2p2bbt_c40000g
3 Traceback (most recent call last):
4   File "<string>", line 51, 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 # Path to lib directory which contains pytan package
2 PYTAN_LIB_PATH = '../lib'
3
4 # connection info for Tanium Server
5 USERNAME = "Tanium User"
6 PASSWORD = "T@n!um"
7 HOST = "172.16.31.128"
8 PORT = "444"
9
10 # Logging conrols
11 LOGLEVEL = 2
12 DEBUGFORMAT = False
13
```

```

14 import sys, tempfile
15 sys.path.append(PYTAN_LIB_PATH)
16
17 import pytan
18 handler = pytan.Handler(
19     username=USERNAME,
20     password=PASSWORD,
21     host=HOST,
22     port=PORT,
23     loglevel=LOGLEVEL,
24     debugformat=DEBUGFORMAT,
25 )
26
27 print handler
28
29 # setup the arguments for getting an object to export as json file
30 get_kwargs = {}
31 get_kwargs["objtype"] = u'userrole'
32 get_kwargs["name"] = u'Administrator'
33
34 # get objects to use as an export to JSON file
35 orig_objs = handler.get(**get_kwargs)
36
37 # export orig_objs to a json file
38 json_file, results = handler.export_to_report_file(
39     obj=orig_objs,
40     export_format='json',
41     report_dir=tempfile.gettempdir(),
42 )
43
44 # call the handler with the create_from_json method, passing in kwargs for arguments
45 # this should throw an exception: pytan.utils.HandlerError
46 import traceback
47
48 # create the object from the exported JSON file
49 create_kwargs = {'objtype': u'userrole', 'json_file': json_file}
50 try:
51     response = handler.create_from_json(**create_kwargs)
52 except Exception as e:
53     traceback.print_exc(file=sys.stdout)

```

### Output from Python Code

```

1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
2 2014-12-08 16:28:45,367 INFO handler: Report file '/var/folders/dk/vjrlr_c53yx6k6g2b2bbt_c40000g
3 Traceback (most recent call last):
4   File "<string>", line 51, 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  # Path to lib directory which contains pytan package
2  PYTAN_LIB_PATH = '../lib'
3
4  # connection info for Tanium Server
5  USERNAME = "Tanium User"
6  PASSWORD = "T@n!um"
7  HOST = "172.16.31.128"
8  PORT = "444"
9
10 # Logging conrols
11 LOGLEVEL = 2
12 DEBUGFORMAT = False
13
14 import sys, tempfile
15 sys.path.append(PYTAN_LIB_PATH)
16
17 import pytan
18 handler = pytan.Handler(
19     username=USERNAME,
20     password=PASSWORD,
21     host=HOST,
22     port=PORT,
23     loglevel=LOGLEVEL,
24     debugformat=DEBUGFORMAT,
25 )
26
27 print handler
28
29 # setup the arguments for getting an object to export as json file
30 get_kwargs = {}
31 get_kwargs["objtype"] = u'setting'
32 get_kwargs["id"] = 1
33
34 # get objects to use as an export to JSON file
35 orig_objs = handler.get(**get_kwargs)
36
37 # export orig_objs to a json file
38 json_file, results = handler.export_to_report_file(
39     obj=orig_objs,
40     export_format='json',
41     report_dir=tempfile.gettempdir(),
42 )
43
44 # call the handler with the create_from_json method, passing in kwargs for arguments
45 # this should throw an exception: pytan.utils.HandlerError
46 import traceback
47
48 # create the object from the exported JSON file
49 create_kwargs = {'objtype': u'setting', 'json_file': json_file}
50 try:
51     response = handler.create_from_json(**create_kwargs)
52 except Exception as e:
53     traceback.print_exc(file=sys.stdout)
```

## Output from Python Code



```

1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
2 2014-12-08 16:28:45,385 INFO handler: Report file '/var/folders/dk/vjrlr_c53yx6k6gzp2bbt_c40000g
3 Traceback (most recent call last):
4   File "<string>", line 51, 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  # Path to lib directory which contains pytan package
2  PYTAN_LIB_PATH = '../lib'
3
4  # connection info for Tanium Server
5  USERNAME = "Tanium User"
6  PASSWORD = "T@n!um"
7  HOST = "172.16.31.128"
8  PORT = "444"
9
10 # Logging conrols
11 LOGLEVEL = 2
12 DEBUGFORMAT = False
13
14 import sys, tempfile
15 sys.path.append(PYTAN_LIB_PATH)
16
17 import pytan
18 handler = pytan.Handler(
19     username=USERNAME,
20     password=PASSWORD,
21     host=HOST,
22     port=PORT,
23     loglevel=LOGLEVEL,
24     debugformat=DEBUGFORMAT,
25 )
26
27 print handler
28
29 # setup the export_obj kwargs for later
30 export_kwargs = {}
31 export_kwargs["export_format"] = u'csv'
32
33 # ask the question that will provide the resultset that we want to use
34 ask_kwargs = {
35     'qtype': 'manual_human',
36     'sensors': [
37         "Computer Name", "IP Route Details", "IP Address",
38         'Folder Name Search with RegEx Match{dirname=Program Files,regex=.*Shared.*}',
39     ],

```

```

40 }
41 response = handler.ask(**ask_kwargs)
42
43 # export the object to a string
44 # (we could just as easily export to a file using export_to_report_file)
45 export_kwargs['obj'] = response['question_results']
46 export_str = handler.export_obj(**export_kwargs)
47
48
49 print ""
50 print "print the export_str returned from export_obj():"
51 if len(out.splitlines()) > 15:
52     out = out.splitlines()[0:15]
53     out.append('..trimmed for brevity..')
54     out = '\n'.join(out)
55
56 print out

```

### Output from Python Code

```

1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
2 2014-12-08 16:28:45,552 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detai
3 2014-12-08 16:28:50,583 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detai
4 2014-12-08 16:28:55,610 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detai
5 2014-12-08 16:29:00,640 INFO      question_progress: Results 17% (Get Computer Name and IP Route Deta
6 2014-12-08 16:29:05,668 INFO      question_progress: Results 17% (Get Computer Name and IP Route Deta
7 2014-12-08 16:29:10,699 INFO      question_progress: Results 50% (Get Computer Name and IP Route Deta
8 2014-12-08 16:29:15,728 INFO      question_progress: Results 67% (Get Computer Name and IP Route Deta
9 2014-12-08 16:29:20,762 INFO      question_progress: Results 83% (Get Computer Name and IP Route Deta
10 2014-12-08 16:29:25,794 INFO      question_progress: Results 100% (Get Computer Name and IP Route Deta
11
12 print the export_str returned from export_obj():
13 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
14 2014-12-08 16:28:45,385 INFO      handler: Report file '/var/folders/dk/vjrlr_c53yx6k6gzp2bbt_c40000g
15 Traceback (most recent call last):
16   File "<string>", line 51, in <module>
17   File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 484, in create_from_json
18     raise HandlerError(m(objtype, json_createable))
19 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 # Path to lib directory which contains pytan package
2 PYTAN_LIB_PATH = '../lib'
3
4 # connection info for Tanium Server
5 USERNAME = "Tanium User"
6 PASSWORD = "T@n!um"
7 HOST = "172.16.31.128"
8 PORT = "444"

```

```

9
10 # Logging controls
11 LOGLEVEL = 2
12 DEBUGFORMAT = False
13
14 import sys, tempfile
15 sys.path.append(PYTAN_LIB_PATH)
16
17 import pytan
18 handler = pytan.Handler(
19     username=USERNAME,
20     password=PASSWORD,
21     host=HOST,
22     port=PORT,
23     loglevel=LOGLEVEL,
24     debugformat=DEBUGFORMAT,
25 )
26
27 print handler
28
29 # setup the export_obj kwargs for later
30 export_kwargs = {}
31 export_kwargs["export_format"] = u'csv'
32 export_kwargs["expand_grouped_columns"] = False
33
34 # ask the question that will provide the resultset that we want to use
35 ask_kwargs = {
36     'qtype': 'manual_human',
37     'sensors': [
38         "Computer Name", "IP Route Details", "IP Address",
39         'Folder Name Search with RegEx Match{dirname=Program Files,regex=.*Shared.*}',
40     ],
41 }
42 response = handler.ask(**ask_kwargs)
43
44 # export the object to a string
45 # (we could just as easily export to a file using export_to_report_file)
46 export_kwargs['obj'] = response['question_results']
47 export_str = handler.export_obj(**export_kwargs)
48
49
50 print ""
51 print "print the export_str returned from export_obj():"
52 if len(out.splitlines()) > 15:
53     out = out.splitlines()[0:15]
54     out.append('..trimmed for brevity..')
55     out = '\n'.join(out)
56
57 print out

```

### Output from Python Code

```

1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
2 2014-12-08 16:29:25,976 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detai
3 2014-12-08 16:29:31,003 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detai
4 2014-12-08 16:29:36,035 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detai
5 2014-12-08 16:29:41,062 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detai

```

```

6 2014-12-08 16:29:46,088 INFO question_progress: Results 0% (Get Computer Name and IP Route Detai
7 2014-12-08 16:29:51,126 INFO question_progress: Results 0% (Get Computer Name and IP Route Detai
8 2014-12-08 16:29:56,153 INFO question_progress: Results 0% (Get Computer Name and IP Route Detai
9 2014-12-08 16:30:01,183 INFO question_progress: Results 0% (Get Computer Name and IP Route Detai
10 2014-12-08 16:30:06,213 INFO question_progress: Results 0% (Get Computer Name and IP Route Detai
11 2014-12-08 16:30:11,240 INFO question_progress: Results 0% (Get Computer Name and IP Route Detai
12 2014-12-08 16:30:16,269 INFO question_progress: Results 0% (Get Computer Name and IP Route Detai
13 2014-12-08 16:30:21,306 INFO question_progress: Results 0% (Get Computer Name and IP Route Detai
14 2014-12-08 16:30:26,331 INFO question_progress: Results 33% (Get Computer Name and IP Route Deta
15 2014-12-08 16:30:31,362 INFO question_progress: Results 100% (Get Computer Name and IP Route Deta
16
17 print the export_str returned from export_obj():
18 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
19 2014-12-08 16:28:45,552 INFO question_progress: Results 0% (Get Computer Name and IP Route Detai
20 2014-12-08 16:28:50,583 INFO question_progress: Results 0% (Get Computer Name and IP Route Detai
21 2014-12-08 16:28:55,610 INFO question_progress: Results 0% (Get Computer Name and IP Route Detai
22 2014-12-08 16:29:00,640 INFO question_progress: Results 17% (Get Computer Name and IP Route Deta
23 2014-12-08 16:29:05,668 INFO question_progress: Results 17% (Get Computer Name and IP Route Deta
24 2014-12-08 16:29:10,699 INFO question_progress: Results 50% (Get Computer Name and IP Route Deta
25 2014-12-08 16:29:15,728 INFO question_progress: Results 67% (Get Computer Name and IP Route Deta
26 2014-12-08 16:29:20,762 INFO question_progress: Results 83% (Get Computer Name and IP Route Deta
27 2014-12-08 16:29:25,794 INFO question_progress: Results 100% (Get Computer Name and IP Route Deta
28
29 print the export_str returned from export_obj():
30 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
31 2014-12-08 16:28:45,385 INFO handler: Report file '/var/folders/dk/vjrlr_c53yx6k6gzp2bbt_c40000g
32 Traceback (most recent call last):
33 ..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 # Path to lib directory which contains pytan package
2 PYTAN_LIB_PATH = '../lib'
3
4 # connection info for Tanium Server
5 USERNAME = "Tanium User"
6 PASSWORD = "T@n!um"
7 HOST = "172.16.31.128"
8 PORT = "444"
9
10 # Logging conrols
11 LOGLEVEL = 2
12 DEBUGFORMAT = False
13
14 import sys, tempfile
15 sys.path.append(PYTAN_LIB_PATH)
16
17 import pytan
18 handler = pytan.Handler(
19     username=USERNAME,
20     password=PASSWORD,

```

```

21     host=HOST,
22     port=PORT,
23     loglevel=LOGLEVEL,
24     debugformat=DEBUGFORMAT,
25 )
26
27 print handler
28
29 # setup the export_obj kwargs for later
30 export_kwargs = {}
31 export_kwargs["export_format"] = u'csv'
32 export_kwargs["expand_grouped_columns"] = True
33
34 # ask the question that will provide the resultset that we want to use
35 ask_kwargs = {
36     'qtype': 'manual_human',
37     'sensors': [
38         "Computer Name", "IP Route Details", "IP Address",
39         'Folder Name Search with RegEx Match{dirname=Program Files,regex=.*Shared.*}',
40     ],
41 }
42 response = handler.ask(**ask_kwargs)
43
44 # export the object to a string
45 # (we could just as easily export to a file using export_to_report_file)
46 export_kwargs['obj'] = response['question_results']
47 export_str = handler.export_obj(**export_kwargs)
48
49
50 print ""
51 print "print the export_str returned from export_obj():"
52 if len(out.splitlines()) > 15:
53     out = out.splitlines()[0:15]
54     out.append('..trimmed for brevity..')
55     out = '\n'.join(out)
56
57 print out

```

### Output from Python Code

```

1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
2 2014-12-08 16:30:31,572 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detai
3 2014-12-08 16:30:36,599 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detai
4 2014-12-08 16:30:41,628 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detai
5 2014-12-08 16:30:46,653 INFO      question_progress: Results 17% (Get Computer Name and IP Route Deta
6 2014-12-08 16:30:51,682 INFO      question_progress: Results 50% (Get Computer Name and IP Route Deta
7 2014-12-08 16:30:56,710 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: 6.2.314.3258
11 2014-12-08 16:29:25,976 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detai
12 2014-12-08 16:29:31,003 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detai
13 2014-12-08 16:29:36,035 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detai
14 2014-12-08 16:29:41,062 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detai
15 2014-12-08 16:29:46,088 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detai
16 2014-12-08 16:29:51,126 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detai
17 2014-12-08 16:29:56,153 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detai

```

```
18 2014-12-08 16:30:01,183 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
19 2014-12-08 16:30:06,213 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
20 2014-12-08 16:30:11,240 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
21 2014-12-08 16:30:16,269 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
22 2014-12-08 16:30:21,306 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
23 2014-12-08 16:30:26,331 INFO question_progress: Results 33% (Get Computer Name and IP Route Detail
24 2014-12-08 16:30:31,362 INFO question_progress: Results 100% (Get Computer Name and IP Route Detail
25 ..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 # Path to lib directory which contains pytan package
2 PYTAN_LIB_PATH = '../lib'
3
4 # connection info for Tanium Server
5 USERNAME = "Tanium User"
6 PASSWORD = "T@n!um"
7 HOST = "172.16.31.128"
8 PORT = "444"
9
10 # Logging conrols
11 LOGLEVEL = 2
12 DEBUGFORMAT = False
13
14 import sys, tempfile
15 sys.path.append(PYTAN_LIB_PATH)
16
17 import pytan
18 handler = pytan.Handler(
19     username=USERNAME,
20     password=PASSWORD,
21     host=HOST,
22     port=PORT,
23     loglevel=LOGLEVEL,
24     debugformat=DEBUGFORMAT,
25 )
26
27 print handler
28
29 # setup the export_obj kwargs for later
30 export_kwargs = {}
31 export_kwargs["header_sort"] = True
32 export_kwargs["export_format"] = u'csv'
33 export_kwargs["header_add_type"] = True
34 export_kwargs["expand_grouped_columns"] = True
35 export_kwargs["header_add_sensor"] = True
36
37 # ask the question that will provide the resultset that we want to use
38 ask_kwargs = {
39     'qtype': 'manual_human',
```

```

40     'sensors': [
41         "Computer Name", "IP Route Details", "IP Address",
42         'Folder Name Search with RegEx Match{dirname=Program Files,regex=.*Shared.*}',
43     ],
44 }
45 response = handler.ask(**ask_kwargs)
46
47 # export the object to a string
48 # (we could just as easily export to a file using export_to_report_file)
49 export_kwargs['obj'] = response['question_results']
50 export_str = handler.export_obj(**export_kwargs)
51
52
53 print ""
54 print "print the export_str returned from export_obj():"
55 if len(out.splitlines()) > 15:
56     out = out.splitlines()[0:15]
57     out.append('..trimmed for brevity..')
58     out = '\n'.join(out)
59
60 print out

```

### Output from Python Code

```

1  Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
2  2014-12-08 16:30:56,943 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detai
3  2014-12-08 16:31:01,971 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detai
4  2014-12-08 16:31:07,002 INFO      question_progress: Results 33% (Get Computer Name and IP Route Deta
5  2014-12-08 16:31:12,030 INFO      question_progress: Results 67% (Get Computer Name and IP Route Deta
6  2014-12-08 16:31:17,058 INFO      question_progress: Results 83% (Get Computer Name and IP Route Deta
7  2014-12-08 16:31:22,085 INFO      question_progress: Results 83% (Get Computer Name and IP Route Deta
8  2014-12-08 16:31:27,114 INFO      question_progress: Results 83% (Get Computer Name and IP Route Deta
9  2014-12-08 16:31:32,138 INFO      question_progress: Results 83% (Get Computer Name and IP Route Deta
10 2014-12-08 16:31:37,164 INFO      question_progress: Results 100% (Get Computer Name and IP Route Deta
11
12 print the export_str returned from export_obj():
13 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
14 2014-12-08 16:30:31,572 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detai
15 2014-12-08 16:30:36,599 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detai
16 2014-12-08 16:30:41,628 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detai
17 2014-12-08 16:30:46,653 INFO      question_progress: Results 17% (Get Computer Name and IP Route Deta
18 2014-12-08 16:30:51,682 INFO      question_progress: Results 50% (Get Computer Name and IP Route Deta
19 2014-12-08 16:30:56,710 INFO      question_progress: Results 100% (Get Computer Name and IP Route Deta
20
21 print the export_str returned from export_obj():
22 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
23 2014-12-08 16:29:25,976 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detai
24 2014-12-08 16:29:31,003 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detai
25 2014-12-08 16:29:36,035 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detai
26 2014-12-08 16:29:41,062 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detai
27 2014-12-08 16:29:46,088 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detai
28 ..trimmed for brevity..

```

## Export resultset json

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

### Example Python Code

```
1  # Path to lib directory which contains pytan package
2  PYTAN_LIB_PATH = '../lib'
3
4  # connection info for Tanium Server
5  USERNAME = "Tanium User"
6  PASSWORD = "T@n!um"
7  HOST = "172.16.31.128"
8  PORT = "444"
9
10 # Logging controls
11 LOGLEVEL = 2
12 DEBUGFORMAT = False
13
14 import sys, tempfile
15 sys.path.append(PYTAN_LIB_PATH)
16
17 import pytan
18 handler = pytan.Handler(
19     username=USERNAME,
20     password=PASSWORD,
21     host=HOST,
22     port=PORT,
23     loglevel=LOGLEVEL,
24     debugformat=DEBUGFORMAT,
25 )
26
27 print handler
28
29 # setup the export_obj kwargs for later
30 export_kwargs = {}
31 export_kwargs["export_format"] = u'json'
32
33 # ask the question that will provide the resultset that we want to use
34 ask_kwargs = {
35     'qtype': 'manual_human',
36     'sensors': [
37         "Computer Name", "IP Route Details", "IP Address",
38         'Folder Name Search with RegEx Match{dirname=Program Files,regex=.*Shared.*}',
39     ],
40 }
41 response = handler.ask(**ask_kwargs)
42
43 # export the object to a string
44 # (we could just as easily export to a file using export_to_report_file)
45 export_kwargs['obj'] = response['question_results']
46 export_str = handler.export_obj(**export_kwargs)
47
48
49 print ""
50 print "print the export_str returned from export_obj():"
51 if len(out.splitlines()) > 15:
52     out = out.splitlines()[0:15]
```



```

53     out.append('..trimmed for brevity..')
54     out = '\n'.join(out)
55
56 print out

```

### Output from Python Code

```

1  Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
2  2014-12-08 16:31:37,503 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detai
3  2014-12-08 16:31:42,529 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detai
4  2014-12-08 16:31:47,559 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detai
5  2014-12-08 16:31:52,587 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detai
6  2014-12-08 16:31:57,615 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detai
7  2014-12-08 16:32:02,640 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detai
8  2014-12-08 16:32:07,665 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detai
9  2014-12-08 16:32:12,693 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detai
10 2014-12-08 16:32:17,719 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detai
11 2014-12-08 16:32:22,747 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detai
12 2014-12-08 16:32:27,775 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detai
13 2014-12-08 16:32:32,805 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detai
14 2014-12-08 16:32:37,831 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detai
15 2014-12-08 16:32:42,858 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detai
16 2014-12-08 16:32:47,887 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detai
17 2014-12-08 16:32:52,922 INFO      question_progress: Results 17% (Get Computer Name and IP Route Deta
18 2014-12-08 16:32:57,948 INFO      question_progress: Results 50% (Get Computer Name and IP Route Deta
19 2014-12-08 16:33:02,976 INFO      question_progress: Results 100% (Get Computer Name and IP Route Deta
20
21 print the export_str returned from export_obj():
22 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
23 2014-12-08 16:30:56,943 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detai
24 2014-12-08 16:31:01,971 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detai
25 2014-12-08 16:31:07,002 INFO      question_progress: Results 33% (Get Computer Name and IP Route Deta
26 2014-12-08 16:31:12,030 INFO      question_progress: Results 67% (Get Computer Name and IP Route Deta
27 2014-12-08 16:31:17,058 INFO      question_progress: Results 83% (Get Computer Name and IP Route Deta
28 2014-12-08 16:31:22,085 INFO      question_progress: Results 83% (Get Computer Name and IP Route Deta
29 2014-12-08 16:31:27,114 INFO      question_progress: Results 83% (Get Computer Name and IP Route Deta
30 2014-12-08 16:31:32,138 INFO      question_progress: Results 83% (Get Computer Name and IP Route Deta
31 2014-12-08 16:31:37,164 INFO      question_progress: Results 100% (Get Computer Name and IP Route Deta
32
33 print the export_str returned from export_obj():
34 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
35 2014-12-08 16:30:31,572 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detai
36 2014-12-08 16:30:36,599 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detai
37 ..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  # Path to lib directory which contains pytan package
2  PYTAN_LIB_PATH = '../lib'
3

```

```
4  # connection info for Tanium Server
5  USERNAME = "Tanium User"
6  PASSWORD = "T@n!um"
7  HOST = "172.16.31.128"
8  PORT = "444"
9
10 # Logging conrols
11 LOGLEVEL = 2
12 DEBUGFORMAT = False
13
14 import sys, tempfile
15 sys.path.append(PYTAN_LIB_PATH)
16
17 import pytan
18 handler = pytan.Handler(
19     username=USERNAME,
20     password=PASSWORD,
21     host=HOST,
22     port=PORT,
23     loglevel=LOGLEVEL,
24     debugformat=DEBUGFORMAT,
25 )
26
27 print handler
28
29 # setup the export_obj kwargs for later
30 export_kwargs = {}
31 export_kwargs["export_format"] = u'csv'
32 export_kwargs["header_sort"] = []
33
34 # ask the question that will provide the resultset that we want to use
35 ask_kwargs = {
36     'qtype': 'manual_human',
37     'sensors': [
38         "Computer Name", "IP Route Details", "IP Address",
39         'Folder Name Search with RegEx Match{dirname=Program Files,regex=.*Shared.*}',
40     ],
41 }
42 response = handler.ask(**ask_kwargs)
43
44 # export the object to a string
45 # (we could just as easily export to a file using export_to_report_file)
46 export_kwargs['obj'] = response['question_results']
47 export_str = handler.export_obj(**export_kwargs)
48
49
50 print ""
51 print "print the export_str returned from export_obj():"
52 if len(out.splitlines()) > 15:
53     out = out.splitlines()[0:15]
54     out.append('..trimmed for brevity..')
55     out = '\n'.join(out)
56
57 print out
```

## Output from Python Code

```

1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
2 2014-12-08 16:33:03,175 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
3 2014-12-08 16:33:08,203 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
4 2014-12-08 16:33:13,228 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
5 2014-12-08 16:33:18,257 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
6 2014-12-08 16:33:23,282 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
7 2014-12-08 16:33:28,308 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
8 2014-12-08 16:33:33,335 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
9 2014-12-08 16:33:38,362 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
10 2014-12-08 16:33:43,390 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
11 2014-12-08 16:33:48,418 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
12 2014-12-08 16:33:53,454 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
13 2014-12-08 16:33:58,481 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
14 2014-12-08 16:34:03,510 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
15 2014-12-08 16:34:08,539 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
16 2014-12-08 16:34:13,572 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
17 2014-12-08 16:34:18,603 INFO question_progress: Results 67% (Get Computer Name and IP Route Detail
18 2014-12-08 16:34:23,630 INFO question_progress: Results 83% (Get Computer Name and IP Route Detail
19 2014-12-08 16:34:28,657 INFO question_progress: Results 83% (Get Computer Name and IP Route Detail
20 2014-12-08 16:34:33,686 INFO question_progress: Results 83% (Get Computer Name and IP Route Detail
21 2014-12-08 16:34:38,716 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: 6.2.314.3258
25 2014-12-08 16:31:37,503 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
26 2014-12-08 16:31:42,529 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
27 2014-12-08 16:31:47,559 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
28 2014-12-08 16:31:52,587 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
29 2014-12-08 16:31:57,615 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
30 2014-12-08 16:32:02,640 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
31 2014-12-08 16:32:07,665 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
32 2014-12-08 16:32:12,693 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
33 2014-12-08 16:32:17,719 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
34 2014-12-08 16:32:22,747 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
35 2014-12-08 16:32:27,775 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
36 2014-12-08 16:32:32,805 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
37 2014-12-08 16:32:37,831 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
38 2014-12-08 16:32:42,858 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 # Path to lib directory which contains pytan package
2 PYTAN_LIB_PATH = '../lib'
3
4 # connection info for Tanium Server
5 USERNAME = "Tanium User"
6 PASSWORD = "T@n!um"
7 HOST = "172.16.31.128"
8 PORT = "444"
9

```

```

10 # Logging controls
11 LOGLEVEL = 2
12 DEBUGFORMAT = False
13
14 import sys, tempfile
15 sys.path.append(PYTAN_LIB_PATH)
16
17 import pytan
18 handler = pytan.Handler(
19     username=USERNAME,
20     password=PASSWORD,
21     host=HOST,
22     port=PORT,
23     loglevel=LOGLEVEL,
24     debugformat=DEBUGFORMAT,
25 )
26
27 print handler
28
29 # setup the export_obj kwargs for later
30 export_kwargs = {}
31 export_kwargs["export_format"] = u'csv'
32 export_kwargs["header_sort"] = True
33
34 # ask the question that will provide the resultset that we want to use
35 ask_kwargs = {
36     'qtype': 'manual_human',
37     'sensors': [
38         "Computer Name", "IP Route Details", "IP Address",
39         'Folder Name Search with RegEx Match{dirname=Program Files,regex=.*Shared.*}',
40     ],
41 }
42 response = handler.ask(**ask_kwargs)
43
44 # export the object to a string
45 # (we could just as easily export to a file using export_to_report_file)
46 export_kwargs['obj'] = response['question_results']
47 export_str = handler.export_obj(**export_kwargs)
48
49
50 print ""
51 print "print the export_str returned from export_obj():"
52 if len(out.splitlines()) > 15:
53     out = out.splitlines()[0:15]
54     out.append('..trimmed for brevity..')
55     out = '\n'.join(out)
56
57 print out

```

### Output from Python Code

```

1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
2 2014-12-08 16:34:38,921 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detai
3 2014-12-08 16:34:43,950 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detai
4 2014-12-08 16:34:48,987 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detai
5 2014-12-08 16:34:54,015 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detai
6 2014-12-08 16:34:59,046 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detai

```

```

7 2014-12-08 16:35:04,073 INFO question_progress: Results 0% (Get Computer Name and IP Route Detai
8 2014-12-08 16:35:09,098 INFO question_progress: Results 0% (Get Computer Name and IP Route Detai
9 2014-12-08 16:35:14,126 INFO question_progress: Results 0% (Get Computer Name and IP Route Detai
10 2014-12-08 16:35:19,155 INFO question_progress: Results 0% (Get Computer Name and IP Route Detai
11 2014-12-08 16:35:24,182 INFO question_progress: Results 0% (Get Computer Name and IP Route Detai
12 2014-12-08 16:35:29,213 INFO question_progress: Results 0% (Get Computer Name and IP Route Detai
13 2014-12-08 16:35:34,241 INFO question_progress: Results 0% (Get Computer Name and IP Route Detai
14 2014-12-08 16:35:39,272 INFO question_progress: Results 0% (Get Computer Name and IP Route Detai
15 2014-12-08 16:35:44,296 INFO question_progress: Results 0% (Get Computer Name and IP Route Detai
16 2014-12-08 16:35:49,327 INFO question_progress: Results 0% (Get Computer Name and IP Route Detai
17 2014-12-08 16:35:54,353 INFO question_progress: Results 0% (Get Computer Name and IP Route Detai
18 2014-12-08 16:35:59,383 INFO question_progress: Results 0% (Get Computer Name and IP Route Detai
19 2014-12-08 16:36:04,419 INFO question_progress: Results 50% (Get Computer Name and IP Route Deta
20 2014-12-08 16:36:09,449 INFO question_progress: Results 50% (Get Computer Name and IP Route Deta
21 2014-12-08 16:36:14,473 INFO question_progress: Results 50% (Get Computer Name and IP Route Deta
22 2014-12-08 16:36:19,502 INFO question_progress: Results 67% (Get Computer Name and IP Route Deta
23 2014-12-08 16:36:24,534 INFO question_progress: Results 100% (Get Computer Name and IP Route Deta
24
25 print the export_str returned from export_obj():
26 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
27 2014-12-08 16:33:03,175 INFO question_progress: Results 0% (Get Computer Name and IP Route Detai
28 2014-12-08 16:33:08,203 INFO question_progress: Results 0% (Get Computer Name and IP Route Detai
29 2014-12-08 16:33:13,228 INFO question_progress: Results 0% (Get Computer Name and IP Route Detai
30 2014-12-08 16:33:18,257 INFO question_progress: Results 0% (Get Computer Name and IP Route Detai
31 2014-12-08 16:33:23,282 INFO question_progress: Results 0% (Get Computer Name and IP Route Detai
32 2014-12-08 16:33:28,308 INFO question_progress: Results 0% (Get Computer Name and IP Route Detai
33 2014-12-08 16:33:33,335 INFO question_progress: Results 0% (Get Computer Name and IP Route Detai
34 2014-12-08 16:33:38,362 INFO question_progress: Results 0% (Get Computer Name and IP Route Detai
35 2014-12-08 16:33:43,390 INFO question_progress: Results 0% (Get Computer Name and IP Route Detai
36 2014-12-08 16:33:48,418 INFO question_progress: Results 0% (Get Computer Name and IP Route Detai
37 2014-12-08 16:33:53,454 INFO question_progress: Results 0% (Get Computer Name and IP Route Detai
38 2014-12-08 16:33:58,481 INFO question_progress: Results 0% (Get Computer Name and IP Route Detai
39 2014-12-08 16:34:03,510 INFO question_progress: Results 0% (Get Computer Name and IP Route Detai
40 2014-12-08 16:34:08,539 INFO question_progress: Results 0% (Get Computer Name and IP Route Detai
41 ..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 # Path to lib directory which contains pytan package
2 PYTAN_LIB_PATH = '../lib'
3
4 # connection info for Tanium Server
5 USERNAME = "Tanium User"
6 PASSWORD = "T@n!um"
7 HOST = "172.16.31.128"
8 PORT = "444"
9
10 # Logging conrols
11 LOGLEVEL = 2
12 DEBUGFORMAT = False
13

```

```

14 import sys, tempfile
15 sys.path.append(PYTAN_LIB_PATH)
16
17 import pytan
18 handler = pytan.Handler(
19     username=USERNAME,
20     password=PASSWORD,
21     host=HOST,
22     port=PORT,
23     loglevel=LOGLEVEL,
24     debugformat=DEBUGFORMAT,
25 )
26
27 print handler
28
29 # setup the export_obj kwargs for later
30 export_kwargs = {}
31 export_kwargs["export_format"] = u'csv'
32 export_kwargs["header_sort"] = False
33
34 # ask the question that will provide the resultset that we want to use
35 ask_kwargs = {
36     'qtype': 'manual_human',
37     'sensors': [
38         "Computer Name", "IP Route Details", "IP Address",
39         'Folder Name Search with RegEx Match{dirname=Program Files,regex=.*Shared.*}',
40     ],
41 }
42 response = handler.ask(**ask_kwargs)
43
44 # export the object to a string
45 # (we could just as easily export to a file using export_to_report_file)
46 export_kwargs['obj'] = response['question_results']
47 export_str = handler.export_obj(**export_kwargs)
48
49
50 print ""
51 print "print the export_str returned from export_obj():"
52 if len(out.splitlines()) > 15:
53     out = out.splitlines()[0:15]
54     out.append('..trimmed for brevity..')
55     out = '\n'.join(out)
56
57 print out

```

### Output from Python Code

```

1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
2 2014-12-08 16:36:24,732 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detai
3 2014-12-08 16:36:29,758 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detai
4 2014-12-08 16:36:34,793 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detai
5 2014-12-08 16:36:39,821 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detai
6 2014-12-08 16:36:44,845 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detai
7 2014-12-08 16:36:49,878 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detai
8 2014-12-08 16:36:54,905 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detai
9 2014-12-08 16:36:59,941 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detai
10 2014-12-08 16:37:04,967 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detai

```

```

11 2014-12-08 16:37:09,995 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
12 2014-12-08 16:37:15,020 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
13 2014-12-08 16:37:20,047 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
14 2014-12-08 16:37:25,076 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
15 2014-12-08 16:37:30,103 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
16 2014-12-08 16:37:35,131 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
17 2014-12-08 16:37:40,159 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
18 2014-12-08 16:37:45,193 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
19 2014-12-08 16:37:50,224 INFO question_progress: Results 50% (Get Computer Name and IP Route Detail
20 2014-12-08 16:37:55,251 INFO question_progress: Results 83% (Get Computer Name and IP Route Detail
21 2014-12-08 16:38:00,284 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: 6.2.314.3258
25 2014-12-08 16:34:38,921 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
26 2014-12-08 16:34:43,950 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
27 2014-12-08 16:34:48,987 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
28 2014-12-08 16:34:54,015 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
29 2014-12-08 16:34:59,046 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
30 2014-12-08 16:35:04,073 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
31 2014-12-08 16:35:09,098 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
32 2014-12-08 16:35:14,126 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
33 2014-12-08 16:35:19,155 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
34 2014-12-08 16:35:24,182 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
35 2014-12-08 16:35:29,213 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
36 2014-12-08 16:35:34,241 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
37 2014-12-08 16:35:39,272 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
38 2014-12-08 16:35:44,296 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
39 ..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  # Path to lib directory which contains pytan package
2  PYTAN_LIB_PATH = '../lib'
3
4  # connection info for Tanium Server
5  USERNAME = "Tanium User"
6  PASSWORD = "T@n!um"
7  HOST = "172.16.31.128"
8  PORT = "444"
9
10 # Logging conrols
11 LOGLEVEL = 2
12 DEBUGFORMAT = False
13
14 import sys, tempfile
15 sys.path.append(PYTAN_LIB_PATH)
16
17 import pytan
18 handler = pytan.Handler(
19     username=USERNAME,

```

```

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

```

### Output from Python Code

```

1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
2 2014-12-08 16:38:00,467 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detai
3 2014-12-08 16:38:05,497 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detai
4 2014-12-08 16:38:10,524 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detai
5 2014-12-08 16:38:15,551 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detai
6 2014-12-08 16:38:20,578 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detai
7 2014-12-08 16:38:25,604 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detai
8 2014-12-08 16:38:30,628 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detai
9 2014-12-08 16:38:35,653 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detai
10 2014-12-08 16:38:40,679 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detai
11 2014-12-08 16:38:45,708 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detai
12 2014-12-08 16:38:50,735 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detai
13 2014-12-08 16:38:55,768 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detai
14 2014-12-08 16:39:00,797 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detai
15 2014-12-08 16:39:05,823 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detai
16 2014-12-08 16:39:10,845 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detai

```



```

17 2014-12-08 16:39:15,876 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
18 2014-12-08 16:39:20,904 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
19 2014-12-08 16:39:25,930 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
20 2014-12-08 16:39:30,955 INFO question_progress: Results 83% (Get Computer Name and IP Route Detail
21 2014-12-08 16:39:35,982 INFO question_progress: Results 83% (Get Computer Name and IP Route Detail
22 2014-12-08 16:39:41,010 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: 6.2.314.3258
26 2014-12-08 16:36:24,732 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
27 2014-12-08 16:36:29,758 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
28 2014-12-08 16:36:34,793 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
29 2014-12-08 16:36:39,821 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
30 2014-12-08 16:36:44,845 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
31 2014-12-08 16:36:49,878 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
32 2014-12-08 16:36:54,905 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
33 2014-12-08 16:36:59,941 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
34 2014-12-08 16:37:04,967 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
35 2014-12-08 16:37:09,995 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
36 2014-12-08 16:37:15,020 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
37 2014-12-08 16:37:20,047 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
38 2014-12-08 16:37:25,076 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
39 2014-12-08 16:37:30,103 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
40 ..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  # Path to lib directory which contains pytan package
2  PYTAN_LIB_PATH = '../lib'
3
4  # connection info for Tanium Server
5  USERNAME = "Tanium User"
6  PASSWORD = "T@n!um"
7  HOST = "172.16.31.128"
8  PORT = "444"
9
10 # Logging conrols
11 LOGLEVEL = 2
12 DEBUGFORMAT = False
13
14 import sys, tempfile
15 sys.path.append(PYTAN_LIB_PATH)
16
17 import pytan
18 handler = pytan.Handler(
19     username=USERNAME,
20     password=PASSWORD,
21     host=HOST,
22     port=PORT,
23     loglevel=LOGLEVEL,
24     debugformat=DEBUGFORMAT,

```

```

25 )
26
27 print handler
28
29 # setup the export_obj kwargs for later
30 export_kwargs = {}
31 export_kwargs["export_format"] = u'csv'
32 export_kwargs["header_add_type"] = False
33
34 # ask the question that will provide the resultset that we want to use
35 ask_kwargs = {
36     'qtype': 'manual_human',
37     'sensors': [
38         "Computer Name", "IP Route Details", "IP Address",
39         'Folder Name Search with RegEx Match{dirname=Program Files,regex=.*Shared.*}',
40     ],
41 }
42 response = handler.ask(**ask_kwargs)
43
44 # export the object to a string
45 # (we could just as easily export to a file using export_to_report_file)
46 export_kwargs['obj'] = response['question_results']
47 export_str = handler.export_obj(**export_kwargs)
48
49
50 print ""
51 print "print the export_str returned from export_obj():"
52 if len(out.splitlines()) > 15:
53     out = out.splitlines()[0:15]
54     out.append('..trimmed for brevity..')
55     out = '\n'.join(out)
56
57 print out

```

## Output from Python Code

```

1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
2 2014-12-08 16:39:41,204 INFO question_progress: Results 0% (Get Computer Name and IP Route Detai
3 2014-12-08 16:39:46,233 INFO question_progress: Results 0% (Get Computer Name and IP Route Detai
4 2014-12-08 16:39:51,260 INFO question_progress: Results 0% (Get Computer Name and IP Route Detai
5 2014-12-08 16:39:56,286 INFO question_progress: Results 0% (Get Computer Name and IP Route Detai
6 2014-12-08 16:40:01,313 INFO question_progress: Results 0% (Get Computer Name and IP Route Detai
7 2014-12-08 16:40:06,340 INFO question_progress: Results 0% (Get Computer Name and IP Route Detai
8 2014-12-08 16:40:11,373 INFO question_progress: Results 0% (Get Computer Name and IP Route Detai
9 2014-12-08 16:40:16,400 INFO question_progress: Results 0% (Get Computer Name and IP Route Detai
10 2014-12-08 16:40:21,427 INFO question_progress: Results 0% (Get Computer Name and IP Route Detai
11 2014-12-08 16:40:26,453 INFO question_progress: Results 50% (Get Computer Name and IP Route Deta
12 2014-12-08 16:40:31,481 INFO question_progress: Results 83% (Get Computer Name and IP Route Deta
13 2014-12-08 16:40:36,509 INFO question_progress: Results 83% (Get Computer Name and IP Route Deta
14 2014-12-08 16:40:41,536 INFO question_progress: Results 83% (Get Computer Name and IP Route Deta
15 2014-12-08 16:40:46,562 INFO question_progress: Results 83% (Get Computer Name and IP Route Deta
16 2014-12-08 16:40:51,588 INFO question_progress: Results 83% (Get Computer Name and IP Route Deta
17 2014-12-08 16:40:56,613 INFO question_progress: Results 83% (Get Computer Name and IP Route Deta
18 2014-12-08 16:41:01,643 INFO question_progress: Results 83% (Get Computer Name and IP Route Deta
19 2014-12-08 16:41:06,670 INFO question_progress: Results 83% (Get Computer Name and IP Route Deta
20 2014-12-08 16:41:11,694 INFO question_progress: Results 83% (Get Computer Name and IP Route Deta
21 2014-12-08 16:41:16,719 INFO question_progress: Results 83% (Get Computer Name and IP Route Deta

```

```

22 2014-12-08 16:41:21,745 INFO      question_progress: Results 83% (Get Computer Name and IP Route Deta
23 2014-12-08 16:41:26,778 INFO      question_progress: Results 83% (Get Computer Name and IP Route Deta
24 2014-12-08 16:41:31,805 INFO      question_progress: Results 83% (Get Computer Name and IP Route Deta
25 2014-12-08 16:41:36,836 INFO      question_progress: Results 83% (Get Computer Name and IP Route Deta
26 2014-12-08 16:41:41,863 INFO      question_progress: Results 100% (Get Computer Name and IP Route Deta
27
28 print the export_str returned from export_obj():
29 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
30 2014-12-08 16:38:00,467 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detai
31 2014-12-08 16:38:05,497 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detai
32 2014-12-08 16:38:10,524 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detai
33 2014-12-08 16:38:15,551 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detai
34 2014-12-08 16:38:20,578 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detai
35 2014-12-08 16:38:25,604 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detai
36 2014-12-08 16:38:30,628 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detai
37 2014-12-08 16:38:35,653 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detai
38 2014-12-08 16:38:40,679 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detai
39 2014-12-08 16:38:45,708 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detai
40 2014-12-08 16:38:50,735 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detai
41 2014-12-08 16:38:55,768 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detai
42 2014-12-08 16:39:00,797 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detai
43 2014-12-08 16:39:05,823 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detai
44 ..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  # Path to lib directory which contains pytan package
2  PYTAN_LIB_PATH = '../lib'
3
4  # connection info for Tanium Server
5  USERNAME = "Tanium User"
6  PASSWORD = "T@n!um"
7  HOST = "172.16.31.128"
8  PORT = "444"
9
10 # Logging conrols
11 LOGLEVEL = 2
12 DEBUGFORMAT = False
13
14 import sys, tempfile
15 sys.path.append(PYTAN_LIB_PATH)
16
17 import pytan
18 handler = pytan.Handler(
19     username=USERNAME,
20     password=PASSWORD,
21     host=HOST,
22     port=PORT,
23     loglevel=LOGLEVEL,
24     debugformat=DEBUGFORMAT,
25 )

```

```

26
27 print handler
28
29 # setup the export_obj kwargs for later
30 export_kwargs = {}
31 export_kwargs["export_format"] = u'csv'
32 export_kwargs["header_add_type"] = True
33
34 # ask the question that will provide the resultset that we want to use
35 ask_kwargs = {
36     'qtype': 'manual_human',
37     'sensors': [
38         "Computer Name", "IP Route Details", "IP Address",
39         'Folder Name Search with RegEx Match{dirname=Program Files,regex=.*Shared.*}',
40     ],
41 }
42 response = handler.ask(**ask_kwargs)
43
44 # export the object to a string
45 # (we could just as easily export to a file using export_to_report_file)
46 export_kwargs['obj'] = response['question_results']
47 export_str = handler.export_obj(**export_kwargs)
48
49
50 print ""
51 print "print the export_str returned from export_obj():"
52 if len(out.splitlines()) > 15:
53     out = out.splitlines()[0:15]
54     out.append('..trimmed for brevity..')
55     out = '\n'.join(out)
56
57 print out

```

## Output from Python Code

```

1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
2 2014-12-08 16:41:42,048 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
3 2014-12-08 16:41:47,073 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
4 2014-12-08 16:41:52,100 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
5 2014-12-08 16:41:57,128 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
6 2014-12-08 16:42:02,151 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
7 2014-12-08 16:42:07,176 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
8 2014-12-08 16:42:12,200 INFO question_progress: Results 33% (Get Computer Name and IP Route Deta
9 2014-12-08 16:42:17,226 INFO question_progress: Results 100% (Get Computer Name and IP Route Det
10
11 print the export_str returned from export_obj():
12 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
13 2014-12-08 16:39:41,204 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
14 2014-12-08 16:39:46,233 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
15 2014-12-08 16:39:51,260 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
16 2014-12-08 16:39:56,286 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
17 2014-12-08 16:40:01,313 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
18 2014-12-08 16:40:06,340 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
19 2014-12-08 16:40:11,373 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
20 2014-12-08 16:40:16,400 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
21 2014-12-08 16:40:21,427 INFO question_progress: Results 0% (Get Computer Name and IP Route Detail
22 2014-12-08 16:40:26,453 INFO question_progress: Results 50% (Get Computer Name and IP Route Deta

```

```

23 2014-12-08 16:40:31,481 INFO      question_progress: Results 83% (Get Computer Name and IP Route Deta
24 2014-12-08 16:40:36,509 INFO      question_progress: Results 83% (Get Computer Name and IP Route Deta
25 2014-12-08 16:40:41,536 INFO      question_progress: Results 83% (Get Computer Name and IP Route Deta
26 2014-12-08 16:40:46,562 INFO      question_progress: Results 83% (Get Computer Name and IP Route Deta
27 ..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  # Path to lib directory which contains pytan package
2  PYTAN_LIB_PATH = '../lib'
3
4  # connection info for Tanium Server
5  USERNAME = "Tanium User"
6  PASSWORD = "T@n!um"
7  HOST = "172.16.31.128"
8  PORT = "444"
9
10 # Logging conrols
11 LOGLEVEL = 2
12 DEBUGFORMAT = False
13
14 import sys, tempfile
15 sys.path.append(PYTAN_LIB_PATH)
16
17 import pytan
18 handler = pytan.Handler(
19     username=USERNAME,
20     password=PASSWORD,
21     host=HOST,
22     port=PORT,
23     loglevel=LOGLEVEL,
24     debugformat=DEBUGFORMAT,
25 )
26
27 print handler
28
29 # setup the export_obj kwargs for later
30 export_kwargs = {}
31 export_kwargs["export_format"] = u'csv'
32 export_kwargs["header_add_sensor"] = False
33
34 # ask the question that will provide the resultset that we want to use
35 ask_kwargs = {
36     'qtype': 'manual_human',
37     'sensors': [
38         "Computer Name", "IP Route Details", "IP Address",
39         'Folder Name Search with RegEx Match{dirname=Program Files,regex=.*Shared.*}',
40     ],
41 }
42 response = handler.ask(**ask_kwargs)
43

```

```
44 # export the object to a string
45 # (we could just as easily export to a file using export_to_report_file)
46 export_kwargs['obj'] = response['question_results']
47 export_str = handler.export_obj(**export_kwargs)
48
49
50 print ""
51 print "print the export_str returned from export_obj():"
52 if len(out.splitlines()) > 15:
53     out = out.splitlines()[0:15]
54     out.append('..trimmed for brevity..')
55     out = '\n'.join(out)
56
57 print out
```

### Output from Python Code

```
1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
2 2014-12-08 16:42:17,402 INFO question_progress: Results 0% (Get Computer Name and IP Route Detai
3 2014-12-08 16:42:22,428 INFO question_progress: Results 0% (Get Computer Name and IP Route Detai
4 2014-12-08 16:42:27,454 INFO question_progress: Results 0% (Get Computer Name and IP Route Detai
5 2014-12-08 16:42:32,483 INFO question_progress: Results 0% (Get Computer Name and IP Route Detai
6 2014-12-08 16:42:37,508 INFO question_progress: Results 0% (Get Computer Name and IP Route Detai
7 2014-12-08 16:42:42,537 INFO question_progress: Results 0% (Get Computer Name and IP Route Detai
8 2014-12-08 16:42:47,566 INFO question_progress: Results 0% (Get Computer Name and IP Route Detai
9 2014-12-08 16:42:52,593 INFO question_progress: Results 0% (Get Computer Name and IP Route Detai
10 2014-12-08 16:42:57,620 INFO question_progress: Results 33% (Get Computer Name and IP Route Deta
11 2014-12-08 16:43:02,649 INFO question_progress: Results 83% (Get Computer Name and IP Route Deta
12 2014-12-08 16:43:07,674 INFO question_progress: Results 100% (Get Computer Name and IP Route Det
13
14 print the export_str returned from export_obj():
15 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
16 2014-12-08 16:41:42,048 INFO question_progress: Results 0% (Get Computer Name and IP Route Detai
17 2014-12-08 16:41:47,073 INFO question_progress: Results 0% (Get Computer Name and IP Route Detai
18 2014-12-08 16:41:52,100 INFO question_progress: Results 0% (Get Computer Name and IP Route Detai
19 2014-12-08 16:41:57,128 INFO question_progress: Results 0% (Get Computer Name and IP Route Detai
20 2014-12-08 16:42:02,151 INFO question_progress: Results 0% (Get Computer Name and IP Route Detai
21 2014-12-08 16:42:07,176 INFO question_progress: Results 0% (Get Computer Name and IP Route Detai
22 2014-12-08 16:42:12,200 INFO question_progress: Results 33% (Get Computer Name and IP Route Deta
23 2014-12-08 16:42:17,226 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: 6.2.314.3258
27 2014-12-08 16:39:41,204 INFO question_progress: Results 0% (Get Computer Name and IP Route Detai
28 2014-12-08 16:39:46,233 INFO question_progress: Results 0% (Get Computer Name and IP Route Detai
29 2014-12-08 16:39:51,260 INFO question_progress: Results 0% (Get Computer Name and IP Route Detai
30 ..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  # Path to lib directory which contains pytan package
2  PYTAN_LIB_PATH = '../lib'
3
4  # connection info for Tanium Server
5  USERNAME = "Tanium User"
6  PASSWORD = "T@n!um"
7  HOST = "172.16.31.128"
8  PORT = "444"
9
10 # Logging controls
11 LOGLEVEL = 2
12 DEBUGFORMAT = False
13
14 import sys, tempfile
15 sys.path.append(PYTAN_LIB_PATH)
16
17 import pytan
18 handler = pytan.Handler(
19     username=USERNAME,
20     password=PASSWORD,
21     host=HOST,
22     port=PORT,
23     loglevel=LOGLEVEL,
24     debugformat=DEBUGFORMAT,
25 )
26
27 print handler
28
29 # setup the export_obj kwargs for later
30 export_kwargs = {}
31 export_kwargs["export_format"] = u'csv'
32 export_kwargs["header_add_sensor"] = True
33
34 # ask the question that will provide the resultset that we want to use
35 ask_kwargs = {
36     'qtype': 'manual_human',
37     'sensors': [
38         "Computer Name", "IP Route Details", "IP Address",
39         'Folder Name Search with RegEx Match{dirname=Program Files,regex=.*Shared.*}',
40     ],
41 }
42 response = handler.ask(**ask_kwargs)
43
44 # export the object to a string
45 # (we could just as easily export to a file using export_to_report_file)
46 export_kwargs['obj'] = response['question_results']
47 export_str = handler.export_obj(**export_kwargs)
48
49
50 print ""
51 print "print the export_str returned from export_obj():"
52 if len(out.splitlines()) > 15:
53     out = out.splitlines()[0:15]
54     out.append('..trimmed for brevity..')
55     out = '\n'.join(out)
56
57 print out

```

## Output from Python Code

```

1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
2 2014-12-08 16:43:07,859 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detai
3 2014-12-08 16:43:12,890 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detai
4 2014-12-08 16:43:17,917 INFO      question_progress: Results 67% (Get Computer Name and IP Route Deta
5 2014-12-08 16:43:22,946 INFO      question_progress: Results 67% (Get Computer Name and IP Route Deta
6 2014-12-08 16:43:27,974 INFO      question_progress: Results 83% (Get Computer Name and IP Route Deta
7 2014-12-08 16:43:33,001 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: 6.2.314.3258
11 2014-12-08 16:42:17,402 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detai
12 2014-12-08 16:42:22,428 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detai
13 2014-12-08 16:42:27,454 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detai
14 2014-12-08 16:42:32,483 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detai
15 2014-12-08 16:42:37,508 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detai
16 2014-12-08 16:42:42,537 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detai
17 2014-12-08 16:42:47,566 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detai
18 2014-12-08 16:42:52,593 INFO      question_progress: Results 0% (Get Computer Name and IP Route Detai
19 2014-12-08 16:42:57,620 INFO      question_progress: Results 33% (Get Computer Name and IP Route Deta
20 2014-12-08 16:43:02,649 INFO      question_progress: Results 83% (Get Computer Name and IP Route Deta
21 2014-12-08 16:43:07,674 INFO      question_progress: Results 100% (Get Computer Name and IP Route Deta
22
23 print the export_str returned from export_obj():
24 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
25 ..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 # Path to lib directory which contains pytan package
2 PYTAN_LIB_PATH = '../lib'
3
4 # connection info for Tanium Server
5 USERNAME = "Tanium User"
6 PASSWORD = "T@n!um"
7 HOST = "172.16.31.128"
8 PORT = "444"
9
10 # Logging conrols
11 LOGLEVEL = 2
12 DEBUGFORMAT = False
13
14 import sys, tempfile
15 sys.path.append(PYTAN_LIB_PATH)
16
17 import pytan
18 handler = pytan.Handler(
19     username=USERNAME,
20     password=PASSWORD,

```



```

21     host=HOST,
22     port=PORT,
23     loglevel=LOGLEVEL,
24     debugformat=DEBUGFORMAT,
25 )
26
27 print handler
28
29 # setup the export_obj kwargs for later
30 export_kwargs = {}
31 export_kwargs["export_format"] = u'csv'
32 export_kwargs["header_sort"] = [[]]
33
34 # ask the question that will provide the resultset that we want to use
35 ask_kwargs = {
36     'qtype': 'manual_human',
37     'sensors': [
38         "Computer Name"
39     ],
40 }
41 response = handler.ask(**ask_kwargs)
42 export_kwargs['obj'] = response['question_results']
43
44 # export the object to a string
45 # this should throw an exception: pytan.utils.HandlerError
46 import traceback
47
48 try:
49     handler.export_obj(**export_kwargs)
50 except Exception as e:
51     traceback.print_exc(file=sys.stdout)

```

### Output from Python Code

```

1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
2 2014-12-08 16:43:33,188 INFO      question_progress: Results 0% (Get Computer Name from all machines)
3 2014-12-08 16:43:38,204 INFO      question_progress: Results 67% (Get Computer Name from all machines)
4 2014-12-08 16:43:43,225 INFO      question_progress: Results 100% (Get Computer Name from all machines)
5 Traceback (most recent call last):
6   File "<string>", line 49, in <module>
7   File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 1402, in export_obj
8     utils.check_dictkey(**check_args)
9   File "/Users/jolsen/gh/pytan/lib/pytan/utils.py", line 2516, in check_dictkey
10     raise HandlerError(err(key, valid_list_types, list_types))
11 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 # Path to lib directory which contains pytan package
2 PYTAN_LIB_PATH = '../lib'

```

```
3
4 # connection info for Tanium Server
5 USERNAME = "Tanium User"
6 PASSWORD = "T@n!um"
7 HOST = "172.16.31.128"
8 PORT = "444"
9
10 # Logging conrols
11 LOGLEVEL = 2
12 DEBUGFORMAT = False
13
14 import sys, tempfile
15 sys.path.append(PYTAN_LIB_PATH)
16
17 import pytan
18 handler = pytan.Handler(
19     username=USERNAME,
20     password=PASSWORD,
21     host=HOST,
22     port=PORT,
23     loglevel=LOGLEVEL,
24     debugformat=DEBUGFORMAT,
25 )
26
27 print handler
28
29 # setup the export_obj kwargs for later
30 export_kwargs = {}
31 export_kwargs["export_format"] = u'csv'
32 export_kwargs["header_sort"] = u'bad'
33
34 # ask the question that will provide the resultset that we want to use
35 ask_kwargs = {
36     'qtype': 'manual_human',
37     'sensors': [
38         "Computer Name"
39     ],
40 }
41 response = handler.ask(**ask_kwargs)
42 export_kwargs['obj'] = response['question_results']
43
44 # export the object to a string
45 # this should throw an exception: pytan.utils.HandlerError
46 import traceback
47
48 try:
49     handler.export_obj(**export_kwargs)
50 except Exception as e:
51     traceback.print_exc(file=sys.stdout)
```

### Output from Python Code

```
1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
2 2014-12-08 16:43:43,303 INFO      question_progress: Results 0% (Get Computer Name from all machines)
3 2014-12-08 16:43:48,320 INFO      question_progress: Results 67% (Get Computer Name from all machines)
4 2014-12-08 16:43:53,334 INFO      question_progress: Results 100% (Get Computer Name from all machine
5 Traceback (most recent call last):
```

```

6   File "<string>", line 49, in <module>
7   File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 1402, in export_obj
8       utils.check_dictkey(**check_args)
9   File "/Users/jolsen/gh/pytan/lib/pytan/utils.py", line 2509, 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  # Path to lib directory which contains pytan package
2  PYTAN_LIB_PATH = '../lib'
3
4  # connection info for Tanium Server
5  USERNAME = "Tanium User"
6  PASSWORD = "T@n!um"
7  HOST = "172.16.31.128"
8  PORT = "444"
9
10 # Logging conrols
11 LOGLEVEL = 2
12 DEBUGFORMAT = False
13
14 import sys, tempfile
15 sys.path.append(PYTAN_LIB_PATH)
16
17 import pytan
18 handler = pytan.Handler(
19     username=USERNAME,
20     password=PASSWORD,
21     host=HOST,
22     port=PORT,
23     loglevel=LOGLEVEL,
24     debugformat=DEBUGFORMAT,
25 )
26
27 print handler
28
29 # setup the export_obj kwargs for later
30 export_kwargs = {}
31 export_kwargs["export_format"] = u'csv'
32 export_kwargs["expand_grouped_columns"] = u'bad'
33
34 # ask the question that will provide the resultset that we want to use
35 ask_kwargs = {
36     'qtype': 'manual_human',
37     'sensors': [
38         "Computer Name"
39     ],
40 }
41 response = handler.ask(**ask_kwargs)
42 export_kwargs['obj'] = response['question_results']

```

```
43
44 # export the object to a string
45 # this should throw an exception: pytan.utils.HandlerError
46 import traceback
47
48 try:
49     handler.export_obj(**export_kwargs)
50 except Exception as e:
51     traceback.print_exc(file=sys.stdout)
```

### Output from Python Code

```
1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
2 2014-12-08 16:43:53,414 INFO      question_progress: Results 0% (Get Computer Name from all machines)
3 2014-12-08 16:43:58,431 INFO      question_progress: Results 0% (Get Computer Name from all machines)
4 2014-12-08 16:44:03,451 INFO      question_progress: Results 100% (Get Computer Name from all machines)
5 Traceback (most recent call last):
6   File "<string>", line 49, in <module>
7     File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 1402, in export_obj
8       utils.check_dictkey(**check_args)
9     File "/Users/jolsen/gh/pytan/lib/pytan/utils.py", line 2509, 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 # Path to lib directory which contains pytan package
2 PYTAN_LIB_PATH = '../lib'
3
4 # connection info for Tanium Server
5 USERNAME = "Tanium User"
6 PASSWORD = "T@n!um"
7 HOST = "172.16.31.128"
8 PORT = "444"
9
10 # Logging conrols
11 LOGLEVEL = 2
12 DEBUGFORMAT = False
13
14 import sys, tempfile
15 sys.path.append(PYTAN_LIB_PATH)
16
17 import pytan
18 handler = pytan.Handler(
19     username=USERNAME,
20     password=PASSWORD,
21     host=HOST,
22     port=PORT,
23     loglevel=LOGLEVEL,
24     debugformat=DEBUGFORMAT,
```

```

25 )
26
27 print handler
28
29 # setup the export_obj kwargs for later
30 export_kwargs = {}
31 export_kwargs["export_format"] = u'csv'
32 export_kwargs["sensors"] = [[]]
33 export_kwargs["header_add_sensor"] = True
34
35 # ask the question that will provide the resultset that we want to use
36 ask_kwargs = {
37     'qtype': 'manual_human',
38     'sensors': [
39         "Computer Name"
40     ],
41 }
42 response = handler.ask(**ask_kwargs)
43 export_kwargs['obj'] = response['question_results']
44
45 # export the object to a string
46 # this should throw an exception: pytan.utils.HandlerError
47 import traceback
48
49 try:
50     handler.export_obj(**export_kwargs)
51 except Exception as e:
52     traceback.print_exc(file=sys.stdout)

```

### Output from Python Code

```

1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
2 2014-12-08 16:44:03,549 INFO      question_progress: Results 0% (Get Computer Name from all machines)
3 2014-12-08 16:44:08,565 INFO      question_progress: Results 50% (Get Computer Name from all machines)
4 2014-12-08 16:44:13,582 INFO      question_progress: Results 83% (Get Computer Name from all machines)
5 2014-12-08 16:44:18,605 INFO      question_progress: Results 100% (Get Computer Name from all machines)
6 Traceback (most recent call last):
7   File "<string>", line 50, in <module>
8   File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 1402, in export_obj
9     utils.check_dictkey(**check_args)
10  File "/Users/jolsen/gh/pytan/lib/pytan/utils.py", line 2516, in check_dictkey
11     raise HandlerError(err(key, valid_list_types, list_types))
12 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 # Path to lib directory which contains pytan package
2 PYTAN_LIB_PATH = '../lib'
3
4 # connection info for Tanium Server

```

```
5  USERNAME = "Tanium User"
6  PASSWORD = "T@n!um"
7  HOST = "172.16.31.128"
8  PORT = "444"
9
10 # Logging conrols
11 LOGLEVEL = 2
12 DEBUGFORMAT = False
13
14 import sys, tempfile
15 sys.path.append(PYTAN_LIB_PATH)
16
17 import pytan
18 handler = pytan.Handler(
19     username=USERNAME,
20     password=PASSWORD,
21     host=HOST,
22     port=PORT,
23     loglevel=LOGLEVEL,
24     debugformat=DEBUGFORMAT,
25 )
26
27 print handler
28
29 # setup the export_obj kwargs for later
30 export_kwargs = {}
31 export_kwargs["export_format"] = u'bad'
32
33 # ask the question that will provide the resultset that we want to use
34 ask_kwargs = {
35     'qtype': 'manual_human',
36     'sensors': [
37         "Computer Name"
38     ],
39 }
40 response = handler.ask(**ask_kwargs)
41 export_kwargs['obj'] = response['question_results']
42
43 # export the object to a string
44 # this should throw an exception: pytan.utils.HandlerError
45 import traceback
46
47 try:
48     handler.export_obj(**export_kwargs)
49 except Exception as e:
50     traceback.print_exc(file=sys.stdout)
```

## Output from Python Code

```
1  Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
2  2014-12-08 16:44:18,688 INFO      question_progress: Results 0% (Get Computer Name from all machines)
3  2014-12-08 16:44:23,702 INFO      question_progress: Results 83% (Get Computer Name from all machines)
4  2014-12-08 16:44:28,719 INFO      question_progress: Results 100% (Get Computer Name from all machines)
5  Traceback (most recent call last):
6    File "<string>", line 48, in <module>
7    File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 1396, in export_obj
8    raise HandlerError(err)
```

```
9 | 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 | # Path to lib directory which contains pytan package
2 | PYTAN_LIB_PATH = '../lib'
3 |
4 | # connection info for Tanium Server
5 | USERNAME = "Tanium User"
6 | PASSWORD = "T@n!um"
7 | HOST = "172.16.31.128"
8 | PORT = "444"
9 |
10 | # Logging conrols
11 | LOGLEVEL = 2
12 | DEBUGFORMAT = False
13 |
14 | import sys, tempfile
15 | sys.path.append(PYTAN_LIB_PATH)
16 |
17 | import pytan
18 | handler = pytan.Handler(
19 |     username=USERNAME,
20 |     password=PASSWORD,
21 |     host=HOST,
22 |     port=PORT,
23 |     loglevel=LOGLEVEL,
24 |     debugformat=DEBUGFORMAT,
25 | )
26 |
27 | print handler
28 |
29 | # setup the export_obj kwargs for later
30 | export_kwargs = {}
31 | export_kwargs["export_format"] = u'csv'
32 |
33 | # get the objects that will provide the basetype that we want to use
34 | get_kwargs = {
35 |     'name': [
36 |         "Computer Name", "IP Route Details", "IP Address",
37 |         'Folder Name Search with RegEx Match',
38 |     ],
39 |     'objtype': 'sensor',
40 | }
41 | response = handler.get(**get_kwargs)
42 |
43 | # export the object to a string
44 | # (we could just as easily export to a file using export_to_report_file)
45 | export_kwargs['obj'] = response
```

```
46 export_str = handler.export_obj(**export_kwargs)
47
48
49 print ""
50 print "print the export_str returned from export_obj():"
51
52 out = export_str
53 if len(out.splitlines()) > 15:
54     out = out.splitlines()[0:15]
55     out.append('..trimmed for brevity..')
56     out = '\n'.join(out)
57
58 print out
```

### Output from Python Code

```
1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
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,2014-12-08T19:20:42,|,"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 # Path to lib directory which contains pytan package
2 PYTAN_LIB_PATH = '../lib'
3
4 # connection info for Tanium Server
5 USERNAME = "Tanium User"
6 PASSWORD = "T@n!um"
7 HOST = "172.16.31.128"
8 PORT = "444"
9
10 # Logging conrols
11 LOGLEVEL = 2
12 DEBUGFORMAT = False
```



```

13
14 import sys, tempfile
15 sys.path.append(PYTAN_LIB_PATH)
16
17 import pytan
18 handler = pytan.Handler(
19     username=USERNAME,
20     password=PASSWORD,
21     host=HOST,
22     port=PORT,
23     loglevel=LOGLEVEL,
24     debugformat=DEBUGFORMAT,
25 )
26
27 print handler
28
29 # setup the export_obj kwargs for later
30 export_kwargs = {}
31 export_kwargs["export_format"] = u'json'
32 export_kwargs["include_type"] = False
33
34 # get the objects that will provide the basetype that we want to use
35 get_kwargs = {
36     'name': [
37         "Computer Name", "IP Route Details", "IP Address",
38         'Folder Name Search with RegEx Match',
39     ],
40     'objtype': 'sensor',
41 }
42 response = handler.get(**get_kwargs)
43
44 # export the object to a string
45 # (we could just as easily export to a file using export_to_report_file)
46 export_kwargs['obj'] = response
47 export_str = handler.export_obj(**export_kwargs)
48
49
50 print ""
51 print "print the export_str returned from export_obj():"
52
53 out = export_str
54 if len(out.splitlines()) > 15:
55     out = out.splitlines()[0:15]
56     out.append('..trimmed for brevity..')
57     out = '\n'.join(out)
58
59 print out

```

### Output from Python Code

```

1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
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-1company.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  # Path to lib directory which contains pytan package
2  PYTAN_LIB_PATH = '../lib'
3
4  # connection info for Tanium Server
5  USERNAME = "Tanium User"
6  PASSWORD = "T@n!um"
7  HOST = "172.16.31.128"
8  PORT = "444"
9
10 # Logging conrols
11 LOGLEVEL = 2
12 DEBUGFORMAT = False
13
14 import sys, tempfile
15 sys.path.append(PYTAN_LIB_PATH)
16
17 import pytan
18 handler = pytan.Handler(
19     username=USERNAME,
20     password=PASSWORD,
21     host=HOST,
22     port=PORT,
23     loglevel=LOGLEVEL,
24     debugformat=DEBUGFORMAT,
25 )
26
27 print handler
28
29 # setup the export_obj kwargs for later
30 export_kwargs = {}
31 export_kwargs["export_format"] = u'json'
32 export_kwargs["explode_json_string_values"] = False
33
34 # get the objects that will provide the basetype that we want to use
35 get_kwargs = {
36     'name': [
```

```

37         "Computer Name", "IP Route Details", "IP Address",
38         'Folder Name Search with RegEx Match',
39     ],
40     'objtype': 'sensor',
41 }
42 response = handler.get(**get_kwargs)
43
44 # export the object to a string
45 # (we could just as easily export to a file using export_to_report_file)
46 export_kwargs['obj'] = response
47 export_str = handler.export_obj(**export_kwargs)
48
49
50 print ""
51 print "print the export_str returned from export_obj():"
52
53 out = export_str
54 if len(out.splitlines()) > 15:
55     out = out.splitlines()[0:15]
56     out.append('..trimmed for brevity..')
57     out = '\n'.join(out)
58
59 print out

```

### Output from Python Code

```

1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
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 # Path to lib directory which contains pytan package
2 PYTAN_LIB_PATH = '../lib'

```

```
3
4 # connection info for Tanium Server
5 USERNAME = "Tanium User"
6 PASSWORD = "T@n!um"
7 HOST = "172.16.31.128"
8 PORT = "444"
9
10 # Logging conrols
11 LOGLEVEL = 2
12 DEBUGFORMAT = False
13
14 import sys, tempfile
15 sys.path.append(PYTAN_LIB_PATH)
16
17 import pytan
18 handler = pytan.Handler(
19     username=USERNAME,
20     password=PASSWORD,
21     host=HOST,
22     port=PORT,
23     loglevel=LOGLEVEL,
24     debugformat=DEBUGFORMAT,
25 )
26
27 print handler
28
29 # setup the export_obj kwargs for later
30 export_kwargs = {}
31 export_kwargs["export_format"] = u'json'
32 export_kwargs["explode_json_string_values"] = True
33
34 # get the objects that will provide the basetype that we want to use
35 get_kwargs = {
36     'name': [
37         "Computer Name", "IP Route Details", "IP Address",
38         "Folder Name Search with RegEx Match",
39     ],
40     'objtype': 'sensor',
41 }
42 response = handler.get(**get_kwargs)
43
44 # export the object to a string
45 # (we could just as easily export to a file using export_to_report_file)
46 export_kwargs['obj'] = response
47 export_str = handler.export_obj(**export_kwargs)
48
49
50 print ""
51 print "print the export_str returned from export_obj():"
52
53 out = export_str
54 if len(out.splitlines()) > 15:
55     out = out.splitlines()[0:15]
56     out.append('..trimmed for brevity..')
57     out = '\n'.join(out)
58
59 print out
```

## Output from Python Code

```

1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
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  # Path to lib directory which contains pytan package
2  PYTAN_LIB_PATH = '../lib'
3
4  # connection info for Tanium Server
5  USERNAME = "Tanium User"
6  PASSWORD = "T@n!um"
7  HOST = "172.16.31.128"
8  PORT = "444"
9
10 # Logging conrols
11 LOGLEVEL = 2
12 DEBUGFORMAT = False
13
14 import sys, tempfile
15 sys.path.append(PYTAN_LIB_PATH)
16
17 import pytan
18 handler = pytan.Handler(
19     username=USERNAME,
20     password=PASSWORD,
21     host=HOST,
22     port=PORT,
23     loglevel=LOGLEVEL,
24     debugformat=DEBUGFORMAT,
25 )
26
27 print handler
28

```

```
29 # setup the export_obj kwargs for later
30 export_kwargs = {}
31 export_kwargs["export_format"] = u'xml'
32
33 # get the objects that will provide the basetype that we want to use
34 get_kwargs = {
35     'name': [
36         "Computer Name", "IP Route Details", "IP Address",
37         'Folder Name Search with RegEx Match',
38     ],
39     'objtype': 'sensor',
40 }
41 response = handler.get(**get_kwargs)
42
43 # export the object to a string
44 # (we could just as easily export to a file using export_to_report_file)
45 export_kwargs['obj'] = response
46 export_str = handler.export_obj(**export_kwargs)
47
48
49 print ""
50 print "print the export_str returned from export_obj():"
51
52 out = export_str
53 if len(out.splitlines()) > 15:
54     out = out.splitlines()[0:15]
55     out.append('..trimmed for brevity..')
56     out = '\n'.join(out)
57
58 print out
```

### Output from Python Code

```
1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
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 false

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

### Example Python Code

```

1  # Path to lib directory which contains pytan package
2  PYTAN_LIB_PATH = '../lib'
3
4  # connection info for Tanium Server
5  USERNAME = "Tanium User"
6  PASSWORD = "T@n!um"
7  HOST = "172.16.31.128"
8  PORT = "444"
9
10 # Logging conrols
11 LOGLEVEL = 2
12 DEBUGFORMAT = False
13
14 import sys, tempfile
15 sys.path.append(PYTAN_LIB_PATH)
16
17 import pytan
18 handler = pytan.Handler(
19     username=USERNAME,
20     password=PASSWORD,
21     host=HOST,
22     port=PORT,
23     loglevel=LOGLEVEL,
24     debugformat=DEBUGFORMAT,
25 )
26
27 print handler
28
29 # setup the export_obj kwargs for later
30 export_kwargs = {}
31 export_kwargs["export_format"] = u'xml'
32 export_kwargs["minimal"] = False
33
34 # get the objects that will provide the basetype that we want to use
35 get_kwargs = {
36     'name': [
37         "Computer Name", "IP Route Details", "IP Address",
38         'Folder Name Search with RegEx Match',
39     ],
40     'objtype': 'sensor',
41 }
42 response = handler.get(**get_kwargs)
43
44 # export the object to a string
45 # (we could just as easily export to a file using export_to_report_file)
46 export_kwargs['obj'] = response
47 export_str = handler.export_obj(**export_kwargs)
48
49
50 print ""
51 print "print the export_str returned from export_obj():"
52

```

```
53 out = export_str
54 if len(out.splitlines()) > 15:
55     out = out.splitlines()[0:15]
56     out.append('..trimmed for brevity..')
57     out = '\n'.join(out)
58
59 print out
```

### Output from Python Code

```
1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
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 # Path to lib directory which contains pytan package
2 PYTAN_LIB_PATH = '../lib'
3
4 # connection info for Tanium Server
5 USERNAME = "Tanium User"
6 PASSWORD = "T@n!um"
7 HOST = "172.16.31.128"
8 PORT = "444"
9
10 # Logging conrols
11 LOGLEVEL = 2
12 DEBUGFORMAT = False
13
14 import sys, tempfile
15 sys.path.append(PYTAN_LIB_PATH)
16
17 import pytan
18 handler = pytan.Handler(
```



```

19     username=USERNAME,
20     password=PASSWORD,
21     host=HOST,
22     port=PORT,
23     loglevel=LOGLEVEL,
24     debugformat=DEBUGFORMAT,
25 )
26
27 print handler
28
29 # setup the export_obj kwargs for later
30 export_kwargs = {}
31 export_kwargs["export_format"] = u'xml'
32 export_kwargs["minimal"] = True
33
34 # get the objects that will provide the basetype that we want to use
35 get_kwargs = {
36     'name': [
37         "Computer Name", "IP Route Details", "IP Address",
38         'Folder Name Search with RegEx Match',
39     ],
40     'objtype': 'sensor',
41 }
42 response = handler.get(**get_kwargs)
43
44 # export the object to a string
45 # (we could just as easily export to a file using export_to_report_file)
46 export_kwargs['obj'] = response
47 export_str = handler.export_obj(**export_kwargs)
48
49
50 print ""
51 print "print the export_str returned from export_obj():"
52
53 out = export_str
54 if len(out.splitlines()) > 15:
55     out = out.splitlines()[0:15]
56     out.append('..trimmed for brevity..')
57     out = '\n'.join(out)
58
59 print out

```

### Output from Python Code

```

1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
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;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  # Path to lib directory which contains pytan package
2  PYTAN_LIB_PATH = '../lib'
3
4  # connection info for Tanium Server
5  USERNAME = "Tanium User"
6  PASSWORD = "T@n!um"
7  HOST = "172.16.31.128"
8  PORT = "444"
9
10 # Logging conrols
11 LOGLEVEL = 2
12 DEBUGFORMAT = False
13
14 import sys, tempfile
15 sys.path.append(PYTAN_LIB_PATH)
16
17 import pytan
18 handler = pytan.Handler(
19     username=USERNAME,
20     password=PASSWORD,
21     host=HOST,
22     port=PORT,
23     loglevel=LOGLEVEL,
24     debugformat=DEBUGFORMAT,
25 )
26
27 print handler
28
29 # setup the export_obj kwargs for later
30 export_kwargs = {}
31 export_kwargs["export_format"] = u'csv'
32 export_kwargs["explode_json_string_values"] = False
33
34 # get the objects that will provide the basetype that we want to use
35 get_kwargs = {
36     'name': [
37         "Computer Name", "IP Route Details", "IP Address",
38         'Folder Name Search with RegEx Match',
39     ],
40     'objtype': 'sensor',
41 }
42 response = handler.get(**get_kwargs)
```

```

43
44 # export the object to a string
45 # (we could just as easily export to a file using export_to_report_file)
46 export_kwargs['obj'] = response
47 export_str = handler.export_obj(**export_kwargs)
48
49
50 print ""
51 print "print the export_str returned from export_obj():"
52
53 out = export_str
54 if len(out.splitlines()) > 15:
55     out = out.splitlines()[0:15]
56     out.append('..trimmed for brevity..')
57     out = '\n'.join(out)
58
59 print out

```

### Output from Python Code

```

1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
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,2014-12-08T19:20:42,|,"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 # Path to lib directory which contains pytan package
2 PYTAN_LIB_PATH = '../lib'
3
4 # connection info for Tanium Server
5 USERNAME = "Tanium User"
6 PASSWORD = "T@n!um"
7 HOST = "172.16.31.128"
8 PORT = "444"

```

```
9
10 # Logging controls
11 LOGLEVEL = 2
12 DEBUGFORMAT = False
13
14 import sys, tempfile
15 sys.path.append(PYTAN_LIB_PATH)
16
17 import pytan
18 handler = pytan.Handler(
19     username=USERNAME,
20     password=PASSWORD,
21     host=HOST,
22     port=PORT,
23     loglevel=LOGLEVEL,
24     debugformat=DEBUGFORMAT,
25 )
26
27 print handler
28
29 # setup the export_obj kwargs for later
30 export_kwargs = {}
31 export_kwargs["export_format"] = u'csv'
32 export_kwargs["explode_json_string_values"] = True
33
34 # get the objects that will provide the basetype that we want to use
35 get_kwargs = {
36     'name': [
37         "Computer Name", "IP Route Details", "IP Address",
38         'Folder Name Search with RegEx Match',
39     ],
40     'objtype': 'sensor',
41 }
42 response = handler.get(**get_kwargs)
43
44 # export the object to a string
45 # (we could just as easily export to a file using export_to_report_file)
46 export_kwargs['obj'] = response
47 export_str = handler.export_obj(**export_kwargs)
48
49
50 print ""
51 print "print the export_str returned from export_obj():"
52
53 out = export_str
54 if len(out.splitlines()) > 15:
55     out = out.splitlines()[0:15]
56     out.append('..trimmed for brevity..')
57     out = '\n'.join(out)
58
59 print out
```

### Output from Python Code

```
1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
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,2014-12-08T19:20:42,|,"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  # Path to lib directory which contains pytan package
2  PYTAN_LIB_PATH = '../lib'
3
4  # connection info for Tanium Server
5  USERNAME = "Tanium User"
6  PASSWORD = "T@n!um"
7  HOST = "172.16.31.128"
8  PORT = "444"
9
10 # Logging conrols
11 LOGLEVEL = 2
12 DEBUGFORMAT = False
13
14 import sys, tempfile
15 sys.path.append(PYTAN_LIB_PATH)
16
17 import pytan
18 handler = pytan.Handler(
19     username=USERNAME,
20     password=PASSWORD,
21     host=HOST,
22     port=PORT,
23     loglevel=LOGLEVEL,
24     debugformat=DEBUGFORMAT,
25 )
26
27 print handler
28
29 # setup the export_obj kwargs for later
30 export_kwargs = {}
31 export_kwargs["export_format"] = u'csv'
32 export_kwargs["header_sort"] = []

```

```
33
34 # get the objects that will provide the basetype that we want to use
35 get_kwargs = {
36     'name': [
37         "Computer Name", "IP Route Details", "IP Address",
38         'Folder Name Search with RegEx Match',
39     ],
40     'objtype': 'sensor',
41 }
42 response = handler.get(**get_kwargs)
43
44 # export the object to a string
45 # (we could just as easily export to a file using export_to_report_file)
46 export_kwargs['obj'] = response
47 export_str = handler.export_obj(**export_kwargs)
48
49
50 print ""
51 print "print the export_str returned from export_obj():"
52
53 out = export_str
54 if len(out.splitlines()) > 15:
55     out = out.splitlines()[0:15]
56     out.append('..trimmed for brevity..')
57     out = '\n'.join(out)
58
59 print out
```

### Output from Python Code

```
1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
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,2014-12-08T19:20:42,|,"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 true

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

**Example Python Code**

```

1  # Path to lib directory which contains pytan package
2  PYTAN_LIB_PATH = '../lib'
3
4  # connection info for Tanium Server
5  USERNAME = "Tanium User"
6  PASSWORD = "T@n!um"
7  HOST = "172.16.31.128"
8  PORT = "444"
9
10 # Logging conrols
11 LOGLEVEL = 2
12 DEBUGFORMAT = False
13
14 import sys, tempfile
15 sys.path.append(PYTAN_LIB_PATH)
16
17 import pytan
18 handler = pytan.Handler(
19     username=USERNAME,
20     password=PASSWORD,
21     host=HOST,
22     port=PORT,
23     loglevel=LOGLEVEL,
24     debugformat=DEBUGFORMAT,
25 )
26
27 print handler
28
29 # setup the export_obj kwargs for later
30 export_kwargs = {}
31 export_kwargs["export_format"] = u'csv'
32 export_kwargs["header_sort"] = True
33
34 # get the objects that will provide the basetype that we want to use
35 get_kwargs = {
36     'name': [
37         "Computer Name", "IP Route Details", "IP Address",
38         "Folder Name Search with RegEx Match",
39     ],
40     'objtype': 'sensor',
41 }
42 response = handler.get(**get_kwargs)
43
44 # export the object to a string
45 # (we could just as easily export to a file using export_to_report_file)
46 export_kwargs['obj'] = response
47 export_str = handler.export_obj(**export_kwargs)
48
49
50 print ""
51 print "print the export_str returned from export_obj():"
52
53 out = export_str
54 if len(out.splitlines()) > 15:
55     out = out.splitlines()[0:15]
56     out.append('..trimmed for brevity..')
57     out = '\n'.join(out)

```

```
58
59 print out
```

### Output from Python Code

```
1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
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,2014-12-08T19:20:42,|,"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 list

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

### Example Python Code

```
1 # Path to lib directory which contains pytan package
2 PYTAN_LIB_PATH = '../lib'
3
4 # connection info for Tanium Server
5 USERNAME = "Tanium User"
6 PASSWORD = "T@n!um"
7 HOST = "172.16.31.128"
8 PORT = "444"
9
10 # Logging conrols
11 LOGLEVEL = 2
12 DEBUGFORMAT = False
13
14 import sys, tempfile
15 sys.path.append(PYTAN_LIB_PATH)
16
17 import pytan
18 handler = pytan.Handler(
19     username=USERNAME,
20     password=PASSWORD,
21     host=HOST,
22     port=PORT,
23     loglevel=LOGLEVEL,
```



```

24         debugformat=DEBUGFORMAT,
25     )
26
27     print handler
28
29     # setup the export_obj kwargs for later
30     export_kwargs = {}
31     export_kwargs["export_format"] = u'csv'
32     export_kwargs["header_sort"] = [u'name', u'description']
33
34     # get the objects that will provide the basetype that we want to use
35     get_kwargs = {
36         'name': [
37             "Computer Name", "IP Route Details", "IP Address",
38             'Folder Name Search with RegEx Match',
39         ],
40         'objtype': 'sensor',
41     }
42     response = handler.get(**get_kwargs)
43
44     # export the object to a string
45     # (we could just as easily export to a file using export_to_report_file)
46     export_kwargs['obj'] = response
47     export_str = handler.export_obj(**export_kwargs)
48
49
50     print ""
51     print "print the export_str returned from export_obj():"
52
53     out = export_str
54     if len(out.splitlines()) > 15:
55         out = out.splitlines()[0:15]
56         out.append('..trimmed for brevity..')
57         out = '\n'.join(out)
58
59     print out

```

### Output from Python Code

```

1  Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
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,2014-12-08T19:20:42,,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 | # Path to lib directory which contains pytan package
2 | PYTAN_LIB_PATH = '../lib'
3 |
4 | # connection info for Tanium Server
5 | USERNAME = "Tanium User"
6 | PASSWORD = "T@n!um"
7 | HOST = "172.16.31.128"
8 | PORT = "444"
9 |
10 | # Logging conrols
11 | LOGLEVEL = 2
12 | DEBUGFORMAT = False
13 |
14 | import sys, tempfile
15 | sys.path.append(PYTAN_LIB_PATH)
16 |
17 | import pytan
18 | handler = pytan.Handler(
19 |     username=USERNAME,
20 |     password=PASSWORD,
21 |     host=HOST,
22 |     port=PORT,
23 |     loglevel=LOGLEVEL,
24 |     debugformat=DEBUGFORMAT,
25 | )
26 |
27 | print handler
28 |
29 | # setup the export_obj kwargs for later
30 | export_kwargs = {}
31 | export_kwargs["export_format"] = u'json'
32 |
33 | # get the objects that will provide the basetype that we want to use
34 | get_kwargs = {
35 |     'name': [
36 |         "Computer Name", "IP Route Details", "IP Address",
37 |         'Folder Name Search with RegEx Match',
38 |     ],
39 |     'objtype': 'sensor',
40 | }
41 | response = handler.get(**get_kwargs)
42 |
43 | # export the object to a string
44 | # (we could just as easily export to a file using export_to_report_file)
45 | export_kwargs['obj'] = response
46 | export_str = handler.export_obj(**export_kwargs)
47 |
```

```

48
49 print ""
50 print "print the export_str returned from export_obj():"
51
52 out = export_str
53 if len(out.splitlines()) > 15:
54     out = out.splitlines()[0:15]
55     out.append('..trimmed for brevity..')
56     out = '\n'.join(out)
57
58 print out

```

### Output from Python Code

```

1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
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 # Path to lib directory which contains pytan package
2 PYTAN_LIB_PATH = '../lib'
3
4 # connection info for Tanium Server
5 USERNAME = "Tanium User"
6 PASSWORD = "T@n!um"
7 HOST = "172.16.31.128"
8 PORT = "444"
9
10 # Logging conrols
11 LOGLEVEL = 2
12 DEBUGFORMAT = False
13
14 import sys, tempfile

```

```
15 sys.path.append(PYTAN_LIB_PATH)
16
17 import pytan
18 handler = pytan.Handler(
19     username=USERNAME,
20     password=PASSWORD,
21     host=HOST,
22     port=PORT,
23     loglevel=LOGLEVEL,
24     debugformat=DEBUGFORMAT,
25 )
26
27 print handler
28
29 # setup the export_obj kwargs for later
30 export_kwargs = {}
31 export_kwargs["export_format"] = u'json'
32 export_kwargs["include_type"] = True
33
34 # get the objects that will provide the basetype that we want to use
35 get_kwargs = {
36     'name': [
37         "Computer Name", "IP Route Details", "IP Address",
38         'Folder Name Search with RegEx Match',
39     ],
40     'objtype': 'sensor',
41 }
42 response = handler.get(**get_kwargs)
43
44 # export the object to a string
45 # (we could just as easily export to a file using export_to_report_file)
46 export_kwargs['obj'] = response
47 export_str = handler.export_obj(**export_kwargs)
48
49
50 print ""
51 print "print the export_str returned from export_obj():"
52
53 out = export_str
54 if len(out.splitlines()) > 15:
55     out = out.splitlines()[0:15]
56     out.append('..trimmed for brevity..')
57     out = '\n'.join(out)
58
59 print out
```

### Output from Python Code

```
1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
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-1company.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  # Path to lib directory which contains pytan package
2  PYTAN_LIB_PATH = '../lib'
3
4  # connection info for Tanium Server
5  USERNAME = "Tanium User"
6  PASSWORD = "T@n!um"
7  HOST = "172.16.31.128"
8  PORT = "444"
9
10 # Logging conrols
11 LOGLEVEL = 2
12 DEBUGFORMAT = False
13
14 import sys, tempfile
15 sys.path.append(PYTAN_LIB_PATH)
16
17 import pytan
18 handler = pytan.Handler(
19     username=USERNAME,
20     password=PASSWORD,
21     host=HOST,
22     port=PORT,
23     loglevel=LOGLEVEL,
24     debugformat=DEBUGFORMAT,
25 )
26
27 print handler
28
29 # setup the export_obj kwargs for later
30 export_kwargs = {}
31 export_kwargs["export_format"] = u'csv'
32 export_kwargs["explode_json_string_values"] = u'bad'
33
34 # get the objects that will provide the basetype that we want to use
35 get_kwargs = {
36     'name': [

```

```
37         "Computer Name", "IP Route Details", "IP Address",
38         'Folder Name Search with RegEx Match',
39     ],
40     'objtype': 'sensor',
41 }
42 response = handler.get(**get_kwargs)
43 export_kwargs['obj'] = response
44
45 # export the object to a string
46 # this should throw an exception: pytan.utils.HandlerError
47 import traceback
48
49 try:
50     handler.export_obj(**export_kwargs)
51 except Exception as e:
52     traceback.print_exc(file=sys.stdout)
```

### Output from Python Code

```
1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
2 Traceback (most recent call last):
3   File "<string>", line 50, in <module>
4   File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 1402, in export_obj
5     utils.check_dictkey(**check_args)
6   File "/Users/jolsen/gh/pytan/lib/pytan/utils.py", line 2509, 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 # Path to lib directory which contains pytan package
2 PYTAN_LIB_PATH = '../lib'
3
4 # connection info for Tanium Server
5 USERNAME = "Tanium User"
6 PASSWORD = "T@n!um"
7 HOST = "172.16.31.128"
8 PORT = "444"
9
10 # Logging conrols
11 LOGLEVEL = 2
12 DEBUGFORMAT = False
13
14 import sys, tempfile
15 sys.path.append(PYTAN_LIB_PATH)
16
17 import pytan
18 handler = pytan.Handler(
19     username=USERNAME,
20     password=PASSWORD,
```

```

21     host=HOST,
22     port=PORT,
23     loglevel=LOGLEVEL,
24     debugformat=DEBUGFORMAT,
25 )
26
27 print handler
28
29 # setup the export_obj kwargs for later
30 export_kwargs = {}
31 export_kwargs["export_format"] = u'csv'
32 export_kwargs["header_sort"] = [[]]
33
34 # get the objects that will provide the basetype that we want to use
35 get_kwargs = {
36     'name': [
37         "Computer Name", "IP Route Details", "IP Address",
38         'Folder Name Search with RegEx Match',
39     ],
40     'objtype': 'sensor',
41 }
42 response = handler.get(**get_kwargs)
43 export_kwargs['obj'] = response
44
45 # export the object to a string
46 # this should throw an exception: pytan.utils.HandlerError
47 import traceback
48
49 try:
50     handler.export_obj(**export_kwargs)
51 except Exception as e:
52     traceback.print_exc(file=sys.stdout)

```

## Output from Python Code

```

1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
2 Traceback (most recent call last):
3   File "<string>", line 50, in <module>
4   File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 1402, in export_obj
5     utils.check_dictkey(**check_args)
6   File "/Users/jolsen/gh/pytan/lib/pytan/utils.py", line 2516, 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 # Path to lib directory which contains pytan package
2 PYTAN_LIB_PATH = '../lib'
3
4 # connection info for Tanium Server

```

```
5  USERNAME = "Tanium User"
6  PASSWORD = "T@n!um"
7  HOST = "172.16.31.128"
8  PORT = "444"
9
10 # Logging conrols
11 LOGLEVEL = 2
12 DEBUGFORMAT = False
13
14 import sys, tempfile
15 sys.path.append(PYTAN_LIB_PATH)
16
17 import pytan
18 handler = pytan.Handler(
19     username=USERNAME,
20     password=PASSWORD,
21     host=HOST,
22     port=PORT,
23     loglevel=LOGLEVEL,
24     debugformat=DEBUGFORMAT,
25 )
26
27 print handler
28
29 # setup the export_obj kwargs for later
30 export_kwargs = {}
31 export_kwargs["export_format"] = u'csv'
32 export_kwargs["header_sort"] = u'bad'
33
34 # get the objects that will provide the basetype that we want to use
35 get_kwargs = {
36     'name': [
37         "Computer Name", "IP Route Details", "IP Address",
38         'Folder Name Search with RegEx Match',
39     ],
40     'objtype': 'sensor',
41 }
42 response = handler.get(**get_kwargs)
43 export_kwargs['obj'] = response
44
45 # export the object to a string
46 # this should throw an exception: pytan.utils.HandlerError
47 import traceback
48
49 try:
50     handler.export_obj(**export_kwargs)
51 except Exception as e:
52     traceback.print_exc(file=sys.stdout)
```

### Output from Python Code

```
1  Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
2  Traceback (most recent call last):
3      File "<string>", line 50, in <module>
4      File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 1402, in export_obj
5          utils.check_dictkey(**check_args)
6      File "/Users/jolsen/gh/pytan/lib/pytan/utils.py", line 2509, 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  # Path to lib directory which contains pytan package
2  PYTAN_LIB_PATH = '../lib'
3
4  # connection info for Tanium Server
5  USERNAME = "Tanium User"
6  PASSWORD = "T@n!um"
7  HOST = "172.16.31.128"
8  PORT = "444"
9
10 # Logging conrols
11 LOGLEVEL = 2
12 DEBUGFORMAT = False
13
14 import sys, tempfile
15 sys.path.append(PYTAN_LIB_PATH)
16
17 import pytan
18 handler = pytan.Handler(
19     username=USERNAME,
20     password=PASSWORD,
21     host=HOST,
22     port=PORT,
23     loglevel=LOGLEVEL,
24     debugformat=DEBUGFORMAT,
25 )
26
27 print handler
28
29 # setup the export_obj kwargs for later
30 export_kwargs = {}
31 export_kwargs["export_format"] = u'xml'
32 export_kwargs["minimal"] = u'bad'
33
34 # get the objects that will provide the basetype that we want to use
35 get_kwargs = {
36     'name': [
37         "Computer Name", "IP Route Details", "IP Address",
38         'Folder Name Search with RegEx Match',
39     ],
40     'objtype': 'sensor',
41 }
42 response = handler.get(**get_kwargs)
43 export_kwargs['obj'] = response
44
45 # export the object to a string
46 # this should throw an exception: pytan.utils.HandlerError

```

```
47 import traceback
48
49 try:
50     handler.export_obj(**export_kwargs)
51 except Exception as e:
52     traceback.print_exc(file=sys.stdout)
```

### Output from Python Code

```
1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
2 Traceback (most recent call last):
3   File "<string>", line 50, in <module>
4   File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 1402, in export_obj
5     utils.check_dictkey(**check_args)
6   File "/Users/jolsen/gh/pytan/lib/pytan/utils.py", line 2509, 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 # Path to lib directory which contains pytan package
2 PYTAN_LIB_PATH = '../lib'
3
4 # connection info for Tanium Server
5 USERNAME = "Tanium User"
6 PASSWORD = "T@n!um"
7 HOST = "172.16.31.128"
8 PORT = "444"
9
10 # Logging conrols
11 LOGLEVEL = 2
12 DEBUGFORMAT = False
13
14 import sys, tempfile
15 sys.path.append(PYTAN_LIB_PATH)
16
17 import pytan
18 handler = pytan.Handler(
19     username=USERNAME,
20     password=PASSWORD,
21     host=HOST,
22     port=PORT,
23     loglevel=LOGLEVEL,
24     debugformat=DEBUGFORMAT,
25 )
26
27 print handler
28
29 # setup the export_obj kwargs for later
30 export_kwargs = {}
```

```

31 export_kwargs["export_format"] = u'json'
32 export_kwargs["include_type"] = u'bad'
33
34 # get the objects that will provide the basetype that we want to use
35 get_kwargs = {
36     'name': [
37         "Computer Name", "IP Route Details", "IP Address",
38         'Folder Name Search with RegEx Match',
39     ],
40     'objtype': 'sensor',
41 }
42 response = handler.get(**get_kwargs)
43 export_kwargs['obj'] = response
44
45 # export the object to a string
46 # this should throw an exception: pytan.utils.HandlerError
47 import traceback
48
49 try:
50     handler.export_obj(**export_kwargs)
51 except Exception as e:
52     traceback.print_exc(file=sys.stdout)

```

### Output from Python Code

```

1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
2 Traceback (most recent call last):
3   File "<string>", line 50, in <module>
4   File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 1402, in export_obj
5     utils.check_dictkey(**check_args)
6   File "/Users/jolsen/gh/pytan/lib/pytan/utils.py", line 2509, 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 # Path to lib directory which contains pytan package
2 PYTAN_LIB_PATH = '../lib'
3
4 # connection info for Tanium Server
5 USERNAME = "Tanium User"
6 PASSWORD = "T@n!um"
7 HOST = "172.16.31.128"
8 PORT = "444"
9
10 # Logging conrols
11 LOGLEVEL = 2
12 DEBUGFORMAT = False
13
14 import sys, tempfile

```

```
15 sys.path.append(PYTAN_LIB_PATH)
16
17 import pytan
18 handler = pytan.Handler(
19     username=USERNAME,
20     password=PASSWORD,
21     host=HOST,
22     port=PORT,
23     loglevel=LOGLEVEL,
24     debugformat=DEBUGFORMAT,
25 )
26
27 print handler
28
29 # setup the export_obj kwargs for later
30 export_kwargs = {}
31 export_kwargs["export_format"] = u'json'
32 export_kwargs["explode_json_string_values"] = u'bad'
33
34 # get the objects that will provide the basetype that we want to use
35 get_kwargs = {
36     'name': [
37         "Computer Name", "IP Route Details", "IP Address",
38         'Folder Name Search with RegEx Match',
39     ],
40     'objtype': 'sensor',
41 }
42 response = handler.get(**get_kwargs)
43 export_kwargs['obj'] = response
44
45 # export the object to a string
46 # this should throw an exception: pytan.utils.HandlerError
47 import traceback
48
49 try:
50     handler.export_obj(**export_kwargs)
51 except Exception as e:
52     traceback.print_exc(file=sys.stdout)
```

### Output from Python Code

```
1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
2 Traceback (most recent call last):
3   File "<string>", line 50, in <module>
4   File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 1402, in export_obj
5     utils.check_dictkey(**check_args)
6   File "/Users/jolsen/gh/pytan/lib/pytan/utils.py", line 2509, 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  # Path to lib directory which contains pytan package
2  PYTAN_LIB_PATH = '../lib'
3
4  # connection info for Tanium Server
5  USERNAME = "Tanium User"
6  PASSWORD = "T@n!um"
7  HOST = "172.16.31.128"
8  PORT = "444"
9
10 # Logging conrols
11 LOGLEVEL = 2
12 DEBUGFORMAT = False
13
14 import sys, tempfile
15 sys.path.append(PYTAN_LIB_PATH)
16
17 import pytan
18 handler = pytan.Handler(
19     username=USERNAME,
20     password=PASSWORD,
21     host=HOST,
22     port=PORT,
23     loglevel=LOGLEVEL,
24     debugformat=DEBUGFORMAT,
25 )
26
27 print handler
28
29 # setup the export_obj kwargs for later
30 export_kwargs = {}
31 export_kwargs["export_format"] = u'bad'
32
33 # get the objects that will provide the basetype that we want to use
34 get_kwargs = {
35     'name': [
36         "Computer Name", "IP Route Details", "IP Address",
37         'Folder Name Search with RegEx Match',
38     ],
39     'objtype': 'sensor',
40 }
41 response = handler.get(**get_kwargs)
42 export_kwargs['obj'] = response
43
44 # export the object to a string
45 # this should throw an exception: pytan.utils.HandlerError
46 import traceback
47
48 try:
49     handler.export_obj(**export_kwargs)
50 except Exception as e:
51     traceback.print_exc(file=sys.stdout)

```

## Output from Python Code

```
1 Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
2 Traceback (most recent call last):
3   File "<string>", line 49, in <module>
4   File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 1396, 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
```

## 1.7.2 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, \*\*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 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

**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

**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 strs, 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



- `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': ['<'],
```

**Maps a given set of human strings into the various filter attributes used by the SOAP API. Also used to verify that a manu**

- `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': 'Use
```

**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': {'handler': 'ask_manual'}}`

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_over_time']`

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

## 1.7.4 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 str 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 `xmldict` 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

## 1.7.5 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.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
        ignored)

    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()
```

### 1.7.6 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_suppli
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)
```

## 1.8 taniumpy package

### 1.8.1 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.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*)

```

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)

```

## 1.8.2 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`

### 1.8.3 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.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.SoopError`  
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
```

## 1.9 xmltodict module

Makes working with XML feel like you are working with JSON

```
xmltodict.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 xmltodict
>>> doc = xmltodict.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
...
>>> xmltodict.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
>>> xmltodict.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.

## 1.10 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.

## 1.11 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, 289

**p**

pytan, 3

pytan.constants, 249

pytan.handler, 234

pytan.utils, 251

**t**

taniumpy, 274

taniumpy.object\_types, 276

taniumpy.object\_types.action, 276

taniumpy.object\_types.action\_list, 276

taniumpy.object\_types.action\_list\_info, 276

taniumpy.object\_types.action\_stop, 276

taniumpy.object\_types.action\_stop\_list, 276

taniumpy.object\_types.all\_objects, 276

taniumpy.object\_types.archived\_question, 276

taniumpy.object\_types.archived\_question\_list, 276

taniumpy.object\_types.audit\_data, 276

taniumpy.object\_types.base, 277

taniumpy.object\_types.cache\_filter, 278

taniumpy.object\_types.cache\_filter\_list, 278

taniumpy.object\_types.cache\_info, 278

taniumpy.object\_types.client\_count, 278

taniumpy.object\_types.client\_status, 278

taniumpy.object\_types.column, 278

taniumpy.object\_types.column\_set, 278

taniumpy.object\_types.computer\_group, 278

taniumpy.object\_types.computer\_group\_list, 279

taniumpy.object\_types.computer\_group\_spec, 279

taniumpy.object\_types.computer\_spec\_list, 279

taniumpy.object\_types.error\_list, 279

taniumpy.object\_types.filter, 279

taniumpy.object\_types.filter\_list, 279

taniumpy.object\_types.group, 279

taniumpy.object\_types.group\_list, 279

taniumpy.object\_types.metadata\_item, 279

taniumpy.object\_types.metadata\_list, 280

taniumpy.object\_types.object\_list, 280

taniumpy.object\_types.object\_list\_types, 280

taniumpy.object\_types.options, 280

taniumpy.object\_types.package\_file, 280

taniumpy.object\_types.package\_file\_list, 280

taniumpy.object\_types.package\_file\_status, 280

taniumpy.object\_types.package\_file\_status\_list, 280

taniumpy.object\_types.package\_file\_template, 280

taniumpy.object\_types.package\_file\_template\_list, 280

taniumpy.object\_types.package\_spec, 281

taniumpy.object\_types.package\_spec\_list, 281

taniumpy.object\_types.parameter, 281

taniumpy.object\_types.parameter\_list, 281

taniumpy.object\_types.parse\_job, 281

taniumpy.object\_types.parse\_job\_list, 281

taniumpy.object\_types.parse\_result, 281

taniumpy.object\_types.parse\_result\_group, 281

taniumpy.object\_types.parse\_result\_group\_list, 281

taniumpy.object\_types.parse\_result\_list, 282

taniumpy.object\_types.plugin, 282

taniumpy.object\_types.plugin\_argument, 282

`taniumpy.object_types.plugin_argument_list`, 282

`taniumpy.object_types.plugin_command_list`, 282

`taniumpy.object_types.plugin_list`, 282

`taniumpy.object_types.plugin_schedule`, 282

`taniumpy.object_types.plugin_schedule_list`, 282

`taniumpy.object_types.plugin_sql`, 282

`taniumpy.object_types.plugin_sql_column`, 283

`taniumpy.object_types.plugin_sql_result`, 283

`taniumpy.object_types.question`, 283

`taniumpy.object_types.question_list`, 283

`taniumpy.object_types.question_list_info`, 283

`taniumpy.object_types.result_info`, 283

`taniumpy.object_types.result_set`, 283

`taniumpy.object_types.row`, 284

`taniumpy.object_types.saved_action`, 284

`taniumpy.object_types.saved_action_approval`, 284

`taniumpy.object_types.saved_action_list`, 284

`taniumpy.object_types.saved_action_policy`, 284

`taniumpy.object_types.saved_action_row_id`, 284

`taniumpy.object_types.saved_question`, 284

`taniumpy.object_types.saved_question_list`, 284

`taniumpy.object_types.select`, 285

`taniumpy.object_types.select_list`, 285

`taniumpy.object_types.sensor`, 285

`taniumpy.object_types.sensor_list`, 285

`taniumpy.object_types.sensor_query`, 285

`taniumpy.object_types.sensor_query_list`, 285

`taniumpy.object_types.sensor_string_hints`, 285

`taniumpy.object_types.sensor_subcolumn`, 285

`taniumpy.object_types.sensor_subcolumn_list`, 285

`taniumpy.object_types.sensor_types`, 286

`taniumpy.object_types.soap_error`, 286

`taniumpy.object_types.system_setting`, 286

`taniumpy.object_types.system_settings_list`, 286

`taniumpy.object_types.system_status_aggregate`, 286

`taniumpy.object_types.system_status_list`, 286

`taniumpy.object_types.upload_file`, 286

`taniumpy.object_types.upload_file_list`, 286

`taniumpy.object_types.upload_file_status`, 286

`taniumpy.object_types.user`, 286

`taniumpy.object_types.user_list`, 287

`taniumpy.object_types.user_permissions`, 287

`taniumpy.object_types.user_role`, 287

`taniumpy.object_types.user_role_list`, 287

`taniumpy.object_types.version_aggregate`, 287

`taniumpy.object_types.version_aggregate_list`, 287

`taniumpy.object_types.white_listed_url`, 287

`taniumpy.object_types.white_listed_url_list`, 287

`taniumpy.object_types.xml_error`, 287

`taniumpy.question_asker`, 275

`taniumpy.session`, 274

`test_pytan_func`, 270

`test_pytan_unit`, 265

`threaded_http`, 290

**X**

`xmltodict`, 288

## Symbols

- `__author__` (in module `pytan`), 3
  - `__copyright__` (in module `pytan`), 3
  - `__license__` (in module `pytan`), 3
  - `__module__` (`test_pytan_func.InvalidServerTests` attribute), 270
  - `__module__` (`test_pytan_func.ValidServerTests` attribute), 270
  - `__module__` (`test_pytan_unit.TestDehumanizeExtractionUtils` attribute), 265
  - `__module__` (`test_pytan_unit.TestDehumanizeQuestionFilterUtils` attribute), 266
  - `__module__` (`test_pytan_unit.TestDehumanizeQuestionOptionsUtils` attribute), 266
  - `__module__` (`test_pytan_unit.TestDehumanizeSensorUtils` attribute), 267
  - `__module__` (`test_pytan_unit.TestGenericUtils` attribute), 267
  - `__module__` (`test_pytan_unit.TestManualBuildObjectUtils` attribute), 268
  - `__module__` (`test_pytan_unit.TestManualPackageDefValidateUtils` attribute), 268
  - `__module__` (`test_pytan_unit.TestManualQuestionFilterDefParseUtils` attribute), 268
  - `__module__` (`test_pytan_unit.TestManualQuestionFilterDefValidateUtils` attribute), 268
  - `__module__` (`test_pytan_unit.TestManualQuestionOptionDefParseUtils` attribute), 269
  - `__module__` (`test_pytan_unit.TestManualSensorDefParseUtils` attribute), 269
  - `__module__` (`test_pytan_unit.TestManualSensorDefValidateUtils` attribute), 269
  - `__module__` (`threaded_http.Handler` attribute), 290
  - `__module__` (`threaded_http.ThreadedHTTPServer` attribute), 290
  - `__version__` (in module `pytan`), 3
  - `_export_class_BaseType()` (`pytan.handler.Handler` method), 249
  - `_export_class_ResultSet()` (`pytan.handler.Handler` method), 249
  - `_export_format_csv()` (`pytan.handler.Handler` method), 249
  - `_export_format_json()` (`pytan.handler.Handler` method), 249
  - `_export_format_xml()` (`pytan.handler.Handler` method), 249
  - `_find()` (`pytan.handler.Handler` method), 249
  - `_get_multi()` (`pytan.handler.Handler` method), 249
  - `_get_package_def()` (`pytan.handler.Handler` method), 249
  - `_get_sensor_defs()` (`pytan.handler.Handler` method), 249
  - `_get_single()` (`pytan.handler.Handler` method), 249
  - `_single_find()` (`pytan.handler.Handler` method), 249
- ## A
- `Action` (class in `taniumpy.object_types.action`), 276
  - `ACTION_RESULT_STATUS` (in module `pytan.constants`), 249
  - `ActionList` (class in `taniumpy.object_types.action_list`), 276
  - `ActionListInfo` (class in `taniumpy.object_types.action_list_info`), 276
  - `ActionStop` (class in `taniumpy.object_types.action_stop`), 276
  - `ActionStopList` (class in `taniumpy.object_types.action_stop_list`), 276
  - `add()` (`taniumpy.session.Session` method), 274
  - `add_get_object_report_argparser()` (in module `pytan.utils`), 256
  - `add_report_argparser()` (in module `pytan.utils`), 257
  - `ADD_OBJECT` (`taniumpy.session.Session` attribute), 274
  - `add_report_file_options()` (in module `pytan.utils`), 256
  - `append()` (`taniumpy.object_types.base.BaseType` method), 277
  - `apply_options_obj()` (in module `pytan.utils`), 261
  - `ArchivedQuestion` (class in `taniumpy.object_types.archived_question`), 276
  - `ArchivedQuestionList` (class in `taniumpy.object_types.archived_question_list`), 276
  - `ask()` (`pytan.handler.Handler` method), 235
  - `ASK_KWARGS` (in module `pytan.constants`), 249
  - `ask_manual()` (`pytan.handler.Handler` method), 236
  - `ask_manual_human()` (`pytan.handler.Handler` method), 237

ask\_saved() (pytan.handler.Handler method), 235  
AuditData (class in taniumpy.object\_types.audit\_data), 276  
AUTH\_RES (taniumpy.session.Session attribute), 274  
authenticate() (taniumpy.session.Session method), 274  
AuthorizationError, 274

## B

BadResponseError, 274  
BaseType (class in taniumpy.object\_types.base), 277  
build\_group\_obj() (in module pytan.utils), 261  
build\_manual\_q() (in module pytan.utils), 262  
build\_metadatalist\_obj() (in module pytan.utils), 262  
build\_param\_obj() (in module pytan.utils), 262  
build\_param\_objlist() (in module pytan.utils), 262  
build\_selectlist\_obj() (in module pytan.utils), 263

## C

CacheFilter (class in taniumpy.object\_types.cache\_filter), 278  
CacheFilterList (class in taniumpy.object\_types.cache\_filter\_list), 278  
CacheInfo (class in taniumpy.object\_types.cache\_info), 278  
change\_console\_format() (in module pytan.utils), 252  
check\_dictkey() (in module pytan.utils), 264  
chk\_def\_key() (in module pytan.utils), 264  
ClientCount (class in taniumpy.object\_types.client\_count), 278  
ClientStatus (class in taniumpy.object\_types.client\_status), 278  
Column (class in taniumpy.object\_types.column), 278  
ColumnSet (class in taniumpy.object\_types.column\_set), 278  
ComputerGroup (class in taniumpy.object\_types.computer\_group), 278  
ComputerGroupList (class in taniumpy.object\_types.computer\_group\_list), 279  
ComputerGroupSpec (class in taniumpy.object\_types.computer\_group\_spec), 279  
ComputerSpecList (class in taniumpy.object\_types.computer\_spec\_list), 279  
create\_from\_json() (pytan.handler.Handler method), 241  
create\_group() (pytan.handler.Handler method), 244  
create\_package() (pytan.handler.Handler method), 245  
create\_sensor() (pytan.handler.Handler method), 246  
create\_user() (pytan.handler.Handler method), 246  
create\_whitelisted\_url() (pytan.handler.Handler method), 247  
CustomArgFormat (class in pytan.utils), 252  
CustomArgParse (class in pytan.utils), 252

## D

data() (in module ddt), 289  
ddt (module), 289  
ddt() (in module ddt), 289  
DEBUG\_FORMAT (in module pytan.constants), 249  
DefinitionParserError, 251  
dehumanize\_package() (in module pytan.utils), 258  
dehumanize\_question\_filters() (in module pytan.utils), 258  
dehumanize\_question\_options() (in module pytan.utils), 258  
dehumanize\_sensors() (in module pytan.utils), 258  
delete() (pytan.handler.Handler method), 247  
delete() (taniumpy.session.Session method), 274  
DELETE\_OBJECT (taniumpy.session.Session attribute), 274  
deploy\_action() (pytan.handler.Handler method), 238  
deploy\_action\_asker() (pytan.handler.Handler method), 241  
deploy\_action\_human() (pytan.handler.Handler method), 239  
derive\_param\_default() (in module pytan.utils), 263  
do\_GET() (threaded\_http.Handler method), 290  
DynamicFormatter (class in taniumpy.session), 274

## E

emit() (pytan.utils.SplitStreamHandler method), 252  
empty\_obj() (in module pytan.utils), 263  
error() (pytan.utils.CustomArgParse method), 252  
ErrorList (class in taniumpy.object\_types.error\_list), 279  
explode\_json() (taniumpy.object\_types.base.BaseType method), 277  
EXPORT\_MAPS (in module pytan.constants), 249  
export\_obj() (pytan.handler.Handler method), 242  
export\_to\_report\_file() (pytan.handler.Handler method), 243  
extract\_filter() (in module pytan.utils), 259  
extract\_options() (in module pytan.utils), 259  
extract\_params() (in module pytan.utils), 259  
extract\_selector() (in module pytan.utils), 259

## F

file\_data() (in module ddt), 289  
Filter (class in taniumpy.object\_types.filter), 279  
FILTER\_MAPS (in module pytan.constants), 250  
FILTER\_RE (in module pytan.constants), 250  
FilterList (class in taniumpy.object\_types.filter\_list), 279  
find() (taniumpy.session.Session method), 275  
flatten\_jsonable() (taniumpy.object\_types.base.BaseType method), 277  
FORMATTER() (taniumpy.session.Session method), 274  
from\_jsonable() (taniumpy.object\_types.base.BaseType static method), 277

fromSOAPBody()  
     umpy.object\_types.base.BaseType  
     method), 277

fromSOAPElement()  
     umpy.object\_types.base.BaseType  
     method), 277

fromSOAPElement()  
     umpy.object\_types.column.Column  
     method), 278

fromSOAPElement()  
     umpy.object\_types.column\_set.ColumnSet  
     class method), 278

fromSOAPElement()  
     umpy.object\_types.result\_info.ResultInfo  
     class method), 283

fromSOAPElement()  
     umpy.object\_types.result\_set.ResultSet  
     class method), 283

fromSOAPElement() (taniumpy.object\_types.row.Row  
     class method), 284

## G

get() (pytan.handler.Handler method), 248

get\_all() (pytan.handler.Handler method), 248

get\_ask\_kwargs() (in module pytan.utils), 260

get\_dict\_list\_items() (in module pytan.utils), 253

get\_dict\_list\_len() (in module pytan.utils), 253

get\_filter\_obj() (in module pytan.utils), 263

get\_grp\_opts() (in module pytan.utils), 257

get\_kwargs\_int() (in module pytan.utils), 260

get\_now() (in module pytan.utils), 254

GET\_OBJ\_MAP (in module pytan.constants), 250

get\_obj\_map() (in module pytan.utils), 261

get\_obj\_params() (in module pytan.utils), 264

GET\_OBJECT (taniumpy.session.Session attribute), 274

get\_q\_obj\_map() (in module pytan.utils), 261

get\_req\_kwargs() (in module pytan.utils), 261

GET\_RESULT\_DATA (taniumpy.session.Session at-  
     tribute), 274

get\_result\_data() (pytan.handler.Handler method), 248

GET\_RESULT\_INFO (taniumpy.session.Session at-  
     tribute), 274

get\_result\_info() (pytan.handler.Handler method), 248

get\_server\_info() (taniumpy.session.Session method),  
     275

get\_value() (taniumpy.session.DynamicFormatter  
     method), 274

getResultData() (taniumpy.session.Session method), 275

getResultInfo() (taniumpy.session.Session method), 275

Group (class in taniumpy.object\_types.group), 279

GroupList (class in taniumpy.object\_types.group\_list),  
     279

## H

Handler (class in pytan.handler), 234

Handler (class in threaded\_http), 290

HandlerError, 251

http\_post() (in module taniumpy.session), 275

HttpError, 274

human\_time() (in module pytan.utils), 254

HumanParserError, 251

## I

IncorrectTypeException, 278

INFO\_FORMAT (in module pytan.constants), 250

INFO\_RES (taniumpy.session.Session attribute), 274

InvalidServerTests (class in test\_pytan\_func), 270

is\_auth (taniumpy.session.Session attribute), 275

is\_dict() (in module pytan.utils), 253

is\_hash\_randomized() (in module ddt), 289

is\_list() (in module pytan.utils), 253

is\_num() (in module pytan.utils), 253

is\_str() (in module pytan.utils), 253

## J

jsonify() (in module pytan.utils), 254

## L

load\_file() (in module taniumpy.session), 275

load\_taniumpy\_from\_json() (pytan.handler.Handler  
     method), 242

LOG\_LEVEL\_MAPS (in module pytan.constants), 250

log\_message() (threaded\_http.Handler method), 290

## M

map\_filter() (in module pytan.utils), 260

map\_option() (in module pytan.utils), 260

map\_options() (in module pytan.utils), 260

MetadataItem (class in taniumpy.object\_types.metadata\_item), 279

MetadataList (class in taniumpy.object\_types.metadata\_list), 280

mk\_test\_name() (in module ddt), 290

## O

ObjectList (class in taniumpy.object\_types.object\_list),  
     280

OPTION\_MAPS (in module pytan.constants), 250

OPTION\_RE (in module pytan.constants), 251

Options (class in taniumpy.object\_types.options), 280

## P

PackageFile (class in taniumpy.object\_types.package\_file), 280

PackageFileList (class in taniumpy.object\_types.package\_file\_list), 280

- PackageFileStatus (class in taniumpy.object\_types.package\_file\_status), 280
- PackageFileStatusList (class in taniumpy.object\_types.package\_file\_status\_list), 280
- PackageFileTemplate (class in taniumpy.object\_types.package\_file\_template), 280
- PackageFileTemplateList (class in taniumpy.object\_types.package\_file\_template\_list), 280
- PackageSpec (class in taniumpy.object\_types.package\_spec), 281
- PackageSpecList (class in taniumpy.object\_types.package\_spec\_list), 281
- PARAM\_DELIM (in module pytan.constants), 251
- PARAM\_KEY\_SPLIT (in module pytan.constants), 251
- PARAM\_RE (in module pytan.constants), 251
- PARAM\_SPLIT\_RE (in module pytan.constants), 251
- Parameter (class in taniumpy.object\_types.parameter), 281
- ParameterList (class in taniumpy.object\_types.parameter\_list), 281
- parse() (in module xmltodict), 288
- parse\_defs() (in module pytan.utils), 265
- ParseJob (class in taniumpy.object\_types.parse\_job), 281
- ParseJobList (class in taniumpy.object\_types.parse\_job\_list), 281
- ParseResult (class in taniumpy.object\_types.parse\_result), 281
- ParseResultGroup (class in taniumpy.object\_types.parse\_result\_group), 281
- ParseResultGroupList (class in taniumpy.object\_types.parse\_result\_group\_list), 281
- ParseResultList (class in taniumpy.object\_types.parse\_result\_list), 282
- Plugin (class in taniumpy.object\_types.plugin), 282
- PluginArgument (class in taniumpy.object\_types.plugin\_argument), 282
- PluginArgumentList (class in taniumpy.object\_types.plugin\_argument\_list), 282
- PluginCommandList (class in taniumpy.object\_types.plugin\_command\_list), 282
- PluginList (class in taniumpy.object\_types.plugin\_list), 282
- PluginSchedule (class in taniumpy.object\_types.plugin\_schedule), 282
- PluginScheduleList (class in taniumpy.object\_types.plugin\_schedule\_list), 282
- PluginSql (class in taniumpy.object\_types.plugin\_sql), 282
- PluginSqlColumn (class in taniumpy.object\_types.plugin\_sql\_column), 283
- PluginSqlResult (class in taniumpy.object\_types.plugin\_sql\_result), 283
- POLLING\_INTERVAL (taniumpy.question\_asker.QuestionAsker attribute), 275
- port\_check() (in module pytan.utils), 254
- print\_help() (pytan.utils.CustomArgParse method), 252
- process\_create\_json\_object\_args() (in module pytan.utils), 257
- process\_delete\_object\_args() (in module pytan.utils), 257
- process\_get\_object\_args() (in module pytan.utils), 257
- pytan (module), 3
- pytan.constants (module), 249
- pytan.handler (module), 234
- pytan.utils (module), 251
- ## Q
- Q\_OBJ\_MAP (in module pytan.constants), 251
- Question (class in taniumpy.object\_types.question), 283
- question\_progress() (in module pytan.utils), 264
- QuestionAsker (class in taniumpy.question\_asker), 275
- QuestionList (class in taniumpy.object\_types.question\_list), 283
- QuestionListInfo (class in taniumpy.object\_types.question\_list\_info), 283
- QuestionTimeoutException, 275
- ## R
- remove\_logging\_handler() (in module pytan.utils), 252
- REQ\_KWARGS (in module pytan.constants), 251
- REQUEST\_BODY (taniumpy.session.Session attribute), 274
- ResultInfo (class in taniumpy.object\_types.result\_info), 283
- ResultSet (class in taniumpy.object\_types.result\_set), 283
- Row (class in taniumpy.object\_types.row), 284
- run() (taniumpy.question\_asker.QuestionAsker method), 275
- RunFalse, 251
- ## S
- save() (taniumpy.session.Session method), 275
- SavedAction (class in taniumpy.object\_types.saved\_action), 284
- SavedActionApproval (class in taniumpy.object\_types.saved\_action\_approval), 284



- SavedActionList (class in taniumpy.object\_types.saved\_action\_list), 284
  - SavedActionPolicy (class in taniumpy.object\_types.saved\_action\_policy), 284
  - SavedActionRowIdList (class in taniumpy.object\_types.saved\_action\_row\_id\_list), 284
  - SavedQuestion (class in taniumpy.object\_types.saved\_question), 284
  - SavedQuestionList (class in taniumpy.object\_types.saved\_question\_list), 284
  - seconds\_from\_now() (in module pytan.utils), 254
  - Select (class in taniumpy.object\_types.select), 285
  - SelectList (class in taniumpy.object\_types.select\_list), 285
  - SELECTORS (in module pytan.constants), 251
  - Sensor (class in taniumpy.object\_types.sensor), 285
  - SENSOR\_TYPE\_MAP (in module pytan.constants), 251
  - SensorList (class in taniumpy.object\_types.sensor\_list), 285
  - SensorQuery (class in taniumpy.object\_types.sensor\_query), 285
  - SensorQueryList (class in taniumpy.object\_types.sensor\_query\_list), 285
  - SensorStringHints (class in taniumpy.object\_types.sensor\_string\_hints), 285
  - SensorSubcolumn (class in taniumpy.object\_types.sensor\_subcolumn), 285
  - SensorSubcolumnList (class in taniumpy.object\_types.sensor\_subcolumn\_list), 285
  - server\_version (taniumpy.session.Session attribute), 275
  - Session (class in taniumpy.session), 274
  - session\_id (taniumpy.session.Session attribute), 275
  - set\_all\_loglevels() (in module pytan.utils), 252
  - set\_log\_levels() (in module pytan.utils), 253
  - setPctCompleteThreshold() (taniumpy.question\_asker.QuestionAsker method), 275
  - setup\_ask\_manual\_argparser() (in module pytan.utils), 256
  - setup\_ask\_saved\_argparser() (in module pytan.utils), 256
  - setup\_console\_logging() (in module pytan.utils), 253
  - setup\_create\_json\_object\_argparser() (in module pytan.utils), 256
  - setup\_delete\_object\_argparser() (in module pytan.utils), 256
  - setup\_deploy\_action\_argparser() (in module pytan.utils), 256
  - setup\_get\_object\_argparser() (in module pytan.utils), 256
  - setup\_get\_result\_argparser() (in module pytan.utils), 256
  - setup\_parser() (in module pytan.utils), 256
  - setup\_stop\_action\_argparser() (in module pytan.utils), 256
  - setup\_test() (test\_pytan\_func.ValidServerTests method), 270
  - setUpClass() (test\_pytan\_func.InvalidServerTests class method), 270
  - setUpClass() (test\_pytan\_func.ValidServerTests class method), 270
  - setUpClass() (test\_pytan\_unit.TestManualBuildObjectUtils class method), 268
  - SOAP\_PORT (taniumpy.session.Session attribute), 274
  - SOAP\_RES (taniumpy.session.Session attribute), 274
  - SoapError (class in taniumpy.object\_types.soap\_error), 286
  - spew() (in module test\_pytan\_func), 274
  - SplitStreamHandler (class in pytan.utils), 252
  - stop() (taniumpy.question\_asker.QuestionAsker method), 275
  - stop\_action() (pytan.handler.Handler method), 241
  - SystemSetting (class in taniumpy.object\_types.system\_setting), 286
  - SystemSettingsList (class in taniumpy.object\_types.system\_settings\_list), 286
  - SystemStatusAggregate (class in taniumpy.object\_types.system\_status\_aggregate), 286
  - SystemStatusList (class in taniumpy.object\_types.system\_status\_list), 286
- ## T
- taniumpy (module), 274
  - taniumpy.object\_types (module), 276
  - taniumpy.object\_types.action (module), 276
  - taniumpy.object\_types.action\_list (module), 276
  - taniumpy.object\_types.action\_list\_info (module), 276
  - taniumpy.object\_types.action\_stop (module), 276
  - taniumpy.object\_types.action\_stop\_list (module), 276
  - taniumpy.object\_types.all\_objects (module), 276
  - taniumpy.object\_types.archived\_question (module), 276
  - taniumpy.object\_types.archived\_question\_list (module), 276
  - taniumpy.object\_types.audit\_data (module), 276
  - taniumpy.object\_types.base (module), 277
  - taniumpy.object\_types.cache\_filter (module), 278
  - taniumpy.object\_types.cache\_filter\_list (module), 278
  - taniumpy.object\_types.cache\_info (module), 278
  - taniumpy.object\_types.client\_count (module), 278
  - taniumpy.object\_types.client\_status (module), 278
  - taniumpy.object\_types.column (module), 278
  - taniumpy.object\_types.column\_set (module), 278
  - taniumpy.object\_types.computer\_group (module), 278

- [taniumpy.object\\_types.computer\\_group\\_list \(module\), 279](#)
- [taniumpy.object\\_types.computer\\_group\\_spec \(module\), 279](#)
- [taniumpy.object\\_types.computer\\_spec\\_list \(module\), 279](#)
- [taniumpy.object\\_types.error\\_list \(module\), 279](#)
- [taniumpy.object\\_types.filter \(module\), 279](#)
- [taniumpy.object\\_types.filter\\_list \(module\), 279](#)
- [taniumpy.object\\_types.group \(module\), 279](#)
- [taniumpy.object\\_types.group\\_list \(module\), 279](#)
- [taniumpy.object\\_types.metadata\\_item \(module\), 279](#)
- [taniumpy.object\\_types.metadata\\_list \(module\), 280](#)
- [taniumpy.object\\_types.object\\_list \(module\), 280](#)
- [taniumpy.object\\_types.object\\_list\\_types \(module\), 280](#)
- [taniumpy.object\\_types.options \(module\), 280](#)
- [taniumpy.object\\_types.package\\_file \(module\), 280](#)
- [taniumpy.object\\_types.package\\_file\\_list \(module\), 280](#)
- [taniumpy.object\\_types.package\\_file\\_status \(module\), 280](#)
- [taniumpy.object\\_types.package\\_file\\_status\\_list \(module\), 280](#)
- [taniumpy.object\\_types.package\\_file\\_template \(module\), 280](#)
- [taniumpy.object\\_types.package\\_file\\_template\\_list \(module\), 280](#)
- [taniumpy.object\\_types.package\\_spec \(module\), 281](#)
- [taniumpy.object\\_types.package\\_spec\\_list \(module\), 281](#)
- [taniumpy.object\\_types.parameter \(module\), 281](#)
- [taniumpy.object\\_types.parameter\\_list \(module\), 281](#)
- [taniumpy.object\\_types.parse\\_job \(module\), 281](#)
- [taniumpy.object\\_types.parse\\_job\\_list \(module\), 281](#)
- [taniumpy.object\\_types.parse\\_result \(module\), 281](#)
- [taniumpy.object\\_types.parse\\_result\\_group \(module\), 281](#)
- [taniumpy.object\\_types.parse\\_result\\_group\\_list \(module\), 281](#)
- [taniumpy.object\\_types.parse\\_result\\_list \(module\), 282](#)
- [taniumpy.object\\_types.plugin \(module\), 282](#)
- [taniumpy.object\\_types.plugin\\_argument \(module\), 282](#)
- [taniumpy.object\\_types.plugin\\_argument\\_list \(module\), 282](#)
- [taniumpy.object\\_types.plugin\\_command\\_list \(module\), 282](#)
- [taniumpy.object\\_types.plugin\\_list \(module\), 282](#)
- [taniumpy.object\\_types.plugin\\_schedule \(module\), 282](#)
- [taniumpy.object\\_types.plugin\\_schedule\\_list \(module\), 282](#)
- [taniumpy.object\\_types.plugin\\_sql \(module\), 282](#)
- [taniumpy.object\\_types.plugin\\_sql\\_column \(module\), 283](#)
- [taniumpy.object\\_types.plugin\\_sql\\_result \(module\), 283](#)
- [taniumpy.object\\_types.question \(module\), 283](#)
- [taniumpy.object\\_types.question\\_list \(module\), 283](#)
- [taniumpy.object\\_types.question\\_list\\_info \(module\), 283](#)
- [taniumpy.object\\_types.result\\_info \(module\), 283](#)
- [taniumpy.object\\_types.result\\_set \(module\), 283](#)
- [taniumpy.object\\_types.row \(module\), 284](#)
- [taniumpy.object\\_types.saved\\_action \(module\), 284](#)
- [taniumpy.object\\_types.saved\\_action\\_approval \(module\), 284](#)
- [taniumpy.object\\_types.saved\\_action\\_list \(module\), 284](#)
- [taniumpy.object\\_types.saved\\_action\\_policy \(module\), 284](#)
- [taniumpy.object\\_types.saved\\_action\\_row\\_id\\_list \(module\), 284](#)
- [taniumpy.object\\_types.saved\\_question \(module\), 284](#)
- [taniumpy.object\\_types.saved\\_question\\_list \(module\), 284](#)
- [taniumpy.object\\_types.select \(module\), 285](#)
- [taniumpy.object\\_types.select\\_list \(module\), 285](#)
- [taniumpy.object\\_types.sensor \(module\), 285](#)
- [taniumpy.object\\_types.sensor\\_list \(module\), 285](#)
- [taniumpy.object\\_types.sensor\\_query \(module\), 285](#)
- [taniumpy.object\\_types.sensor\\_query\\_list \(module\), 285](#)
- [taniumpy.object\\_types.sensor\\_string\\_hints \(module\), 285](#)
- [taniumpy.object\\_types.sensor\\_subcolumn \(module\), 285](#)
- [taniumpy.object\\_types.sensor\\_subcolumn\\_list \(module\), 285](#)
- [taniumpy.object\\_types.sensor\\_types \(module\), 286](#)
- [taniumpy.object\\_types.soap\\_error \(module\), 286](#)
- [taniumpy.object\\_types.system\\_setting \(module\), 286](#)
- [taniumpy.object\\_types.system\\_settings\\_list \(module\), 286](#)
- [taniumpy.object\\_types.system\\_status\\_aggregate \(module\), 286](#)
- [taniumpy.object\\_types.system\\_status\\_list \(module\), 286](#)
- [taniumpy.object\\_types.upload\\_file \(module\), 286](#)
- [taniumpy.object\\_types.upload\\_file\\_list \(module\), 286](#)
- [taniumpy.object\\_types.upload\\_file\\_status \(module\), 286](#)
- [taniumpy.object\\_types.user \(module\), 286](#)
- [taniumpy.object\\_types.user\\_list \(module\), 287](#)
- [taniumpy.object\\_types.user\\_permissions \(module\), 287](#)
- [taniumpy.object\\_types.user\\_role \(module\), 287](#)
- [taniumpy.object\\_types.user\\_role\\_list \(module\), 287](#)
- [taniumpy.object\\_types.version\\_aggregate \(module\), 287](#)
- [taniumpy.object\\_types.version\\_aggregate\\_list \(module\), 287](#)
- [taniumpy.object\\_types.white\\_listed\\_url \(module\), 287](#)
- [taniumpy.object\\_types.white\\_listed\\_url\\_list \(module\), 287](#)
- [taniumpy.object\\_types.xml\\_error \(module\), 287](#)
- [taniumpy.question\\_asker \(module\), 275](#)
- [taniumpy.session \(module\), 274](#)
- [test\\_app\\_port\(\) \(in module pytan.utils\), 255](#)
- [test\\_ask\\_kwargs\(\) \(test\\_pytan\\_unit.TestGenericUtils method\), 267](#)
- [test\\_build\\_group\\_obj\(\) \(test\\_pytan\\_unit.TestManualBuildObjectUtils method\), 268](#)
- [test\\_build\\_manual\\_q\(\) \(test\\_pytan\\_unit.TestManualBuildObjectUtils method\), 268](#)
- [test\\_build\\_selectlist\\_obj\\_invalid\\_filter\(\) \(test\\_pytan\\_unit.TestManualBuildObjectUtils](#)



method), 268

test\_build\_selectlist\_obj\_missing\_value()  
(test\_pytan\_unit.TestManualBuildObjectUtils  
method), 268

test\_build\_selectlist\_obj\_noparamssensorobj\_noparams()  
(test\_pytan\_unit.TestManualBuildObjectUtils  
method), 268

test\_build\_selectlist\_obj\_noparamssensorobj\_withparams()  
(test\_pytan\_unit.TestManualBuildObjectUtils  
method), 268

test\_build\_selectlist\_obj\_withparamssensorobj\_noparams()  
(test\_pytan\_unit.TestManualBuildObjectUtils  
method), 268

test\_build\_selectlist\_obj\_withparamssensorobj\_withparams()  
(test\_pytan\_unit.TestManualBuildObjectUtils  
method), 268

test\_empty\_args\_dict() (test\_pytan\_unit.TestDehumanizeSensorUtils  
method), 267

test\_empty\_args\_list() (test\_pytan\_unit.TestDehumanizeSensorUtils  
method), 267

test\_empty\_args\_str() (test\_pytan\_unit.TestDehumanizeSensorUtils  
method), 267

test\_empty\_filterlist() (test\_pytan\_unit.TestDehumanizeQuestionFilterUtils  
method), 266

test\_empty\_filterstr() (test\_pytan\_unit.TestDehumanizeQuestionFilterUtils  
method), 266

test\_empty\_obj() (test\_pytan\_unit.TestGenericUtils  
method), 267

test\_empty\_optionlist() (test\_pytan\_unit.TestDehumanizeQuestionOptionsUtils  
method), 266

test\_empty\_optionstr() (test\_pytan\_unit.TestDehumanizeQuestionOptionsUtils  
method), 266

test\_extract\_filter\_invalid()  
(test\_pytan\_unit.TestDehumanizeExtractionUtils  
method), 265

test\_extract\_filter\_nofilter()  
(test\_pytan\_unit.TestDehumanizeExtractionUtils  
method), 266

test\_extract\_filter\_valid()  
(test\_pytan\_unit.TestDehumanizeExtractionUtils  
method), 266

test\_extract\_filter\_valid\_all()  
(test\_pytan\_unit.TestDehumanizeExtractionUtils  
method), 266

test\_extract\_options\_invalid\_option()  
(test\_pytan\_unit.TestDehumanizeExtractionUtils  
method), 266

test\_extract\_options\_many()  
(test\_pytan\_unit.TestDehumanizeExtractionUtils  
method), 266

test\_extract\_options\_missing\_value\_max\_data\_age()  
(test\_pytan\_unit.TestDehumanizeExtractionUtils  
method), 266

test\_extract\_options\_missing\_value\_value\_type()  
(test\_pytan\_unit.TestDehumanizeExtractionUtils  
method), 266

test\_extract\_options\_nooptions()  
(test\_pytan\_unit.TestDehumanizeExtractionUtils  
method), 266

test\_extract\_options\_single()  
(test\_pytan\_unit.TestDehumanizeExtractionUtils  
method), 266

test\_extract\_params() (test\_pytan\_unit.TestDehumanizeExtractionUtils  
method), 266

test\_extract\_params\_missing\_seperator()  
(test\_pytan\_unit.TestDehumanizeExtractionUtils  
method), 266

test\_extract\_params\_multiparams()  
(test\_pytan\_unit.TestDehumanizeExtractionUtils  
method), 266

test\_extract\_params\_noparams()  
(test\_pytan\_unit.TestDehumanizeExtractionUtils  
method), 266

test\_extract\_selector() (test\_pytan\_unit.TestDehumanizeExtractionUtils  
method), 266

test\_extract\_selector\_use\_name\_if\_noselector()  
(test\_pytan\_unit.TestDehumanizeExtractionUtils  
method), 266

test\_get\_now() (test\_pytan\_unit.TestGenericUtils  
method), 267

test\_get\_obj\_map() (test\_pytan\_unit.TestGenericUtils  
method), 267

test\_get\_q\_obj\_map() (test\_pytan\_unit.TestGenericUtils  
method), 267

test\_invalid1() (test\_pytan\_unit.TestManualPackageDefValidateUtils  
method), 268

test\_invalid1() (test\_pytan\_unit.TestManualQuestionFilterDefValidateUtils  
method), 268

test\_invalid1() (test\_pytan\_unit.TestManualSensorDefValidateUtils  
method), 269

test\_invalid2() (test\_pytan\_unit.TestManualPackageDefValidateUtils  
method), 268

test\_invalid2() (test\_pytan\_unit.TestManualSensorDefValidateUtils  
method), 270

test\_invalid3() (test\_pytan\_unit.TestManualSensorDefValidateUtils  
method), 270

test\_invalid4() (test\_pytan\_unit.TestManualSensorDefValidateUtils  
method), 270

test\_invalid\_connect\_1\_bad\_username()  
(test\_pytan\_func.InvalidServerTests method),  
270

test\_invalid\_connect\_2\_bad\_host\_and\_non\_ssl\_port()  
(test\_pytan\_func.InvalidServerTests method),  
270

test\_invalid\_connect\_3\_bad\_password()  
(test\_pytan\_func.InvalidServerTests method),  
270

<code>test_invalid_connect_4_bad_host_and_bad_port()</code> ( <code>test_pytan_func.InvalidServerTests</code> method), <a href="#">270</a>	<code>test_invalid_export_basetype_6_invalid_export_basetype_json_bad_explode()</code> ( <code>test_pytan_func.ValidServerTests</code> method), <a href="#">271</a>
<code>test_invalid_create_object_1_invalid_create_sensor()</code> ( <code>test_pytan_func.ValidServerTests</code> method), <a href="#">270</a>	<code>test_invalid_export_basetype_7_invalid_export_basetype_bad_format()</code> ( <code>test_pytan_func.ValidServerTests</code> method), <a href="#">271</a>
<code>test_invalid_create_object_from_json_1_invalid_create_savedata_invalid_from_json()</code> ( <code>test_pytan_func.ValidServerTests</code> method), <a href="#">270</a>	<code>test_invalid_export_resultset_1_invalid_export_resultset_csv_bad_sort_subtype()</code> ( <code>test_pytan_func.ValidServerTests</code> method), <a href="#">271</a>
<code>test_invalid_create_object_from_json_2_invalid_create_client_invalid_from_json()</code> ( <code>test_pytan_func.ValidServerTests</code> method), <a href="#">270</a>	<code>test_invalid_export_resultset_2_invalid_export_resultset_csv_bad_sort_type()</code> ( <code>test_pytan_func.ValidServerTests</code> method), <a href="#">271</a>
<code>test_invalid_create_object_from_json_3_invalid_create_user_invalid_from_json()</code> ( <code>test_pytan_func.ValidServerTests</code> method), <a href="#">270</a>	<code>test_invalid_export_resultset_3_invalid_export_resultset_csv_bad_expand_type()</code> ( <code>test_pytan_func.ValidServerTests</code> method), <a href="#">271</a>
<code>test_invalid_create_object_from_json_4_invalid_create_settings_invalid_from_json()</code> ( <code>test_pytan_func.ValidServerTests</code> method), <a href="#">270</a>	<code>test_invalid_export_resultset_4_invalid_export_resultset_csv_bad_sensors_type()</code> ( <code>test_pytan_func.ValidServerTests</code> method), <a href="#">271</a>
<code>test_invalid_deploy_action_1_invalid_deploy_action_run_failed()</code> ( <code>test_pytan_func.ValidServerTests</code> method), <a href="#">270</a>	<code>test_invalid_export_resultset_5_invalid_export_resultset_bad_format()</code> ( <code>test_pytan_func.ValidServerTests</code> method), <a href="#">271</a>
<code>test_invalid_deploy_action_2_invalid_deploy_action_package_help()</code> ( <code>test_pytan_func.ValidServerTests</code> method), <a href="#">270</a>	<code>test_invalid_filter1()</code> ( <code>test_pytan_unit.TestDehumanizeQuestionFilterUtils</code> method), <a href="#">266</a>
<code>test_invalid_deploy_action_3_invalid_deploy_action_package()</code> ( <code>test_pytan_func.ValidServerTests</code> method), <a href="#">270</a>	<code>test_invalid_filter2()</code> ( <code>test_pytan_unit.TestDehumanizeQuestionFilterUtils</code> method), <a href="#">266</a>
<code>test_invalid_deploy_action_4_invalid_deploy_action_options_help()</code> ( <code>test_pytan_func.ValidServerTests</code> method), <a href="#">270</a>	<code>test_invalid_filter3()</code> ( <code>test_pytan_unit.TestDehumanizeQuestionFilterUtils</code> method), <a href="#">266</a>
<code>test_invalid_deploy_action_5_invalid_deploy_action_empty_package()</code> ( <code>test_pytan_func.ValidServerTests</code> method), <a href="#">270</a>	<code>test_invalid_get_object_1_invalid_get_action_single_by_name()</code> ( <code>test_pytan_func.ValidServerTests</code> method), <a href="#">271</a>
<code>test_invalid_deploy_action_6_invalid_deploy_action_filters_help()</code> ( <code>test_pytan_func.ValidServerTests</code> method), <a href="#">270</a>	<code>test_invalid_get_object_2_invalid_get_question_by_name()</code> ( <code>test_pytan_func.ValidServerTests</code> method), <a href="#">271</a>
<code>test_invalid_deploy_action_7_invalid_deploy_action_missing_parameters()</code> ( <code>test_pytan_func.ValidServerTests</code> method), <a href="#">271</a>	<code>test_invalid_option1()</code> ( <code>test_pytan_unit.TestDehumanizeQuestionOptionUtils</code> method), <a href="#">266</a>
<code>test_invalid_export_basetype_1_invalid_export_basetype_csv_bad_export_question()</code> ( <code>test_pytan_func.ValidServerTests</code> method), <a href="#">271</a>	<code>test_invalid_option2()</code> ( <code>test_pytan_unit.TestDehumanizeQuestionOptionUtils</code> method), <a href="#">266</a>
<code>test_invalid_export_basetype_2_invalid_export_basetype_csv_bad_valid_subtype()</code> ( <code>test_pytan_func.ValidServerTests</code> method), <a href="#">271</a>	<code>test_invalid_port()</code> ( <code>test_pytan_unit.TestGenericUtils</code> method), <a href="#">267</a>
<code>test_invalid_export_basetype_3_invalid_export_basetype_csv_bad_valid_type()</code> ( <code>test_pytan_func.ValidServerTests</code> method), <a href="#">271</a>	<code>test_invalid_ask_manual_human_question_parameter_1()</code> ( <code>test_pytan_func.ValidServerTests</code> method), <a href="#">271</a>
<code>test_invalid_export_basetype_4_invalid_export_basetype_x_invalid_ask_manual_human_question_filter()</code> ( <code>test_pytan_func.ValidServerTests</code> method), <a href="#">271</a>	<code>test_invalid_ask_manual_human_question_filter_help()</code> ( <code>test_pytan_func.ValidServerTests</code> method), <a href="#">271</a>
<code>test_invalid_export_basetype_5_invalid_export_basetype_json_invalid_ask_manual_human_question_parameter_2()</code> ( <code>test_pytan_func.ValidServerTests</code> method), <a href="#">271</a>	<code>test_invalid_ask_manual_human_question_option()</code> ( <code>test_pytan_func.ValidServerTests</code> method), <a href="#">271</a>
	<code>test_invalid_ask_manual_human_question_type()</code> ( <code>test_pytan_func.ValidServerTests</code> method), <a href="#">271</a>
	<code>test_invalid_ask_manual_human_question_type_2()</code> ( <code>test_pytan_func.ValidServerTests</code> method), <a href="#">271</a>
	<code>test_invalid_ask_manual_human_question_type_3()</code> ( <code>test_pytan_func.ValidServerTests</code> method), <a href="#">271</a>
	<code>test_invalid_ask_manual_human_question_type_4()</code> ( <code>test_pytan_func.ValidServerTests</code> method), <a href="#">271</a>
	<code>test_invalid_ask_manual_human_question_type_5()</code> ( <code>test_pytan_func.ValidServerTests</code> method), <a href="#">271</a>

test\_invalid\_question\_6\_invalid\_ask\_manual\_human\_question\_option\_help() (test\_pytan\_unit.TestManualQuestionFilterDefParseUtils method), 271

test\_invalid\_question\_7\_invalid\_ask\_manual\_question\_sensor() (test\_pytan\_func.ValidServerTests method), 271

test\_invalid\_question\_8\_invalid\_ask\_manual\_human\_question\_sensor\_help() (test\_pytan\_func.ValidServerTests method), 271

test\_is\_dict() (test\_pytan\_unit.TestGenericUtils method), 267

test\_is\_list() (test\_pytan\_unit.TestGenericUtils method), 267

test\_is\_not\_dict() (test\_pytan\_unit.TestGenericUtils method), 267

test\_is\_not\_list() (test\_pytan\_unit.TestGenericUtils method), 267

test\_is\_not\_num() (test\_pytan\_unit.TestGenericUtils method), 267

test\_is\_not\_str() (test\_pytan\_unit.TestGenericUtils method), 267

test\_is\_num() (test\_pytan\_unit.TestGenericUtils method), 267

test\_is\_str() (test\_pytan\_unit.TestGenericUtils method), 267

test\_jsonify() (test\_pytan\_unit.TestGenericUtils method), 267

test\_multi\_filter\_list() (test\_pytan\_unit.TestDehumanizeQuestionFilterUtils method), 266

test\_multi\_list\_complex() (test\_pytan\_unit.TestDehumanizeSensorUtils method), 267

test\_option\_list\_many() (test\_pytan\_unit.TestDehumanizeQuestionOptionUtils method), 266

test\_option\_list\_multi() (test\_pytan\_unit.TestDehumanizeQuestionOptionUtils method), 266

test\_option\_list\_single() (test\_pytan\_unit.TestDehumanizeQuestionOptionUtils method), 266

test\_option\_str() (test\_pytan\_unit.TestDehumanizeQuestionOptionUtils method), 266

test\_parse\_complex() (test\_pytan\_unit.TestManualSensorDefParseUtils method), 269

test\_parse\_dict\_hash() (test\_pytan\_unit.TestManualSensorDefParseUtils method), 269

test\_parse\_dict\_id() (test\_pytan\_unit.TestManualSensorDefParseUtils method), 269

test\_parse\_dict\_name() (test\_pytan\_unit.TestManualSensorDefParseUtils method), 269

test\_parse\_emptydict() (test\_pytan\_unit.TestManualQuestionFilterDefParseUtils method), 268

test\_parse\_emptydict() (test\_pytan\_unit.TestManualQuestionOptionDefParseUtils method), 269

test\_parse\_emptydict() (test\_pytan\_unit.TestManualSensorDefParseUtils method), 269

test\_parse\_emptylist() (test\_pytan\_unit.TestManualQuestionFilterDefParseUtils method), 268

test\_parse\_emptylist() (test\_pytan\_unit.TestManualQuestionOptionDefParseUtils method), 269

test\_parse\_emptylist() (test\_pytan\_unit.TestManualSensorDefParseUtils method), 269

test\_parse\_emptystr() (test\_pytan\_unit.TestManualQuestionFilterDefParseUtils method), 268

test\_parse\_emptystr() (test\_pytan\_unit.TestManualQuestionOptionDefParseUtils method), 269

test\_parse\_emptystr() (test\_pytan\_unit.TestManualSensorDefParseUtils method), 269

test\_parse\_list() (test\_pytan\_unit.TestManualQuestionOptionDefParseUtils method), 269

test\_parse\_multi\_filter() (test\_pytan\_unit.TestManualQuestionFilterDefParseUtils method), 268

test\_parse\_noargs() (test\_pytan\_unit.TestManualQuestionFilterDefParseUtils method), 268

test\_parse\_noargs() (test\_pytan\_unit.TestManualQuestionOptionDefParseUtils method), 269

test\_parse\_noargs() (test\_pytan\_unit.TestManualSensorDefParseUtils method), 269

test\_parse\_none() (test\_pytan\_unit.TestManualQuestionFilterDefParseUtils method), 268

test\_parse\_none() (test\_pytan\_unit.TestManualQuestionOptionDefParseUtils method), 269

test\_parse\_none() (test\_pytan\_unit.TestManualSensorDefParseUtils method), 269

test\_parse\_options\_dict() (test\_pytan\_unit.TestManualQuestionOptionDefParseUtils method), 269

test\_parse\_single\_filter() (test\_pytan\_unit.TestManualQuestionFilterDefParseUtils method), 268

test\_parse\_str() (test\_pytan\_unit.TestManualQuestionFilterDefParseUtils method), 268

test\_parse\_str() (test\_pytan\_unit.TestManualQuestionOptionDefParseUtils method), 269

test\_parse\_str1() (test\_pytan\_unit.TestManualSensorDefParseUtils method), 269

test\_pytan\_func (module), 270

test\_pytan\_unit (module), 265

test\_req\_kwargs() (test\_pytan\_unit.TestGenericUtils method), 267

test\_single\_filter\_list() (test\_pytan\_unit.TestDehumanizeQuestionFilterUtils method), 266

test\_single\_filter\_str() (test\_pytan\_unit.TestDehumanizeQuestionFilterUtils method), 266

test\_single\_str() (test\_pytan\_unit.TestDehumanizeSensorUtils method), 267

test\_single\_str\_complex1() (test\_pytan\_unit.TestDehumanizeSensorUtils method), 267

test\_single\_str\_complex2() (test\_pytan\_unit.TestDehumanizeSensorUtils method), 267

method), 267

test\_single\_str\_with\_filter()  
(test\_pytan\_unit.TestDehumanizeSensorUtils  
method), 267

test\_valid1() (test\_pytan\_unit.TestManualPackageDefValidateUtils  
method), 268

test\_valid1() (test\_pytan\_unit.TestManualQuestionFilterDefValidateUtils  
method), 269

test\_valid1() (test\_pytan\_unit.TestManualSensorDefValidateUtils  
method), 270

test\_valid2() (test\_pytan\_unit.TestManualPackageDefValidateUtils  
method), 268

test\_valid2() (test\_pytan\_unit.TestManualQuestionFilterDefValidateUtils  
method), 269

test\_valid2() (test\_pytan\_unit.TestManualSensorDefValidateUtils  
method), 270

test\_valid3() (test\_pytan\_unit.TestManualSensorDefValidateUtils  
method), 270

test\_valid4() (test\_pytan\_unit.TestManualSensorDefValidateUtils  
method), 270

test\_valid\_create\_object\_1\_create\_user()  
(test\_pytan\_func.ValidServerTests  
method), 271

test\_valid\_create\_object\_2\_create\_package()  
(test\_pytan\_func.ValidServerTests  
method), 271

test\_valid\_create\_object\_3\_create\_group()  
(test\_pytan\_func.ValidServerTests  
method), 271

test\_valid\_create\_object\_4\_create\_whitelisted\_url()  
(test\_pytan\_func.ValidServerTests  
method), 271

test\_valid\_create\_object\_from\_json\_1\_create\_package\_from\_json()  
(test\_pytan\_func.ValidServerTests  
method), 271

test\_valid\_create\_object\_from\_json\_2\_create\_user\_from\_json()  
(test\_pytan\_func.ValidServerTests  
method), 271

test\_valid\_create\_object\_from\_json\_3\_create\_saved\_question\_from\_json()  
(test\_pytan\_func.ValidServerTests  
method), 271

test\_valid\_create\_object\_from\_json\_4\_create\_action\_from\_json()  
(test\_pytan\_func.ValidServerTests  
method), 271

test\_valid\_create\_object\_from\_json\_5\_create\_sensor\_from\_json()  
(test\_pytan\_func.ValidServerTests  
method), 271

test\_valid\_create\_object\_from\_json\_6\_create\_question\_from\_json()  
(test\_pytan\_func.ValidServerTests  
method), 271

test\_valid\_create\_object\_from\_json\_7\_create\_whitelisted\_url\_from\_json()  
(test\_pytan\_func.ValidServerTests  
method), 271

test\_valid\_create\_object\_from\_json\_8\_create\_group\_from\_json()  
(test\_pytan\_func.ValidServerTests  
method), 271

test\_valid\_deploy\_action\_1\_deploy\_action\_simple\_against\_windows\_computers()  
(test\_pytan\_func.ValidServerTests  
method), 271

test\_valid\_deploy\_action\_2\_deploy\_action\_simple\_without\_results()  
(test\_pytan\_func.ValidServerTests  
method), 271

test\_valid\_deploy\_action\_3\_deploy\_action\_with\_params\_against\_windows\_computers()  
(test\_pytan\_func.ValidServerTests  
method), 272

test\_valid\_deploy\_action\_4\_deploy\_action\_simple()  
(test\_pytan\_func.ValidServerTests  
method), 272

test\_valid\_export\_basetype\_10\_export\_basetype\_xml\_default\_options()  
(test\_pytan\_func.ValidServerTests  
method), 272

test\_valid\_export\_basetype\_11\_export\_basetype\_csv\_with\_explode\_true()  
(test\_pytan\_func.ValidServerTests  
method), 272

test\_valid\_export\_basetype\_12\_export\_basetype\_json\_explode\_false()  
(test\_pytan\_func.ValidServerTests  
method), 272

test\_valid\_export\_basetype\_13\_export\_basetype\_json\_type\_false()  
(test\_pytan\_func.ValidServerTests  
method), 272

test\_valid\_export\_basetype\_14\_export\_basetype\_json\_default\_options()  
(test\_pytan\_func.ValidServerTests  
method), 272

test\_valid\_export\_basetype\_1\_export\_basetype\_csv\_with\_sort\_list()  
(test\_pytan\_func.ValidServerTests  
method), 272

test\_valid\_export\_basetype\_2\_export\_basetype\_csv\_with\_explode\_false()  
(test\_pytan\_func.ValidServerTests  
method), 272

test\_valid\_export\_basetype\_3\_export\_basetype\_json\_type\_true()  
(test\_pytan\_func.ValidServerTests  
method), 272

test\_valid\_export\_basetype\_4\_export\_basetype\_xml\_minimal\_false()  
(test\_pytan\_func.ValidServerTests  
method), 272

test\_valid\_export\_basetype\_5\_export\_basetype\_xml\_minimal\_true()  
(test\_pytan\_func.ValidServerTests  
method), 272

test\_valid\_export\_basetype\_6\_export\_basetype\_csv\_with\_sort\_empty\_list()  
(test\_pytan\_func.ValidServerTests  
method), 272

test\_valid\_export\_basetype\_7\_export\_basetype\_csv\_default\_options()  
(test\_pytan\_func.ValidServerTests  
method), 272

test\_valid\_export\_basetype\_8\_export\_basetype\_json\_explode\_true()  
(test\_pytan\_func.ValidServerTests  
method), 272

test_valid_export_basetype_9_export_basetype_csv_with_sort_true() (test_pytan_func.ValidServerTests method), <a href="#">272</a>	test_valid_get_object_14_get_sensor_by_id() (test_pytan_func.ValidServerTests method), <a href="#">272</a>
test_valid_export_resultset_10_export_resultset_csv_defaults_valid() (test_pytan_func.ValidServerTests method), <a href="#">272</a>	test_valid_get_object_15_get_all_groups() (test_pytan_func.ValidServerTests method), <a href="#">272</a>
test_valid_export_resultset_11_export_resultset_csv_type_true() (test_pytan_func.ValidServerTests method), <a href="#">272</a>	test_valid_get_object_16_get_all_sensors() (test_pytan_func.ValidServerTests method), <a href="#">273</a>
test_valid_export_resultset_12_export_resultset_csv_all_options_valid() (test_pytan_func.ValidServerTests method), <a href="#">272</a>	test_valid_get_object_17_get_sensor_by_mixed() (test_pytan_func.ValidServerTests method), <a href="#">273</a>
test_valid_export_resultset_13_export_resultset_csv_sort_false() (test_pytan_func.ValidServerTests method), <a href="#">272</a>	test_valid_get_object_18_get_whitelisted_url_by_id() (test_pytan_func.ValidServerTests method), <a href="#">273</a>
test_valid_export_resultset_1_export_resultset_json() (test_pytan_func.ValidServerTests method), <a href="#">272</a>	test_valid_get_object_19_get_group_by_name() (test_pytan_func.ValidServerTests method), <a href="#">273</a>
test_valid_export_resultset_2_export_resultset_csv_sensor_true() (test_pytan_func.ValidServerTests method), <a href="#">272</a>	test_valid_get_object_1_get_all_users() (test_pytan_func.ValidServerTests method), <a href="#">273</a>
test_valid_export_resultset_3_export_resultset_csv_type_false() (test_pytan_func.ValidServerTests method), <a href="#">272</a>	test_valid_get_object_20_get_all_whitelisted_urls() (test_pytan_func.ValidServerTests method), <a href="#">273</a>
test_valid_export_resultset_4_export_resultset_csv_expand_false() (test_pytan_func.ValidServerTests method), <a href="#">272</a>	test_valid_get_object_21_get_sensor_by_hash() (test_pytan_func.ValidServerTests method), <a href="#">273</a>
test_valid_export_resultset_5_export_resultset_csv_sort_empty() (test_pytan_func.ValidServerTests method), <a href="#">272</a>	test_valid_get_object_22_get_package_by_name() (test_pytan_func.ValidServerTests method), <a href="#">273</a>
test_valid_export_resultset_6_export_resultset_csv_sort_true() (test_pytan_func.ValidServerTests method), <a href="#">272</a>	test_valid_get_object_23_get_all_clients() (test_pytan_func.ValidServerTests method), <a href="#">273</a>
test_valid_export_resultset_7_export_resultset_csv_sort_list() (test_pytan_func.ValidServerTests method), <a href="#">272</a>	test_valid_get_object_24_get_sensor_by_names() (test_pytan_func.ValidServerTests method), <a href="#">273</a>
test_valid_export_resultset_8_export_resultset_csv_sensor_false() (test_pytan_func.ValidServerTests method), <a href="#">272</a>	test_valid_get_object_25_get_all_packages() (test_pytan_func.ValidServerTests method), <a href="#">273</a>
test_valid_export_resultset_9_export_resultset_csv_expand_true() (test_pytan_func.ValidServerTests method), <a href="#">272</a>	test_valid_get_object_26_get_saved_question_by_name() (test_pytan_func.ValidServerTests method), <a href="#">273</a>
test_valid_get_object_10_get_all_saved_questions() (test_pytan_func.ValidServerTests method), <a href="#">272</a>	test_valid_get_object_27_get_all_actions() (test_pytan_func.ValidServerTests method), <a href="#">273</a>
test_valid_get_object_11_get_user_by_name() (test_pytan_func.ValidServerTests method), <a href="#">272</a>	test_valid_get_object_28_get_user_by_id() (test_pytan_func.ValidServerTests method), <a href="#">273</a>
test_valid_get_object_12_get_all_userroles() (test_pytan_func.ValidServerTests method), <a href="#">272</a>	test_valid_get_object_29_get_sensor_by_name() (test_pytan_func.ValidServerTests method), <a href="#">273</a>
test_valid_get_object_13_get_all_questions() (test_pytan_func.ValidServerTests method), <a href="#">272</a>	test_valid_get_object_2_get_action_by_id() (test_pytan_func.ValidServerTests method), <a href="#">273</a>





TestManualQuestionOptionDefParseUtils (class in test\_pytan\_unit), 269

TestManualSensorDefParseUtils (class in test\_pytan\_unit), 269

TestManualSensorDefValidateUtils (class in test\_pytan\_unit), 269

threaded\_http (module), 290

threaded\_http() (in module threaded\_http), 290

ThreadedHTTPServer (class in threaded\_http), 290

to\_flat\_dict() (taniumpy.object\_types.base.BaseType method), 277

to\_flat\_dict\_explode\_json() (taniumpy.object\_types.base.BaseType method), 277

to\_json() (taniumpy.object\_types.base.BaseType static method), 277

to\_json() (taniumpy.object\_types.result\_set.ResultSet static method), 283

to\_jsonable() (taniumpy.object\_types.base.BaseType method), 277

to\_jsonable() (taniumpy.object\_types.result\_set.ResultSet method), 283

toSOAPBody() (taniumpy.object\_types.base.BaseType method), 277

toSOAPElement() (taniumpy.object\_types.base.BaseType method), 277

VersionAggregate (class in taniumpy.object\_types.version\_aggregate), 287

VersionAggregateList (class in taniumpy.object\_types.version\_aggregate\_list), 287

## W

WhiteListedUrl (class in taniumpy.object\_types.white\_listed\_url), 287

WhiteListedUrlList (class in taniumpy.object\_types.white\_listed\_url\_list), 287

write\_csv() (taniumpy.object\_types.base.BaseType static method), 277

write\_csv() (taniumpy.object\_types.result\_set.ResultSet static method), 284

## X

xml\_pretty() (in module pytan.utils), 255

xml\_pretty\_resultobj() (in module pytan.utils), 255

xml\_pretty\_resultxml() (in module pytan.utils), 255

XmlError (class in taniumpy.object\_types.xml\_error), 287

xmldict (module), 288

## U

unpack() (in module ddt), 290

unparse() (in module xmldict), 289

UPDATE\_OBJECT (taniumpy.session.Session attribute), 274

UploadFile (class in taniumpy.object\_types.upload\_file), 286

UploadFileList (class in taniumpy.object\_types.upload\_file\_list), 286

UploadFileStatus (class in taniumpy.object\_types.upload\_file\_status), 286

User (class in taniumpy.object\_types.user), 286

UserList (class in taniumpy.object\_types.user\_list), 287

UserPermissions (class in taniumpy.object\_types.user\_permissions), 287

UserRole (class in taniumpy.object\_types.user\_role), 287

UserRoleList (class in taniumpy.object\_types.user\_role\_list), 287

## V

val\_package\_def() (in module pytan.utils), 265

val\_q\_filter\_defs() (in module pytan.utils), 265

val\_sensor\_defs() (in module pytan.utils), 265

ValidServerTests (class in test\_pytan\_func), 270

version\_check() (in module pytan.utils), 255