PyTan Documentation

Release 1.0.1

Jim Olsen

December 08, 2014

CONTENTS

CHAPTER ONE

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

```python
# Path to lib directory which contains pytan package
PYTAN_LIB_PATH = '../lib'

# connection info for Tanium Server
USERNAME = "Tanium User"
PASSWORD = "T@n!um"
HOST = "172.16.31.128"
PORT = "444"

# Logging conrols
LOGLEVEL = 2
DEBUGFORMAT = False

import sys, tempfile
sys.path.append(PYTAN_LIB_PATH)

import pytan
handler = pytan.Handler(
    username=USERNAME,
    password=PASSWORD,
    host=HOST,
    port=PORT,
    loglevel=LOGLEVEL,
    debugformat=DEBUGFORMAT,
)

print handler
```

Output from Python Code

```
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

```python
# Path to lib directory which contains pytan package
PYTAN_LIB_PATH = '../lib'

# connection info for Tanium Server
USERNAME = "Tanium User"
PASSWORD = "T@n!um"
HOST = "172.16.31.128"
PORT = "444"

# Logging conrols
LOGLEVEL = 2
DEBUGFORMAT = False

import sys, tempfile
sys.path.append(PYTAN_LIB_PATH)

import pytan
handler = pytan.Handler(
    username=USERNAME,
    password=PASSWORD,
    host=HOST,
    port=PORT,
    loglevel=LOGLEVEL,
    debugformat=DEBUGFORMAT,
)

print handler

# setup the arguments for the handler method
kwargs = {}
kwargs["qtype"] = u'saved'
kwargs["name"] = [u'Installed Applications']

# call the handler with the ask method, passing in kwargs for arguments
response = handler.ask(**kwargs)
import pprint, io

print ""
print "Type of response: ", type(response)

print ""
print "Pretty print of response:"
print pprint.pformat(response)

print ""
print "Equivalent Question if it were to be asked in the Tanium Console: "
print response['question_object'].query_text

# create an IO stream to store CSV results to
out = io.BytesIO()

# call the write_csv() method to convert response to CSV and store it in out
response['question_results'].write_csv(out, response['question_results'])

print ""
print "CSV Results of response: "
out = out.getvalue()
if len(out.splitlines()) > 15:
    out = out.splitlines()[0:15]
    out.append('..trimmed for brevity..')
    out = '\n'.join(out)
print out
```

Output from Python Code

```
Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
2014-12-08 16:19:51,121 INFO     question_progress: Results 60000% (Get Installed Applications from

Type of response:  <type 'dict'>

Pretty print of response:
{'question_object': <taniumpy.object_types.saved_question.SavedQuestion object at 0x10212e290>,
 'question_results': <taniumpy.object_types.result_set.ResultSet object at 0x102b1fa90>}

Equivalent Question if it were to be asked in the Tanium Console:
Get Installed Applications from all machines

CSV Results of response:
Count,Name,Silent Uninstall String,Uninstallable,Version
714,[too many results],None,None,None
1,update-manager-core,nothing,Not Uninstallable,1:0.196.12
1,libminiupnpc8,nothing,Not Uninstallable,1.6-3ubuntu2.14.04.1
1,iso-codes,nothing,Not Uninstallable,3.52-1
1,docbook-dtds,nothing,Not Uninstallable,1.0
1,libexttextcat-2.0-0,nothing,Not Uninstallable,3.4.3-1ubuntu1
1,Google Search,nothing,Not Uninstallable,37.0.2062.120
1,gnome-user-share,nothing,Not Uninstallable,2.28.2
1,libblkid1:amd64,nothing,Not Uninstallable,2.20.1-5.1ubuntu20.1
1,fipscheck-lib,nothing,Not Uninstallable,1.2.0
1,gsm,nothing,Not Uninstallable,1.0.13
1,VoiceOver Quickstart,nothing,Not Uninstallable,6.0
1,VoiceOver Utility,nothing,Not Uninstallable,6.0
1,growisofs,nothing,Not Uninstallable,7.1-10build1
..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

```
# Path to lib directory which contains pytan package
PYTAN_LIB_PATH = '../lib'

# connection info for Tanium Server
USERNAME = "Tanium User"
PASSWORD = "T@n!um"
HOST = "172.16.31.128"
PORT = "444"

# Logging conrols
LOGLEVEL = 2
DEBUGFORMAT = False

import sys, tempfile
sys.path.append(PYTAN_LIB_PATH)

import pytan
handler = pytan.Handler(
    username=USERNAME,
    password=PASSWORD,
    host=HOST,
    port=PORT,
    loglevel=LOGLEVEL,
    debugformat=DEBUGFORMAT,
)

print handler

# setup the arguments for the handler method
kwargs = {}
kwargs["qtype"] = u'saved'
kwargs["name"] = u'Installed Applications'

# call the handler with the ask method, passing in kwargs for arguments
response = handler.ask(**kwargs)
import pprint, io

print ""
print "Type of response: ", type(response)

print ""
print "Pretty print of response:"
print pprint.pformat(response)

print ""
print "Equivalent Question if it were to be asked in the Tanium Console: "
print response['question_object'].query_text

# create an IO stream to store CSV results to
out = io.BytesIO()

# call the write_csv() method to convert response to CSV and store it in out
```

```
response['question_results'].write_csv(out, response['question_results'])

print ""
print "CSV Results of response: "
out = out.getvalue()
if len(out.splitlines()) > 15:
    out = out.splitlines()[0:15]
    out.append('..trimmed for brevity..')
    out = '\n'.join(out)
print out
```

Output from Python Code

```
Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
2014-12-08 16:19:51,778 INFO     question_progress: Results 60000% (Get Installed Applications from

Type of response:  <type 'dict'>

Pretty print of response:
{'question_object': <taniumpy.object_types.saved_question.SavedQuestion object at 0x102b24790>,
 'question_results': <taniumpy.object_types.result_set.ResultSet object at 0x1029e7fd0>}

Equivalent Question if it were to be asked in the Tanium Console:
Get Installed Applications from all machines

CSV Results of response:
Count,Name,Silent Uninstall String,Uninstallable,Version
714,[too many results],None,None,None
1,update-manager-core,nothing,Not Uninstallable,1:0.196.12
1,libminiupnpc8,nothing,Not Uninstallable,1.6-3ubuntu2.14.04.1
1,iso-codes,nothing,Not Uninstallable,3.52-1
1,docbook-dtds,nothing,Not Uninstallable,1.0
1,libexttextcat-2.0-0,nothing,Not Uninstallable,3.4.3-1ubuntu1
1,Google Search,nothing,Not Uninstallable,37.0.2062.120
1,gnome-user-share,nothing,Not Uninstallable,2.28.2
1,libblkid1:amd64,nothing,Not Uninstallable,2.20.1-5.1ubuntu20.1
1,fipscheck-lib,nothing,Not Uninstallable,1.2.0
1,gsm,nothing,Not Uninstallable,1.0.13
1,VoiceOver Quickstart,nothing,Not Uninstallable,6.0
1,VoiceOver Utility,nothing,Not Uninstallable,6.0
1,growisofs,nothing,Not Uninstallable,7.1-10build1
..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

```
# Path to lib directory which contains pytan package
PYTAN_LIB_PATH = '../lib'

```

```
# connection info for Tanium Server
USERNAME = "Tanium User"
PASSWORD = "T@n!um"
HOST = "172.16.31.128"
PORT = "444"

# Logging conrols
LOGLEVEL = 2
DEBUGFORMAT = False

import sys, tempfile
sys.path.append(PYTAN_LIB_PATH)

import pytan
handler = pytan.Handler(
    username=USERNAME,
    password=PASSWORD,
    host=HOST,
    port=PORT,
    loglevel=LOGLEVEL,
    debugformat=DEBUGFORMAT,
)

print handler

# setup the arguments for the handler method
kwargs = {}
kwargs["sensors"] = u'Computer Name'
kwargs["qtype"] = u'manual_human'

# call the handler with the ask method, passing in kwargs for arguments
response = handler.ask(**kwargs)
import pprint, io

print ""
print "Type of response: ", type(response)

print ""
print "Pretty print of response:"
print pprint.pformat(response)

print ""
print "Equivalent Question if it were to be asked in the Tanium Console: "
print response['question_object'].query_text

# create an IO stream to store CSV results to
out = io.BytesIO()

# call the write_csv() method to convert response to CSV and store it in out
response['question_results'].write_csv(out, response['question_results'])

print ""
print "CSV Results of response: "
out = out.getvalue()
if len(out.splitlines()) > 15:
    out = out.splitlines()[0:15]
    out.append('..trimmed for brevity..')
    out = '\n'.join(out)
```

```
print out
```

Output from Python Code

```
Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
2014-12-08 16:19:52,403 INFO     question_progress: Results 0% (Get Computer Name from all machines)
2014-12-08 16:19:57,417 INFO     question_progress: Results 67% (Get Computer Name from all machines
2014-12-08 16:20:02,434 INFO     question_progress: Results 100% (Get Computer Name from all machine

Type of response:  <type 'dict'>

Pretty print of response:
{'question_object': <taniumpy.object_types.question.Question object at 0x1021b6dd0>,
 'question_results': <taniumpy.object_types.result_set.ResultSet object at 0x1029fe610>}

Equivalent Question if it were to be asked in the Tanium Console:
Get Computer Name from all machines

CSV Results of response:
Computer Name
Casus-Belli.local
jtanium1.localdomain
ubuntu.(none)
localhost.(none)
Jims-Mac.local
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

```
# Path to lib directory which contains pytan package
PYTAN_LIB_PATH = '../lib'

# connection info for Tanium Server
USERNAME = "Tanium User"
PASSWORD = "T@n!um"
HOST = "172.16.31.128"
PORT = "444"

# Logging conrols
LOGLEVEL = 2
DEBUGFORMAT = False

import sys, tempfile
sys.path.append(PYTAN_LIB_PATH)

import pytan
handler = pytan.Handler(
    username=USERNAME,
    password=PASSWORD,
    host=HOST,
    port=PORT,
    loglevel=LOGLEVEL,
    debugformat=DEBUGFORMAT,
)

print handler

# setup the arguments for the handler method
kwargs = {}
kwargs["sensors"] = [u'Computer Name', u'Installed Applications']
kwargs["qtype"] = u'manual_human'

# call the handler with the ask method, passing in kwargs for arguments
response = handler.ask(**kwargs)
import pprint, io

print ""
print "Type of response: ", type(response)

print ""
print "Pretty print of response:"
print pprint.pformat(response)

print ""
print "Equivalent Question if it were to be asked in the Tanium Console: "
print response['question_object'].query_text

# create an IO stream to store CSV results to
out = io.BytesIO()

# call the write_csv() method to convert response to CSV and store it in out
response['question_results'].write_csv(out, response['question_results'])

print ""
print "CSV Results of response: "
out = out.getvalue()
if len(out.splitlines()) > 15:
    out = out.splitlines()[0:15]
    out.append('..trimmed for brevity..')
    out = '\n'.join(out)
print out
```

Output from Python Code

```
Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
2014-12-08 16:20:02,573 INFO     question_progress: Results 0% (Get Computer Name and Installed Appl
2014-12-08 16:20:07,596 INFO     question_progress: Results 67% (Get Computer Name and Installed App
2014-12-08 16:20:12,620 INFO     question_progress: Results 100% (Get Computer Name and Installed Ap

Type of response:  <type 'dict'>

Pretty print of response:
{'question_object': <taniumpy.object_types.question.Question object at 0x1029f7390>,
 'question_results': <taniumpy.object_types.result_set.ResultSet object at 0x1028dfa10>}

```

```
Equivalent Question if it were to be asked in the Tanium Console:
Get Computer Name and Installed Applications from all machines

CSV Results of response:
Computer Name,Name,Silent Uninstall String,Uninstallable,Version
Casus-Belli.local,"Google Search
MakePDF
Wish
Time Machine
AppleGraphicsWarning
soagent
SpeechService
AinuIM
Pass Viewer
PressAndHold
PluginIM
UserNotificationCenter
FaceTime
ScreenSaverEngine
..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

```python
# Path to lib directory which contains pytan package
PYTAN_LIB_PATH = '../lib'

# connection info for Tanium Server
USERNAME = "Tanium User"
PASSWORD = "T@n!um"
HOST = "172.16.31.128"
PORT = "444"

# Logging conrols
LOGLEVEL = 2
DEBUGFORMAT = False

import sys, tempfile
sys.path.append(PYTAN_LIB_PATH)

import pytan
handler = pytan.Handler(
    username=USERNAME,
    password=PASSWORD,
    host=HOST,
    port=PORT,
    loglevel=LOGLEVEL,
    debugformat=DEBUGFORMAT,
)

```

```python
print handler

# setup the arguments for the handler method
kwargs = {}
kwargs["sensors"] = [u'name:Computer Name', u'name:Installed Applications']
kwargs["qtype"] = u'manual_human'

# call the handler with the ask method, passing in kwargs for arguments
response = handler.ask(**kwargs)
import pprint, io

print ""
print "Type of response: ", type(response)

print ""
print "Pretty print of response:"
print pprint.pformat(response)

print ""
print "Equivalent Question if it were to be asked in the Tanium Console: "
print response['question_object'].query_text

# create an IO stream to store CSV results to
out = io.BytesIO()

# call the write_csv() method to convert response to CSV and store it in out
response['question_results'].write_csv(out, response['question_results'])

print ""
print "CSV Results of response: "
out = out.getvalue()
if len(out.splitlines()) > 15:
    out = out.splitlines()[0:15]
    out.append('..trimmed for brevity..')
    out = '\n'.join(out)
print out
```

Output from Python Code

```
Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
2014-12-08 16:20:12,928 INFO     question_progress: Results 0% (Get Computer Name and Installed Appl
2014-12-08 16:20:17,953 INFO     question_progress: Results 83% (Get Computer Name and Installed App
2014-12-08 16:20:22,978 INFO     question_progress: Results 100% (Get Computer Name and Installed Ap

Type of response:  <type 'dict'>

Pretty print of response:
{'question_object': <taniumpy.object_types.question.Question object at 0x1028e2950>,
 'question_results': <taniumpy.object_types.result_set.ResultSet object at 0x102b6f550>}

Equivalent Question if it were to be asked in the Tanium Console:
Get Computer Name and Installed Applications from all machines

CSV Results of response:
Computer Name,Name,Silent Uninstall String,Uninstallable,Version
Casus-Belli.local,"Google Search
MakePDF
```

```
Wish
Time Machine
AppleGraphicsWarning
soagent
SpeechService
AinuIM
Pass Viewer
PressAndHold
PluginIM
UserNotificationCenter
FaceTime
ScreenSaverEngine
..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

```python
# Path to lib directory which contains pytan package
PYTAN_LIB_PATH = '../lib'

# connection info for Tanium Server
USERNAME = "Tanium User"
PASSWORD = "T@n!um"
HOST = "172.16.31.128"
PORT = "444"

# Logging conrols
LOGLEVEL = 2
DEBUGFORMAT = False

import sys, tempfile
sys.path.append(PYTAN_LIB_PATH)

import pytan
handler = pytan.Handler(
    username=USERNAME,
    password=PASSWORD,
    host=HOST,
    port=PORT,
    loglevel=LOGLEVEL,
    debugformat=DEBUGFORMAT,
)

print handler

# setup the arguments for the handler method
kwargs = {}
kwargs["sensors"] = u'Folder Name Search with RegEx Match{dirname=Program Files,regex=Microsoft.*}'
```

```
kwargs["qtype"] = u'manual_human'

# call the handler with the ask method, passing in kwargs for arguments
response = handler.ask(**kwargs)
import pprint, io

print ""
print "Type of response: ", type(response)

print ""
print "Pretty print of response:"
print pprint.pformat(response)

print ""
print "Equivalent Question if it were to be asked in the Tanium Console: "
print response['question_object'].query_text

# create an IO stream to store CSV results to
out = io.BytesIO()

# call the write_csv() method to convert response to CSV and store it in out
response['question_results'].write_csv(out, response['question_results'])

print ""
print "CSV Results of response: "
out = out.getvalue()
if len(out.splitlines()) > 15:
    out = out.splitlines()[0:15]
    out.append('..trimmed for brevity..')
    out = '\n'.join(out)
print out
```

Output from Python Code

```
Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
2014-12-08 16:20:23,296 INFO     question_progress: Results 0% (Get Folder Name Search with RegEx Ma
2014-12-08 16:20:28,314 INFO     question_progress: Results 0% (Get Folder Name Search with RegEx Ma
2014-12-08 16:20:33,331 INFO     question_progress: Results 0% (Get Folder Name Search with RegEx Ma
2014-12-08 16:20:38,351 INFO     question_progress: Results 0% (Get Folder Name Search with RegEx Ma
2014-12-08 16:20:43,372 INFO     question_progress: Results 17% (Get Folder Name Search with RegEx M
2014-12-08 16:20:48,394 INFO     question_progress: Results 83% (Get Folder Name Search with RegEx M
2014-12-08 16:20:53,410 INFO     question_progress: Results 100% (Get Folder Name Search with RegEx 

Type of response:  <type 'dict'>

Pretty print of response:
{'question_object': <taniumpy.object_types.question.Question object at 0x102b6fd90>,
 'question_results': <taniumpy.object_types.result_set.ResultSet object at 0x102bb1810>}

Equivalent Question if it were to be asked in the Tanium Console:
Get Folder Name Search with RegEx Match[No, Program Files, No, ] from all machines

CSV Results of response:
Count,"Folder Name Search with RegEx Match[No, Program Files, No, ]"
1,C:\Program Files\Tanium\Tanium Server\ApacheBackup2014-09-16-20-44-23\cgi-bin
2,C:\Program Files\VMware\VMware Tools\plugins\vmsvc
1,C:\Program Files\Microsoft SQL Server\110\Setup Bootstrap\SQLServer2012\1040_ITA_LP\x64\1040\help
```

```
1,C:\Program Files\Common Files\Microsoft Shared\VS7Debug
1,C:\Program Files\Tanium\Tanium Server\Apache24\manual\style
1,C:\Program Files\Tanium\Tanium Server\Apache24\htdocs\console\history
2,C:\Program Files\Common Files\VMware\Drivers\vmci\sockets\include
2,C:\Program Files\Common Files\Microsoft Shared\ink\ar-SA
1,C:\Program Files\Tanium\Tanium Server\plugins\console\Dashboards
1,C:\Program Files\Tanium\Tanium Server\CertificateBackup2014-11-17-11-17-33
2,C:\Program Files\Common Files\SpeechEngines\Microsoft
1,C:\Program Files\Tanium\Tanium Server\ApacheBackup2014-09-16-20-44-23\modules
2,C:\Program Files\Common Files\Microsoft Shared\ink\ru-RU
1,C:\Program Files\Microsoft SQL Server\110\DTS\ForEachEnumerators\en
..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

```python
# Path to lib directory which contains pytan package
PYTAN_LIB_PATH = '../lib'

# connection info for Tanium Server
USERNAME = "Tanium User"
PASSWORD = "T@n!um"
HOST = "172.16.31.128"
PORT = "444"

# Logging conrols
LOGLEVEL = 2
DEBUGFORMAT = False

import sys, tempfile
sys.path.append(PYTAN_LIB_PATH)

import pytan
handler = pytan.Handler(
    username=USERNAME,
    password=PASSWORD,
    host=HOST,
    port=PORT,
    loglevel=LOGLEVEL,
    debugformat=DEBUGFORMAT,
)

print handler

# setup the arguments for the handler method
kwargs = {}
kwargs["sensors"] = u'Computer Name{fake=Dweedle}'
kwargs["qtype"] = u'manual_human'

# call the handler with the ask method, passing in kwargs for arguments
```

```python
response = handler.ask(**kwargs)
import pprint, io

print ""
print "Type of response: ", type(response)

print ""
print "Pretty print of response:"
print pprint.pformat(response)

print ""
print "Equivalent Question if it were to be asked in the Tanium Console: "
print response['question_object'].query_text

# create an IO stream to store CSV results to
out = io.BytesIO()

# call the write_csv() method to convert response to CSV and store it in out
response['question_results'].write_csv(out, response['question_results'])

print ""
print "CSV Results of response: "
out = out.getvalue()
if len(out.splitlines()) > 15:
    out = out.splitlines()[0:15]
    out.append('..trimmed for brevity..')
    out = '\n'.join(out)
print out
```

Output from Python Code

```
Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
2014-12-08 16:20:53,557 INFO     question_progress: Results 0% (Get Computer Name from all machines)
2014-12-08 16:20:58,574 INFO     question_progress: Results 100% (Get Computer Name from all machine

Type of response:  <type 'dict'>

Pretty print of response:
{'question_object': <taniumpy.object_types.question.Question object at 0x102b82450>,
 'question_results': <taniumpy.object_types.result_set.ResultSet object at 0x1029f70d0>}

Equivalent Question if it were to be asked in the Tanium Console:
Get Computer Name from all machines

CSV Results of response:
Computer Name
Casus-Belli.local
jtanium1.localdomain
ubuntu.(none)
localhost.(none)
Jims-Mac.local
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

```
# Path to lib directory which contains pytan package
PYTAN_LIB_PATH = '../lib'

# connection info for Tanium Server
USERNAME = "Tanium User"
PASSWORD = "T@n!um"
HOST = "172.16.31.128"
PORT = "444"

# Logging conrols
LOGLEVEL = 2
DEBUGFORMAT = False

import sys, tempfile
sys.path.append(PYTAN_LIB_PATH)

import pytan
handler = pytan.Handler(
    username=USERNAME,
    password=PASSWORD,
    host=HOST,
    port=PORT,
    loglevel=LOGLEVEL,
    debugformat=DEBUGFORMAT,
)

print handler

# setup the arguments for the handler method
kwargs = {}
kwargs["sensors"] = [u'Folder Name Search with RegEx Match{dirname=Program Files,regex=Microsoft.*}'
 u'Computer Name']
kwargs["qtype"] = u'manual_human'

# call the handler with the ask method, passing in kwargs for arguments
response = handler.ask(**kwargs)
import pprint, io

print ""
print "Type of response: ", type(response)

print ""
print "Pretty print of response:"
print pprint.pformat(response)

print ""
print "Equivalent Question if it were to be asked in the Tanium Console: "
print response['question_object'].query_text

# create an IO stream to store CSV results to
out = io.BytesIO()

# call the write_csv() method to convert response to CSV and store it in out
response['question_results'].write_csv(out, response['question_results'])

print ""
print "CSV Results of response: "
out = out.getvalue()
if len(out.splitlines()) > 15:
    out = out.splitlines()[0:15]
    out.append('..trimmed for brevity..')
    out = '\n'.join(out)
print out
```

Output from Python Code

```
Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
2014-12-08 16:20:58,703 INFO     question_progress: Results 0% (Get Computer Name from all machines)
2014-12-08 16:21:03,720 INFO     question_progress: Results 0% (Get Computer Name from all machines)
2014-12-08 16:21:08,737 INFO     question_progress: Results 0% (Get Computer Name from all machines)
2014-12-08 16:21:13,757 INFO     question_progress: Results 0% (Get Computer Name from all machines)
2014-12-08 16:21:18,777 INFO     question_progress: Results 0% (Get Computer Name from all machines)
2014-12-08 16:21:23,793 INFO     question_progress: Results 0% (Get Computer Name from all machines)
2014-12-08 16:21:28,811 INFO     question_progress: Results 0% (Get Computer Name from all machines)
2014-12-08 16:21:33,827 INFO     question_progress: Results 0% (Get Computer Name from all machines)
2014-12-08 16:21:38,843 INFO     question_progress: Results 0% (Get Computer Name from all machines)
2014-12-08 16:21:43,863 INFO     question_progress: Results 0% (Get Computer Name from all machines)
2014-12-08 16:21:48,878 INFO     question_progress: Results 0% (Get Computer Name from all machines)
2014-12-08 16:21:53,901 INFO     question_progress: Results 0% (Get Computer Name from all machines)
2014-12-08 16:21:58,915 INFO     question_progress: Results 0% (Get Computer Name from all machines)
2014-12-08 16:22:03,935 INFO     question_progress: Results 33% (Get Computer Name from all machines
2014-12-08 16:22:08,954 INFO     question_progress: Results 100% (Get Computer Name from all machine

Type of response:  <type 'dict'>

Pretty print of response:
{'question_object': <taniumpy.object_types.question.Question object at 0x102129990>,
 'question_results': <taniumpy.object_types.result_set.ResultSet object at 0x102967e10>}

Equivalent Question if it were to be asked in the Tanium Console:
Get Computer Name from all machines

CSV Results of response:
Computer Name,"Folder Name Search with RegEx Match[No, Program Files, No, ]"
Casus-Belli.local,Windows Only
ubuntu.(none),Windows Only
localhost.(none),Windows Only
Jims-Mac.local,Windows Only
jtanium1.localdomain,"C:\Program Files\Tanium\Tanium Server\ApacheBackup2014-09-16-20-44-23\cgi-bin
C:\Program Files\VMware\VMware Tools\plugins\vmsvc
C:\Program Files\Microsoft SQL Server\110\Setup Bootstrap\SQLServer2012\1040_ITA_LP\x64\1040\help
C:\Program Files\Common Files\Microsoft Shared\VS7Debug
C:\Program Files\Tanium\Tanium Server\Apache24\manual\style
C:\Program Files\Tanium\Tanium Server\Apache24\htdocs\console\history
```

```
C:\Program Files\Common Files\VMware\Drivers\vmci\sockets\include
C:\Program Files\Common Files\Microsoft Shared\ink\ar-SA
C:\Program Files\Tanium\Tanium Server\plugins\console\Dashboards
C:\Program Files\Tanium\Tanium Server\CertificateBackup2014-11-17-11-17-33
..trimmed for brevity..
```

Ask manual human question sensor with parameters and no supplied parameters

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

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

Example Python Code

```python
# Path to lib directory which contains pytan package
PYTAN_LIB_PATH = '../lib'

# connection info for Tanium Server
USERNAME = "Tanium User"
PASSWORD = "T@n!um"
HOST = "172.16.31.128"
PORT = "444"

# Logging conrols
LOGLEVEL = 2
DEBUGFORMAT = False

import sys, tempfile
sys.path.append(PYTAN_LIB_PATH)

import pytan
handler = pytan.Handler(
    username=USERNAME,
    password=PASSWORD,
    host=HOST,
    port=PORT,
    loglevel=LOGLEVEL,
    debugformat=DEBUGFORMAT,
)

print handler

# setup the arguments for the handler method
kwargs = {}
kwargs["sensors"] = u'Folder Name Search with RegEx Match'
kwargs["qtype"] = u'manual_human'

# call the handler with the ask method, passing in kwargs for arguments
response = handler.ask(**kwargs)
import pprint, io

print ""
print "Type of response: ", type(response)

print ""
print "Pretty print of response:"
print pprint.pformat(response)

print ""
print "Equivalent Question if it were to be asked in the Tanium Console: "
print response['question_object'].query_text

# create an IO stream to store CSV results to
out = io.BytesIO()

# call the write_csv() method to convert response to CSV and store it in out
response['question_results'].write_csv(out, response['question_results'])

print ""
print "CSV Results of response: "
out = out.getvalue()
if len(out.splitlines()) > 15:
    out = out.splitlines()[0:15]
    out.append('..trimmed for brevity..')
    out = '\n'.join(out)
print out
```

Output from Python Code

```
Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
2014-12-08 16:22:09,102 INFO     question_progress: Results 0% (Get Folder Name Search with RegEx Ma
2014-12-08 16:22:14,122 INFO     question_progress: Results 0% (Get Folder Name Search with RegEx Ma
2014-12-08 16:22:19,141 INFO     question_progress: Results 0% (Get Folder Name Search with RegEx Ma
2014-12-08 16:22:24,157 INFO     question_progress: Results 0% (Get Folder Name Search with RegEx Ma
2014-12-08 16:22:29,173 INFO     question_progress: Results 0% (Get Folder Name Search with RegEx Ma
2014-12-08 16:22:34,194 INFO     question_progress: Results 0% (Get Folder Name Search with RegEx Ma
2014-12-08 16:22:39,212 INFO     question_progress: Results 0% (Get Folder Name Search with RegEx Ma
2014-12-08 16:22:44,230 INFO     question_progress: Results 0% (Get Folder Name Search with RegEx Ma
2014-12-08 16:22:49,247 INFO     question_progress: Results 0% (Get Folder Name Search with RegEx Ma
2014-12-08 16:22:54,270 INFO     question_progress: Results 33% (Get Folder Name Search with RegEx M
2014-12-08 16:22:59,299 INFO     question_progress: Results 67% (Get Folder Name Search with RegEx M
2014-12-08 16:23:04,324 INFO     question_progress: Results 67% (Get Folder Name Search with RegEx M
2014-12-08 16:23:09,344 INFO     question_progress: Results 83% (Get Folder Name Search with RegEx M
2014-12-08 16:23:14,368 INFO     question_progress: Results 100% (Get Folder Name Search with RegEx

Type of response:  <type 'dict'>

Pretty print of response:
{'question_object': <taniumpy.object_types.question.Question object at 0x1021a5ad0>,
 'question_results': <taniumpy.object_types.result_set.ResultSet object at 0x102b2e110>}

Equivalent Question if it were to be asked in the Tanium Console:
Get Folder Name Search with RegEx Match[No, , No, ] from all machines

CSV Results of response:
Count,"Folder Name Search with RegEx Match[No, , No, ]"
40981,[too many results]
2,C:\Windows\winsxs\amd64_microsoft-windows-s..structure.resources_31bf3856ad364e35_6.1.7600.16385_e
2,C:\Windows\winsxs\x86_microsoft-windows-e..-host-authenticator_31bf3856ad364e35_6.1.7601.17514_non
1,C:\Windows\winsxs\amd64_microsoft-windows-ocspsvc_31bf3856ad364e35_6.1.7601.22807_none_3bfeae72930
```

```
1,C:\Windows\winsxs\amd64_microsoft-windows-c..ityclient.resources_31bf3856ad364e35_6.1.7601.22865_e
1,C:\Program Files (x86)\Tanium\Tanium Client\Downloads\Action_192\grep\share\locale\bg\LC_MESSAGES
2,C:\Windows\assembly\NativeImages_v2.0.50727_64\System.Xml
1,C:\Users\Jim Olsen\Desktop\SysinternalsSuite
2,C:\Windows\winsxs\amd64_microsoft-windows-scripting.resources_31bf3856ad364e35_6.1.7600.16385_en-u
2,C:\Windows\winsxs\x86_microsoft-windows-mlang.resources_31bf3856ad364e35_6.1.7600.16385_ru-ru_cf3a
1,C:\Windows\winsxs\x86_microsoft-windows-directshow-dvdsupport_31bf3856ad364e35_6.1.7601.21987_none
1,C:\Windows\winsxs\amd64_microsoft-windows-ie-internetexplorer_31bf3856ad364e35_11.2.9600.17041_non
1,C:\Users\Jim Olsen\AppData\Local\Google
2,C:\Windows\winsxs\x86_microsoft-windows-e..nt-client.resources_31bf3856ad364e35_6.1.7600.16385_en-
..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

```python
# Path to lib directory which contains pytan package
PYTAN_LIB_PATH = '../lib'

# connection info for Tanium Server
USERNAME = "Tanium User"
PASSWORD = "T@n!um"
HOST = "172.16.31.128"
PORT = "444"

# Logging conrols
LOGLEVEL = 2
DEBUGFORMAT = False

import sys, tempfile
sys.path.append(PYTAN_LIB_PATH)

import pytan
handler = pytan.Handler(
    username=USERNAME,
    password=PASSWORD,
    host=HOST,
    port=PORT,
    loglevel=LOGLEVEL,
    debugformat=DEBUGFORMAT,
)

print handler

# setup the arguments for the handler method
kwargs = {}
kwargs["sensors"] = u'Folder Name Search with RegEx Match{dirname=Program Files,regex=Microsoft.*},
kwargs["qtype"] = u'manual_human'

```

```
# call the handler with the ask method, passing in kwargs for arguments
response = handler.ask(**kwargs)
import pprint, io

print ""
print "Type of response: ", type(response)

print ""
print "Pretty print of response:"
print pprint.pformat(response)

print ""
print "Equivalent Question if it were to be asked in the Tanium Console: "
print response['question_object'].query_text

# create an IO stream to store CSV results to
out = io.BytesIO()

# call the write_csv() method to convert response to CSV and store it in out
response['question_results'].write_csv(out, response['question_results'])

print ""
print "CSV Results of response: "
out = out.getvalue()
if len(out.splitlines()) > 15:
    out = out.splitlines()[0:15]
    out.append('..trimmed for brevity..')
    out = '\n'.join(out)
print out
```

Output from Python Code

```
Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
2014-12-08 16:23:14,712 INFO     question_progress: Results 0% (Get Folder Name Search with RegEx Ma
2014-12-08 16:23:19,729 INFO     question_progress: Results 0% (Get Folder Name Search with RegEx Ma
2014-12-08 16:23:24,747 INFO     question_progress: Results 0% (Get Folder Name Search with RegEx Ma
2014-12-08 16:23:29,765 INFO     question_progress: Results 0% (Get Folder Name Search with RegEx Ma
2014-12-08 16:23:34,783 INFO     question_progress: Results 0% (Get Folder Name Search with RegEx Ma
2014-12-08 16:23:39,800 INFO     question_progress: Results 0% (Get Folder Name Search with RegEx Ma
2014-12-08 16:23:44,820 INFO     question_progress: Results 0% (Get Folder Name Search with RegEx Ma
2014-12-08 16:23:49,838 INFO     question_progress: Results 0% (Get Folder Name Search with RegEx Ma
2014-12-08 16:23:54,858 INFO     question_progress: Results 0% (Get Folder Name Search with RegEx Ma
2014-12-08 16:23:59,886 INFO     question_progress: Results 0% (Get Folder Name Search with RegEx Ma
2014-12-08 16:24:04,902 INFO     question_progress: Results 0% (Get Folder Name Search with RegEx Ma
2014-12-08 16:24:09,919 INFO     question_progress: Results 0% (Get Folder Name Search with RegEx Ma
2014-12-08 16:24:14,940 INFO     question_progress: Results 0% (Get Folder Name Search with RegEx Ma
2014-12-08 16:24:19,957 INFO     question_progress: Results 0% (Get Folder Name Search with RegEx Ma
2014-12-08 16:24:24,975 INFO     question_progress: Results 50% (Get Folder Name Search with RegEx M
2014-12-08 16:24:29,990 INFO     question_progress: Results 83% (Get Folder Name Search with RegEx M
2014-12-08 16:24:35,009 INFO     question_progress: Results 83% (Get Folder Name Search with RegEx M
2014-12-08 16:24:40,028 INFO     question_progress: Results 100% (Get Folder Name Search with RegEx

Type of response:  <type 'dict'>

Pretty print of response:
{'question_object': <taniumpy.object_types.question.Question object at 0x102b34650>,
 'question_results': <taniumpy.object_types.result_set.ResultSet object at 0x102116e50>}
```

```

Equivalent Question if it were to be asked in the Tanium Console:
Get Folder Name Search with RegEx Match[No, Program Files, No, ] contains "Shared" from all machines

CSV Results of response:
Count,"Folder Name Search with RegEx Match[No, Program Files, No, ]"
4,[no results]
1,C:\Program Files\Common Files\Microsoft Shared\VS7Debug
2,C:\Program Files\Common Files\Microsoft Shared\ink\ar-SA
2,C:\Program Files\Common Files\Microsoft Shared\ink\ru-RU
2,C:\Program Files\Common Files\Microsoft Shared\ink\fsdefinitions\keypad
2,C:\Program Files\Common Files\Microsoft Shared\ink
2,C:\Program Files\Common Files\Microsoft Shared\ink\sv-SE
2,C:\Program Files\Common Files\Microsoft Shared\ink\uk-UA
2,C:\Program Files\Common Files\Microsoft Shared\ink\sl-SI
2,C:\Program Files\Common Files\Microsoft Shared\ink\hu-HU
2,C:\Program Files\Common Files\Microsoft Shared\ink\zh-TW
2,C:\Program Files\Common Files\Microsoft Shared\ink\zh-CN
2,C:\Program Files\Common Files\Microsoft Shared\ink\fi-FI
2,C:\Program Files\Common Files\Microsoft Shared
..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

```python
# Path to lib directory which contains pytan package
PYTAN_LIB_PATH = '../lib'

# connection info for Tanium Server
USERNAME = "Tanium User"
PASSWORD = "T@n!um"
HOST = "172.16.31.128"
PORT = "444"

# Logging conrols
LOGLEVEL = 2
DEBUGFORMAT = False

import sys, tempfile
sys.path.append(PYTAN_LIB_PATH)

import pytan
handler = pytan.Handler(
    username=USERNAME,
    password=PASSWORD,
    host=HOST,
    port=PORT,
    loglevel=LOGLEVEL,
    debugformat=DEBUGFORMAT,
)

print handler

# setup the arguments for the handler method
kwargs = {}
kwargs["sensors"] = u'Operating System, that contains:Windows, opt:match_all_values, opt:ignore_case
kwargs["qtype"] = u'manual_human'

# call the handler with the ask method, passing in kwargs for arguments
response = handler.ask(**kwargs)
import pprint, io

print ""
print "Type of response: ", type(response)

print ""
print "Pretty print of response:"
print pprint.pformat(response)

print ""
print "Equivalent Question if it were to be asked in the Tanium Console: "
print response['question_object'].query_text

# create an IO stream to store CSV results to
out = io.BytesIO()

# call the write_csv() method to convert response to CSV and store it in out
response['question_results'].write_csv(out, response['question_results'])

print ""
print "CSV Results of response: "
out = out.getvalue()
if len(out.splitlines()) > 15:
    out = out.splitlines()[0:15]
    out.append('..trimmed for brevity..')
    out = '\n'.join(out)
print out
```

Output from Python Code

```
Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
2014-12-08 16:24:40,124 INFO     question_progress: Results 0% (Get Operating System contains "Windo
2014-12-08 16:24:45,141 INFO     question_progress: Results 50% (Get Operating System contains "Wind
2014-12-08 16:24:50,158 INFO     question_progress: Results 100% (Get Operating System contains "Win

Type of response:  <type 'dict'>

Pretty print of response:
{'question_object': <taniumpy.object_types.question.Question object at 0x102130210>,
 'question_results': <taniumpy.object_types.result_set.ResultSet object at 0x1021236d0>}

Equivalent Question if it were to be asked in the Tanium Console:
```

```
Get Operating System contains "Windows" from all machines

CSV Results of response:
Count,Operating System
4,[no results]
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

```python
# Path to lib directory which contains pytan package
PYTAN_LIB_PATH = '../lib'

# connection info for Tanium Server
USERNAME = "Tanium User"
PASSWORD = "T@n!um"
HOST = "172.16.31.128"
PORT = "444"

# Logging conrols
LOGLEVEL = 2
DEBUGFORMAT = False

import sys, tempfile
sys.path.append(PYTAN_LIB_PATH)

import pytan
handler = pytan.Handler(
    username=USERNAME,
    password=PASSWORD,
    host=HOST,
    port=PORT,
    loglevel=LOGLEVEL,
    debugformat=DEBUGFORMAT,
)

print handler

# setup the arguments for the handler method
kwargs = {}
kwargs["sensors"] = u'Folder Name Search with RegEx Match{dirname=Program Files,regex=Microsoft.*},
kwargs["qtype"] = u'manual_human'

# call the handler with the ask method, passing in kwargs for arguments
response = handler.ask(**kwargs)
import pprint, io

```

```
print ""
print "Type of response: ", type(response)

print ""
print "Pretty print of response:"
print pprint.pformat(response)

print ""
print "Equivalent Question if it were to be asked in the Tanium Console: "
print response['question_object'].query_text

# create an IO stream to store CSV results to
out = io.BytesIO()

# call the write_csv() method to convert response to CSV and store it in out
response['question_results'].write_csv(out, response['question_results'])

print ""
print "CSV Results of response: "
out = out.getvalue()
if len(out.splitlines()) > 15:
    out = out.splitlines()[0:15]
    out.append('..trimmed for brevity..')
    out = '\n'.join(out)
print out
```

Output from Python Code

```
Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
2014-12-08 16:24:50,266 INFO     question_progress: Results 0% (Get Folder Name Search with RegEx Ma
2014-12-08 16:24:55,284 INFO     question_progress: Results 0% (Get Folder Name Search with RegEx Ma
2014-12-08 16:25:00,307 INFO     question_progress: Results 17% (Get Folder Name Search with RegEx M
2014-12-08 16:25:05,326 INFO     question_progress: Results 67% (Get Folder Name Search with RegEx M
2014-12-08 16:25:10,344 INFO     question_progress: Results 83% (Get Folder Name Search with RegEx M
2014-12-08 16:25:15,369 INFO     question_progress: Results 100% (Get Folder Name Search with RegEx

Type of response:  <type 'dict'>

Pretty print of response:
{'question_object': <taniumpy.object_types.question.Question object at 0x102967b90>,
 'question_results': <taniumpy.object_types.result_set.ResultSet object at 0x102b2fd50>}

Equivalent Question if it were to be asked in the Tanium Console:
Get Folder Name Search with RegEx Match[No, Program Files, No, ] contains "Shared" from all machines

CSV Results of response:
Count,"Folder Name Search with RegEx Match[No, Program Files, No, ]"
4,[no results]
1,C:\Program Files\Common Files\Microsoft Shared\VS7Debug
2,C:\Program Files\Common Files\Microsoft Shared\ink\ar-SA
2,C:\Program Files\Common Files\Microsoft Shared\ink\ru-RU
2,C:\Program Files\Common Files\Microsoft Shared\ink\fsdefinitions\keypad
2,C:\Program Files\Common Files\Microsoft Shared\ink
2,C:\Program Files\Common Files\Microsoft Shared\ink\sv-SE
2,C:\Program Files\Common Files\Microsoft Shared\ink\uk-UA
2,C:\Program Files\Common Files\Microsoft Shared\ink\sl-SI
2,C:\Program Files\Common Files\Microsoft Shared\ink\hu-HU
```

```
2,C:\Program Files\Common Files\Microsoft Shared\ink\zh-TW
2,C:\Program Files\Common Files\Microsoft Shared\ink\zh-CN
2,C:\Program Files\Common Files\Microsoft Shared\ink\fi-FI
2,C:\Program Files\Common Files\Microsoft Shared
..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

```python
# Path to lib directory which contains pytan package
PYTAN_LIB_PATH = '../lib'

# connection info for Tanium Server
USERNAME = "Tanium User"
PASSWORD = "T@n!um"
HOST = "172.16.31.128"
PORT = "444"

# Logging conrols
LOGLEVEL = 2
DEBUGFORMAT = False

import sys, tempfile
sys.path.append(PYTAN_LIB_PATH)

import pytan
handler = pytan.Handler(
    username=USERNAME,
    password=PASSWORD,
    host=HOST,
    port=PORT,
    loglevel=LOGLEVEL,
    debugformat=DEBUGFORMAT,
)

print handler

# setup the arguments for the handler method
kwargs = {}
kwargs["sensors"] = u'Operating System, that contains:Windows, opt:max_data_age:3600, opt:value_type
kwargs["qtype"] = u'manual_human'

# call the handler with the ask method, passing in kwargs for arguments
response = handler.ask(**kwargs)
import pprint, io
```

```

print ""
print "Type of response: ", type(response)

print ""
print "Pretty print of response:"
print pprint.pformat(response)

print ""
print "Equivalent Question if it were to be asked in the Tanium Console: "
print response['question_object'].query_text

# create an IO stream to store CSV results to
out = io.BytesIO()

# call the write_csv() method to convert response to CSV and store it in out
response['question_results'].write_csv(out, response['question_results'])

print ""
print "CSV Results of response: "
out = out.getvalue()
if len(out.splitlines()) > 15:
    out = out.splitlines()[0:15]
    out.append('..trimmed for brevity..')
    out = '\n'.join(out)
print out
```

Output from Python Code

```
Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
2014-12-08 16:25:15,477 INFO     question_progress: Results 0% (Get Operating System contains "Windo
2014-12-08 16:25:20,494 INFO     question_progress: Results 67% (Get Operating System contains "Wind
2014-12-08 16:25:25,510 INFO     question_progress: Results 100% (Get Operating System contains "Win

Type of response:  <type 'dict'>

Pretty print of response:
{'question_object': <taniumpy.object_types.question.Question object at 0x102b1fbd0>,
 'question_results': <taniumpy.object_types.result_set.ResultSet object at 0x1022e0bd0>}

Equivalent Question if it were to be asked in the Tanium Console:
Get Operating System contains "Windows" from all machines

CSV Results of response:
Count,Operating System
4,[no results]
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

```
# Path to lib directory which contains pytan package
PYTAN_LIB_PATH = '../lib'

# connection info for Tanium Server
USERNAME = "Tanium User"
PASSWORD = "T@n!um"
HOST = "172.16.31.128"
PORT = "444"

# Logging conrols
LOGLEVEL = 2
DEBUGFORMAT = False

import sys, tempfile
sys.path.append(PYTAN_LIB_PATH)

import pytan
handler = pytan.Handler(
    username=USERNAME,
    password=PASSWORD,
    host=HOST,
    port=PORT,
    loglevel=LOGLEVEL,
    debugformat=DEBUGFORMAT,
)

print handler

# setup the arguments for the handler method
kwargs = {}
kwargs["sensors"] = u'Operating System, that contains:Windows'
kwargs["qtype"] = u'manual_human'

# call the handler with the ask method, passing in kwargs for arguments
response = handler.ask(**kwargs)
import pprint, io

print ""
print "Type of response: ", type(response)

print ""
print "Pretty print of response:"
print pprint.pformat(response)

print ""
print "Equivalent Question if it were to be asked in the Tanium Console: "
print response['question_object'].query_text

# create an IO stream to store CSV results to
out = io.BytesIO()

# call the write_csv() method to convert response to CSV and store it in out
response['question_results'].write_csv(out, response['question_results'])

```

```python
print ""
print "CSV Results of response: "
out = out.getvalue()
if len(out.splitlines()) > 15:
    out = out.splitlines()[0:15]
    out.append('..trimmed for brevity..')
    out = '\n'.join(out)
print out
```

Output from Python Code

```
Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
2014-12-08 16:25:25,602 INFO     question_progress: Results 0% (Get Operating System contains "Windo
2014-12-08 16:25:30,621 INFO     question_progress: Results 50% (Get Operating System contains "Wind
2014-12-08 16:25:35,641 INFO     question_progress: Results 100% (Get Operating System contains "Win

Type of response:  <type 'dict'>

Pretty print of response:
{'question_object': <taniumpy.object_types.question.Question object at 0x102123ed0>,
 'question_results': <taniumpy.object_types.result_set.ResultSet object at 0x10218fc50>}

Equivalent Question if it were to be asked in the Tanium Console:
Get Operating System contains "Windows" from all machines

CSV Results of response:
Count,Operating System
4,[no results]
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

```python
# Path to lib directory which contains pytan package
PYTAN_LIB_PATH = '../lib'

# connection info for Tanium Server
USERNAME = "Tanium User"
PASSWORD = "T@n!um"
HOST = "172.16.31.128"
PORT = "444"

```

```
# Logging conrols
LOGLEVEL = 2
DEBUGFORMAT = False

import sys, tempfile
sys.path.append(PYTAN_LIB_PATH)

import pytan
handler = pytan.Handler(
    username=USERNAME,
    password=PASSWORD,
    host=HOST,
    port=PORT,
    loglevel=LOGLEVEL,
    debugformat=DEBUGFORMAT,
)

print handler

# setup the arguments for the handler method
kwargs = {}
kwargs["question_filters"] = [u'Operating System, that contains:Windows',
 u'Operating System, that does not contain:Windows']
kwargs["sensors"] = [u'Computer Name',
 u'Folder Name Search with RegEx Match{dirname=Program Files,regex=Microsoft.*, invalidparam=test},
kwargs["question_options"] = [u'ignore_case', u'or']
kwargs["qtype"] = u'manual_human'

# call the handler with the ask method, passing in kwargs for arguments
response = handler.ask(**kwargs)
import pprint, io

print ""
print "Type of response: ", type(response)

print ""
print "Pretty print of response:"
print pprint.pformat(response)

print ""
print "Equivalent Question if it were to be asked in the Tanium Console: "
print response['question_object'].query_text

# create an IO stream to store CSV results to
out = io.BytesIO()

# call the write_csv() method to convert response to CSV and store it in out
response['question_results'].write_csv(out, response['question_results'])

print ""
print "CSV Results of response: "
out = out.getvalue()
if len(out.splitlines()) > 15:
    out = out.splitlines()[0:15]
    out.append('..trimmed for brevity..')
    out = '\n'.join(out)
print out
```

Output from Python Code

```
Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
2014-12-08 16:25:35,790 INFO     question_progress: Results 0% (Get Computer Name and Folder Name Se
2014-12-08 16:25:40,818 INFO     question_progress: Results 0% (Get Computer Name and Folder Name Se
2014-12-08 16:25:45,841 INFO     question_progress: Results 17% (Get Computer Name and Folder Name S
2014-12-08 16:25:50,868 INFO     question_progress: Results 50% (Get Computer Name and Folder Name S
2014-12-08 16:25:55,894 INFO     question_progress: Results 67% (Get Computer Name and Folder Name S
2014-12-08 16:26:00,919 INFO     question_progress: Results 100% (Get Computer Name and Folder Name

Type of response:  <type 'dict'>

Pretty print of response:
{'question_object': <taniumpy.object_types.question.Question object at 0x102967a10>,
 'question_results': <taniumpy.object_types.result_set.ResultSet object at 0x1022e0490>}

Equivalent Question if it were to be asked in the Tanium Console:
Get Computer Name and Folder Name Search with RegEx Match[No, Program Files, No, ] contains "Shared"

CSV Results of response:
Computer Name,"Folder Name Search with RegEx Match[No, Program Files, No, ]"
Casus-Belli.local,[no results]
ubuntu.(none),[no results]
localhost.(none),[no results]
Jims-Mac.local,[no results]
jtanium1.localdomain,"C:\Program Files\Common Files\Microsoft Shared\VS7Debug
C:\Program Files\Common Files\Microsoft Shared\ink\ar-SA
C:\Program Files\Common Files\Microsoft Shared\ink\ru-RU
C:\Program Files\Common Files\Microsoft Shared\ink\fsdefinitions\keypad
C:\Program Files\Common Files\Microsoft Shared\ink
C:\Program Files\Common Files\Microsoft Shared\ink\sv-SE
C:\Program Files\Common Files\Microsoft Shared\ink\uk-UA
C:\Program Files\Common Files\Microsoft Shared\ink\sl-SI
C:\Program Files\Common Files\Microsoft Shared\ink\hu-HU
C:\Program Files\Common Files\Microsoft Shared\ink\zh-TW
..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

```
# Path to lib directory which contains pytan package
PYTAN_LIB_PATH = '../lib'

# connection info for Tanium Server
USERNAME = "Tanium User"
PASSWORD = "T@n!um"
HOST = "172.16.31.128"
PORT = "444"

# Logging conrols
LOGLEVEL = 2
```

```
DEBUGFORMAT = False

import sys, tempfile
sys.path.append(PYTAN_LIB_PATH)

import pytan
handler = pytan.Handler(
    username=USERNAME,
    password=PASSWORD,
    host=HOST,
    port=PORT,
    loglevel=LOGLEVEL,
    debugformat=DEBUGFORMAT,
)

print handler

# setup the arguments for the handler method
kwargs = {}
kwargs["question_filters"] = [u'Installed Applications, that contains:Google Search',
 u'Installed Applications, that contains:Google Chrome']
kwargs["sensors"] = [u'Computer Name',
 u'Last Logged In User',
 u'Installed Applications, that contains:Google Search',
 u'Installed Applications, that contains:Google Chrome']
kwargs["question_options"] = [u'ignore_case', u'and']
kwargs["qtype"] = u'manual_human'

# call the handler with the ask method, passing in kwargs for arguments
response = handler.ask(**kwargs)
import pprint, io

print ""
print "Type of response: ", type(response)

print ""
print "Pretty print of response:"
print pprint.pformat(response)

print ""
print "Equivalent Question if it were to be asked in the Tanium Console: "
print response['question_object'].query_text

# create an IO stream to store CSV results to
out = io.BytesIO()

# call the write_csv() method to convert response to CSV and store it in out
response['question_results'].write_csv(out, response['question_results'])

print ""
print "CSV Results of response: "
out = out.getvalue()
if len(out.splitlines()) > 15:
    out = out.splitlines()[0:15]
    out.append('..trimmed for brevity..')
    out = '\n'.join(out)
print out
```

Output from Python Code

```
Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
2014-12-08 16:26:01,111 INFO     question_progress: Results 0% (Get Computer Name and Last Logged In
2014-12-08 16:26:06,148 INFO     question_progress: Results 67% (Get Computer Name and Last Logged I
2014-12-08 16:26:11,188 INFO     question_progress: Results 100% (Get Computer Name and Last Logged 

Type of response:  <type 'dict'>

Pretty print of response:
{'question_object': <taniumpy.object_types.question.Question object at 0x102b0c790>,
 'question_results': <taniumpy.object_types.result_set.ResultSet object at 0x1029cf510>}

Equivalent Question if it were to be asked in the Tanium Console:
Get Computer Name and Last Logged In User and Installed Applications contains "Google Search" and In

CSV Results of response:
Computer Name,Last Logged In User,Name,Name,Silent Uninstall String,Silent Uninstall String,Uninstal
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

```python
# Path to lib directory which contains pytan package
PYTAN_LIB_PATH = '../lib'

# connection info for Tanium Server
USERNAME = "Tanium User"
PASSWORD = "T@n!um"
HOST = "172.16.31.128"
PORT = "444"

# Logging conrols
LOGLEVEL = 2
DEBUGFORMAT = False

import sys, tempfile
sys.path.append(PYTAN_LIB_PATH)

import pytan
handler = pytan.Handler(
    username=USERNAME,
    password=PASSWORD,
    host=HOST,
    port=PORT,
    loglevel=LOGLEVEL,
    debugformat=DEBUGFORMAT,
)

print handler

# setup the arguments for the handler method
kwargs = {}
kwargs["question_filter_defs"] = [{u'filter': {u'not_flag': 0,
              u'operator': u'RegexMatch',
              u'value': u'.*Windows.*'},
  u'name': u'Operating System'}]
kwargs["sensor_defs"] = [u'Computer Name',
 {u'filter': {u'not_flag': 0,
              u'operator': u'RegexMatch',
              u'value': u'.*Shared.*'},
  u'name': u'Folder Name Search with RegEx Match',
  u'options': {u'ignore_case_flag': 0,
               u'max_age_seconds': 3600,
               u'value_type': u'string'},
  u'params': {u'dirname': u'Program Files'}}]
kwargs["question_option_defs"] = {u'and_flag': 0, u'ignore_case_flag': 0, u'max_age_seconds': 3600}
kwargs["qtype"] = u'manual'

# call the handler with the ask method, passing in kwargs for arguments
response = handler.ask(**kwargs)
import pprint, io

print ""
print "Type of response: ", type(response)

print ""
print "Pretty print of response:"
print pprint.pformat(response)

print ""
print "Equivalent Question if it were to be asked in the Tanium Console: "
print response['question_object'].query_text

# create an IO stream to store CSV results to
out = io.BytesIO()

# call the write_csv() method to convert response to CSV and store it in out
response['question_results'].write_csv(out, response['question_results'])

print ""
print "CSV Results of response: "
out = out.getvalue()
if len(out.splitlines()) > 15:
    out = out.splitlines()[0:15]
    out.append('..trimmed for brevity..')
    out = '\n'.join(out)
print out
```

Output from Python Code

```
Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
2014-12-08 16:26:11,367 INFO     question_progress: Results 0% (Get Computer Name and Folder Name Se
2014-12-08 16:26:16,396 INFO     question_progress: Results 33% (Get Computer Name and Folder Name S
2014-12-08 16:26:21,433 INFO     question_progress: Results 67% (Get Computer Name and Folder Name S
2014-12-08 16:26:26,459 INFO     question_progress: Results 67% (Get Computer Name and Folder Name S
2014-12-08 16:26:31,481 INFO     question_progress: Results 83% (Get Computer Name and Folder Name S
2014-12-08 16:26:36,503 INFO     question_progress: Results 100% (Get Computer Name and Folder Name

Type of response:  <type 'dict'>

Pretty print of response:
{'question_object': <taniumpy.object_types.question.Question object at 0x1029571d0>,
 'question_results': <taniumpy.object_types.result_set.ResultSet object at 0x102116710>}

Equivalent Question if it were to be asked in the Tanium Console:
Get Computer Name and Folder Name Search with RegEx Match[No, Program Files, No, ] contains "Shared"

CSV Results of response:
Computer Name,"Folder Name Search with RegEx Match[No, Program Files, No, ]"
jtanium1.localdomain,"C:\Program Files\Common Files\Microsoft Shared\VS7Debug
C:\Program Files\Common Files\Microsoft Shared\ink\ar-SA
C:\Program Files\Common Files\Microsoft Shared\ink\ru-RU
C:\Program Files\Common Files\Microsoft Shared\ink\fsdefinitions\keypad
C:\Program Files\Common Files\Microsoft Shared\ink
C:\Program Files\Common Files\Microsoft Shared\ink\sv-SE
C:\Program Files\Common Files\Microsoft Shared\ink\uk-UA
C:\Program Files\Common Files\Microsoft Shared\ink\sl-SI
C:\Program Files\Common Files\Microsoft Shared\ink\hu-HU
C:\Program Files\Common Files\Microsoft Shared\ink\zh-TW
C:\Program Files\Common Files\Microsoft Shared\ink\zh-CN
C:\Program Files\Common Files\Microsoft Shared\ink\fi-FI
C:\Program Files\Common Files\Microsoft Shared
C:\Program Files\Common Files\Microsoft Shared\ink\da-DK
..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

```python
# Path to lib directory which contains pytan package
PYTAN_LIB_PATH = '../lib'

# connection info for Tanium Server
USERNAME = "Tanium User"
PASSWORD = "T@n!um"
HOST = "172.16.31.128"
PORT = "444"

# Logging conrols
LOGLEVEL = 2
DEBUGFORMAT = False
```

```

import sys, tempfile
sys.path.append(PYTAN_LIB_PATH)

import pytan
handler = pytan.Handler(
    username=USERNAME,
    password=PASSWORD,
    host=HOST,
    port=PORT,
    loglevel=LOGLEVEL,
    debugformat=DEBUGFORMAT,
)

print handler

# setup the arguments for the handler method
kwargs = {}
kwargs["filters_help"] = True
kwargs["qtype"] = u'manual_human'


# call the handler with the ask method, passing in kwargs for arguments
# this should throw an exception: pytan.utils.PytanHelp
import traceback
try:
    handler.ask(**kwargs)
except Exception as e:
    traceback.print_exc(file=sys.stdout)
```

Output from Python Code

```
Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
Traceback (most recent call last):
  File "<string>", line 39, in <module>
  File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 128, in ask
    result = getattr(self, q_obj_map['handler'])(**kwargs)
  File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 379, in ask_manual_human
    raise PytanHelp(utils.help_filters())
PytanHelp:
Filters Help
============

Filters are used generously throughout pytan. When used as part of a
sensor string, they control what data is shown for the columns that
the sensor returns. When filters are used for whole question filters,
they control what rows will be returned. They are used by Groups to
define group membership, deploy actions to determine which machines
should have the action deployed to it, and more.

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

Valid Filters
-------------

```

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

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

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

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

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

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

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

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

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

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

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

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

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

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

    'not lessequal'
        Help: Filter for not less than or equal to VALUE
```

```
            Example: "Sensor1, that not lessequal:VALUE"

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

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

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

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

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

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

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

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

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

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

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

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

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

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

```

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

    '='
        Help: Filter for equals to VALUE
        Example: "Sensor1, that =:VALUE"

    'equal'
        Help: Filter for equals to VALUE
        Example: "Sensor1, that equal:VALUE"

    'equals'
        Help: Filter for equals to VALUE
        Example: "Sensor1, that equals:VALUE"

    'eq'
        Help: Filter for equals to VALUE
        Example: "Sensor1, that eq:VALUE"

    '!='
        Help: Filter for not equals to VALUE
        Example: "Sensor1, that !=:VALUE"

    'not equal'
        Help: Filter for not equals to VALUE
        Example: "Sensor1, that not equal:VALUE"

    'notequal'
        Help: Filter for not equals to VALUE
        Example: "Sensor1, that notequal:VALUE"

    'not equals'
        Help: Filter for not equals to VALUE
        Example: "Sensor1, that not equals:VALUE"

    'notequals'
        Help: Filter for not equals to VALUE
        Example: "Sensor1, that notequals:VALUE"

    'ne'
        Help: Filter for not equals to VALUE
        Example: "Sensor1, that ne:VALUE"

    'contains'
        Help: Filter for contains VALUE (adds .* before and after VALUE)
        Example: "Sensor1, that contains:VALUE"

    'does not contain'
        Help: Filter for does not contain VALUE (adds .* before and after VALUE)
        Example: "Sensor1, that does not contain:VALUE"

    'doesnotcontain'
        Help: Filter for does not contain VALUE (adds .* before and after VALUE)
        Example: "Sensor1, that doesnotcontain:VALUE"

    'not contains'
        Help: Filter for does not contain VALUE (adds .* before and after VALUE)
        Example: "Sensor1, that not contains:VALUE"

    'notcontains'
        Help: Filter for does not contain VALUE (adds .* before and after VALUE)
        Example: "Sensor1, that notcontains:VALUE"

    'starts with'
        Help: Filter for starts with VALUE (adds .* after VALUE)
        Example: "Sensor1, that starts with:VALUE"

    'startswith'
        Help: Filter for starts with VALUE (adds .* after VALUE)
        Example: "Sensor1, that startswith:VALUE"

    'does not start with'
        Help: Filter for does not start with VALUE (adds .* after VALUE)
        Example: "Sensor1, that does not start with:VALUE"

    'doesnotstartwith'
        Help: Filter for does not start with VALUE (adds .* after VALUE)
        Example: "Sensor1, that doesnotstartwith:VALUE"

    'not starts with'
        Help: Filter for does not start with VALUE (adds .* after VALUE)
        Example: "Sensor1, that not starts with:VALUE"

    'notstartswith'
        Help: Filter for does not start with VALUE (adds .* after VALUE)
        Example: "Sensor1, that notstartswith:VALUE"

    'ends with'
        Help: Filter for ends with VALUE (adds .* before VALUE)
        Example: "Sensor1, that ends with:VALUE"

    'endswith'
        Help: Filter for ends with VALUE (adds .* before VALUE)
        Example: "Sensor1, that endswith:VALUE"

    'does not end with'
        Help: Filter for does bit end with VALUE (adds .* before VALUE)
        Example: "Sensor1, that does not end with:VALUE"

    'doesnotendwith'
        Help: Filter for does bit end with VALUE (adds .* before VALUE)
        Example: "Sensor1, that doesnotendwith:VALUE"

    'not ends with'
        Help: Filter for does bit end with VALUE (adds .* before VALUE)
        Example: "Sensor1, that not ends with:VALUE"

    'notstartswith'
        Help: Filter for does bit end with VALUE (adds .* before VALUE)
        Example: "Sensor1, that notstartswith:VALUE"

    'is not'
        Help: Filter for non regular expression match for VALUE
        Example: "Sensor1, that is not:VALUE"

    'not regex'
        Help: Filter for non regular expression match for VALUE
        Example: "Sensor1, that not regex:VALUE"

    'notregex'
        Help: Filter for non regular expression match for VALUE
        Example: "Sensor1, that notregex:VALUE"

    'not regex match'
        Help: Filter for non regular expression match for VALUE
        Example: "Sensor1, that not regex match:VALUE"

    'notregexmatch'
        Help: Filter for non regular expression match for VALUE
        Example: "Sensor1, that notregexmatch:VALUE"

    'nre'
        Help: Filter for non regular expression match for VALUE
        Example: "Sensor1, that nre:VALUE"

    'is'
        Help: Filter for regular expression match for VALUE
        Example: "Sensor1, that is:VALUE"

    'regex'
        Help: Filter for regular expression match for VALUE
        Example: "Sensor1, that regex:VALUE"

    'regex match'
        Help: Filter for regular expression match for VALUE
        Example: "Sensor1, that regex match:VALUE"

    'regexmatch'
        Help: Filter for regular expression match for VALUE
        Example: "Sensor1, that regexmatch:VALUE"

    're'
        Help: Filter for regular expression match for VALUE
        Example: "Sensor1, that re:VALUE"
```

Invalid ask manual human question option help

Have ask_manual_human() return the help for options

Example Python Code

```python
# Path to lib directory which contains pytan package
PYTAN_LIB_PATH = '../lib'

# connection info for Tanium Server
USERNAME = "Tanium User"
PASSWORD = "T@n!um"
HOST = "172.16.31.128"
PORT = "444"

```

```python
# Logging conrols
LOGLEVEL = 2
DEBUGFORMAT = False

import sys, tempfile
sys.path.append(PYTAN_LIB_PATH)

import pytan
handler = pytan.Handler(
    username=USERNAME,
    password=PASSWORD,
    host=HOST,
    port=PORT,
    loglevel=LOGLEVEL,
    debugformat=DEBUGFORMAT,
)

print handler

# setup the arguments for the handler method
kwargs = {}
kwargs["options_help"] = True
kwargs["qtype"] = u'manual_human'


# call the handler with the ask method, passing in kwargs for arguments
# this should throw an exception: pytan.utils.PytanHelp
import traceback
try:
    handler.ask(**kwargs)
except Exception as e:
    traceback.print_exc(file=sys.stdout)
```

Output from Python Code

```
Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
Traceback (most recent call last):
  File "<string>", line 39, in <module>
  File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 128, in ask
    result = getattr(self, q_obj_map['handler'])(**kwargs)
  File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 382, in ask_manual_human
    raise PytanHelp(utils.help_options())
PytanHelp:
Options Help
============

Options are used for controlling how filters act. When options are
used as part of a sensor string, they change how the filters
supplied as part of that sensor operate. When options are used for
whole question options, they change how all of the question filters
operate.

When options are supplied for a sensor string, they must be
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'
        Help: Use 'or' for all of the filters supplied
        Usable on: group
        Example for sensor: "Sensor1, opt:or"
        Example for question: "or"
```

Invalid ask manual human question sensor help

Have ask_manual_human() return the help for sensors

Example Python Code

```python
# Path to lib directory which contains pytan package
PYTAN_LIB_PATH = '../lib'

# connection info for Tanium Server
USERNAME = "Tanium User"
PASSWORD = "T@n!um"
HOST = "172.16.31.128"
PORT = "444"

# Logging conrols
LOGLEVEL = 2
DEBUGFORMAT = False

import sys, tempfile
sys.path.append(PYTAN_LIB_PATH)

import pytan
handler = pytan.Handler(
    username=USERNAME,
    password=PASSWORD,
    host=HOST,
    port=PORT,
    loglevel=LOGLEVEL,
    debugformat=DEBUGFORMAT,
)

print handler

# setup the arguments for the handler method
kwargs = {}
kwargs["qtype"] = u'manual_human'
kwargs["sensors_help"] = True


# call the handler with the ask method, passing in kwargs for arguments
# this should throw an exception: pytan.utils.PytanHelp
import traceback
try:
    handler.ask(**kwargs)
except Exception as e:
    traceback.print_exc(file=sys.stdout)
```

Output from Python Code

```
Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
Traceback (most recent call last):
  File "<string>", line 39, in <module>
  File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 128, in ask
    result = getattr(self, q_obj_map['handler'])(**kwargs)
  File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 376, in ask_manual_human
    raise PytanHelp(utils.help_sensors())
PytanHelp:
Sensors Help
============

Supplying sensors controls what columns will be showed when you ask a
question.

A sensor string is a human string that describes, at a minimum, a sensor.
It can also optionally define a selector for the sensor, parameters for
the sensor, a filter for the sensor, and options for the filter for the
sensor. Sensors can be provided as a string or a list of strings.

Examples for basic sensors
---------------------------------

Supplying a single sensor:

    'Computer Name'

Supplying two sensors in a list of strings:

    ['Computer Name', 'IP Route Details']

Supplying multiple sensors with selectors (name is the default
selector if none is supplied):

    [
        'Computer Name',
        'name:Computer Name',
        'id:1',
        'hash:123456789',
    ]

Sensor Parameters
-----------------

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

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

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

    * if the parameter does not require a default value, the
    parameter is left blank and not supplied.
    * 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
```

```
that sensor works.

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

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

Examples for sensors with options
---------------------------------

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

    'Computer Name, opt:max_data_age:60'

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

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

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

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

Supplying a sensor with a filter and a set of options that
causes Tanium to recognize the value type as String (which is
the default type for most sensors), re-fetch data older than
10 minutes, match any values, and match case:

    'Computer Name', that contains:Windows, ' \
    opt:value_type:string, opt:max_data_age:600, ' \
    'opt:match_any_value, opt:match_case'
```

Invalid ask manual human question filter

Ask a question using an invalid filter.

Example Python Code

```python
# Path to lib directory which contains pytan package
PYTAN_LIB_PATH = '../lib'

# connection info for Tanium Server
USERNAME = "Tanium User"
PASSWORD = "T@n!um"
HOST = "172.16.31.128"
PORT = "444"

# Logging conrols
LOGLEVEL = 2
DEBUGFORMAT = False
```

```python

import sys, tempfile
sys.path.append(PYTAN_LIB_PATH)

import pytan
handler = pytan.Handler(
    username=USERNAME,
    password=PASSWORD,
    host=HOST,
    port=PORT,
    loglevel=LOGLEVEL,
    debugformat=DEBUGFORMAT,
)

print handler

# setup the arguments for the handler method
kwargs = {}
kwargs["sensors"] = u'Computer name, that does not meet:little'
kwargs["qtype"] = u'manual_human'


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

Output from Python Code

```
Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
Traceback (most recent call last):
  File "<string>", line 39, in <module>
  File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 128, in ask
    result = getattr(self, q_obj_map['handler'])(**kwargs)
  File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 399, in ask_manual_human
    sensor_defs = utils.dehumanize_sensors(sensors)
  File "/Users/jolsen/gh/pytan/lib/pytan/utils.py", line 1304, in dehumanize_sensors
    s, parsed_filter = extract_filter(s)
  File "/Users/jolsen/gh/pytan/lib/pytan/utils.py", line 1666, in extract_filter
    raise HumanParserError(err(split_filter[1]))
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

```python
# Path to lib directory which contains pytan package
PYTAN_LIB_PATH = '../lib'

```

```python
# connection info for Tanium Server
USERNAME = "Tanium User"
PASSWORD = "T@n!um"
HOST = "172.16.31.128"
PORT = "444"

# Logging conrols
LOGLEVEL = 2
DEBUGFORMAT = False

import sys, tempfile
sys.path.append(PYTAN_LIB_PATH)

import pytan
handler = pytan.Handler(
    username=USERNAME,
    password=PASSWORD,
    host=HOST,
    port=PORT,
    loglevel=LOGLEVEL,
    debugformat=DEBUGFORMAT,
)

print handler

# setup the arguments for the handler method
kwargs = {}
kwargs["sensor_defs"] = u'Dweedle Dee and Dum'
kwargs["qtype"] = u'manual'


# call the handler with the ask method, passing in kwargs for arguments
# this should throw an exception: pytan.utils.HandlerError
import traceback
try:
    handler.ask(**kwargs)
except Exception as e:
    traceback.print_exc(file=sys.stdout)
```

Output from Python Code

```
Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
Traceback (most recent call last):
  File "<string>", line 39, in <module>
  File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 128, in ask
    result = getattr(self, q_obj_map['handler'])(**kwargs)
  File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 271, in ask_manual
    sensor_defs = self._get_sensor_defs(sensor_defs)
  File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 1784, in _get_sensor_defs
    d['sensor_obj'] = self.get('sensor', **def_search)[0]
  File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 1567, in get
    return self._get_multi(obj_map, **kwargs)
  File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 1729, in _get_multi
    found = self._find(api_obj_multi, **kwargs)
  File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 1694, in _find
    raise HandlerError(err(search_str))
```

```
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

```python
# Path to lib directory which contains pytan package
PYTAN_LIB_PATH = '../lib'

# connection info for Tanium Server
USERNAME = "Tanium User"
PASSWORD = "T@n!um"
HOST = "172.16.31.128"
PORT = "444"

# Logging conrols
LOGLEVEL = 2
DEBUGFORMAT = False

import sys, tempfile
sys.path.append(PYTAN_LIB_PATH)

import pytan
handler = pytan.Handler(
    username=USERNAME,
    password=PASSWORD,
    host=HOST,
    port=PORT,
    loglevel=LOGLEVEL,
    debugformat=DEBUGFORMAT,
)

print handler

# setup the arguments for the handler method
kwargs = {}
kwargs["sensors"] = u'Folder Name Search with RegEx Match{dirname=Program Files,regex=.*}{}'
kwargs["qtype"] = u'manual_human'


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

Output from Python Code

```
Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
Traceback (most recent call last):
  File "<string>", line 39, in <module>
  File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 128, in ask
    result = getattr(self, q_obj_map['handler'])(**kwargs)
  File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 399, in ask_manual_human
    sensor_defs = utils.dehumanize_sensors(sensors)
  File "/Users/jolsen/gh/pytan/lib/pytan/utils.py", line 1302, in dehumanize_sensors
    s, parsed_params = extract_params(s)
  File "/Users/jolsen/gh/pytan/lib/pytan/utils.py", line 1472, in extract_params
    raise HumanParserError(err(s))
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

```python
# Path to lib directory which contains pytan package
PYTAN_LIB_PATH = '../lib'

# connection info for Tanium Server
USERNAME = "Tanium User"
PASSWORD = "T@n!um"
HOST = "172.16.31.128"
PORT = "444"

# Logging conrols
LOGLEVEL = 2
DEBUGFORMAT = False

import sys, tempfile
sys.path.append(PYTAN_LIB_PATH)

import pytan
handler = pytan.Handler(
    username=USERNAME,
    password=PASSWORD,
    host=HOST,
    port=PORT,
    loglevel=LOGLEVEL,
    debugformat=DEBUGFORMAT,
)

print handler

# setup the arguments for the handler method
kwargs = {}
kwargs["sensors"] = u'Operating system, opt:bad'
kwargs["qtype"] = u'manual_human'


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

Output from Python Code

```
Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
Traceback (most recent call last):
  File "<string>", line 39, in <module>
  File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 128, in ask
    result = getattr(self, q_obj_map['handler'])(**kwargs)
  File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 399, in ask_manual_human
    sensor_defs = utils.dehumanize_sensors(sensors)
  File "/Users/jolsen/gh/pytan/lib/pytan/utils.py", line 1303, in dehumanize_sensors
    s, parsed_options = extract_options(s)
  File "/Users/jolsen/gh/pytan/lib/pytan/utils.py", line 1541, in extract_options
    parsed_options = map_options(parsed_options, ['filter'])
  File "/Users/jolsen/gh/pytan/lib/pytan/utils.py", line 1571, in map_options
    raise HumanParserError(err(option))
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

```
# Path to lib directory which contains pytan package
PYTAN_LIB_PATH = '../lib'

# connection info for Tanium Server
USERNAME = "Tanium User"
PASSWORD = "T@n!um"
HOST = "172.16.31.128"
PORT = "444"

# Logging conrols
LOGLEVEL = 2
DEBUGFORMAT = False

import sys, tempfile
sys.path.append(PYTAN_LIB_PATH)

import pytan
handler = pytan.Handler(
    username=USERNAME,
    password=PASSWORD,
    host=HOST,
    port=PORT,
    loglevel=LOGLEVEL,
    debugformat=DEBUGFORMAT,
)

print handler
```

```python

# setup the arguments for the handler method
kwargs = {}
kwargs["sensors"] = u'Computer Name{Dweedle}'
kwargs["qtype"] = u'manual_human'


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

Output from Python Code

```
Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
Traceback (most recent call last):
  File "<string>", line 39, in <module>
  File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 128, in ask
    result = getattr(self, q_obj_map['handler'])(**kwargs)
  File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 399, in ask_manual_human
    sensor_defs = utils.dehumanize_sensors(sensors)
  File "/Users/jolsen/gh/pytan/lib/pytan/utils.py", line 1302, in dehumanize_sensors
    s, parsed_params = extract_params(s)
  File "/Users/jolsen/gh/pytan/lib/pytan/utils.py", line 1490, in extract_params
    raise HumanParserError(err(sp, constants.PARAM_KEY_SPLIT))
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

```python
# Path to lib directory which contains pytan package
PYTAN_LIB_PATH = '../lib'

# connection info for Tanium Server
USERNAME = "Tanium User"
PASSWORD = "T@n!um"
HOST = "172.16.31.128"
PORT = "444"

# Logging conrols
LOGLEVEL = 2
DEBUGFORMAT = False

import sys, tempfile
sys.path.append(PYTAN_LIB_PATH)

```

```python
import pytan
handler = pytan.Handler(
    username=USERNAME,
    password=PASSWORD,
    host=HOST,
    port=PORT,
    loglevel=LOGLEVEL,
    debugformat=DEBUGFORMAT,
)

print handler

# setup the arguments for the handler method
kwargs = {}
kwargs["objtype"] = u'action'
kwargs["id"] = 1

# call the handler with the get method, passing in kwargs for arguments
response = handler.get(**kwargs)

print ""
print "Type of response: ", type(response)

print ""
print "print of response:"
print response

print ""
print "length of response (number of objects returned): "
print len(response)

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

print out
```

Output from Python Code

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

Type of response:  <class 'taniumpy.object_types.action_list.ActionList'>

print of response:
ActionList, len: 1

length of response (number of objects returned):
1

print the first object returned in JSON format:
{
  "_type": "action",
  "action_group": {
    "_type": "group",
    "id": 0,
    "name": "Default"
  },
  "comment": "Scans for unmanaged assets on the network.",
  "creation_time": "2014-12-08T19:22:33",
  "distribute_seconds": 600,
  "expire_seconds": 1800,
  "history_saved_question": {
    "_type": "saved_question",
    "id": 173
  },
..trimmed for brevity..
```

Get question by id

Get a question by id

Example Python Code

```python
# Path to lib directory which contains pytan package
PYTAN_LIB_PATH = '../lib'

# connection info for Tanium Server
USERNAME = "Tanium User"
PASSWORD = "T@n!um"
HOST = "172.16.31.128"
PORT = "444"

# Logging conrols
LOGLEVEL = 2
DEBUGFORMAT = False

import sys, tempfile
sys.path.append(PYTAN_LIB_PATH)

import pytan
handler = pytan.Handler(
    username=USERNAME,
    password=PASSWORD,
    host=HOST,
    port=PORT,
    loglevel=LOGLEVEL,
    debugformat=DEBUGFORMAT,
)

print handler

# setup the arguments for the handler method
kwargs = {}
kwargs["objtype"] = u'question'
kwargs["id"] = 1

# call the handler with the get method, passing in kwargs for arguments
response = handler.get(**kwargs)
```

```

print ""
print "Type of response: ", type(response)

print ""
print "print of response:"
print response

print ""
print "length of response (number of objects returned): "
print len(response)

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

print out
```

Output from Python Code

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

Type of response:  <class 'taniumpy.object_types.question_list.QuestionList'>

print of response:
QuestionList, len: 1

length of response (number of objects returned):
1

print the first object returned in JSON format:
{
  "_type": "question",
  "action_tracking_flag": 0,
  "context_group": {
    "_type": "group",
    "id": 0
  },
  "expiration": "2014-12-08T19:30:12",
  "expire_seconds": 0,
  "force_computer_id_flag": 1,
  "hidden_flag": 0,
  "id": 1,
  "management_rights_group": {
    "_type": "group",
    "id": 0
..trimmed for brevity..
```

Get saved question by names

Get two saved questions by name

Example Python Code

```
# Path to lib directory which contains pytan package
PYTAN_LIB_PATH = '../lib'

# connection info for Tanium Server
USERNAME = "Tanium User"
PASSWORD = "T@n!um"
HOST = "172.16.31.128"
PORT = "444"

# Logging conrols
LOGLEVEL = 2
DEBUGFORMAT = False

import sys, tempfile
sys.path.append(PYTAN_LIB_PATH)

import pytan
handler = pytan.Handler(
    username=USERNAME,
    password=PASSWORD,
    host=HOST,
    port=PORT,
    loglevel=LOGLEVEL,
    debugformat=DEBUGFORMAT,
)

print handler

# setup the arguments for the handler method
kwargs = {}
kwargs["objtype"] = u'saved_question'
kwargs["name"] = [u'Installed Applications', u'Computer Name']

# call the handler with the get method, passing in kwargs for arguments
response = handler.get(**kwargs)

print ""
print "Type of response: ", type(response)

print ""
print "print of response:"
print response

print ""
print "length of response (number of objects returned): "
print len(response)

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

print out
```

Output from Python Code

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

Type of response:  <class 'taniumpy.object_types.saved_question_list.SavedQuestionList'>

print of response:
SavedQuestionList, len: 2

length of response (number of objects returned):
2

print the first object returned in JSON format:
{
  "_type": "saved_question",
  "action_tracking_flag": 0,
  "archive_enabled_flag": 0,
  "archive_owner": {
    "_type": "user",
    "id": 1,
    "name": "Jim Olsen"
  },
  "expire_seconds": 600,
  "hidden_flag": 0,
  "id": 92,
  "issue_seconds": 120,
  "issue_seconds_never_flag": 0,
  "keep_seconds": 3600,
..trimmed for brevity..
```

Get userrole by id

Get a user role by id.

Example Python Code

```
# Path to lib directory which contains pytan package
PYTAN_LIB_PATH = '../lib'

# connection info for Tanium Server
USERNAME = "Tanium User"
PASSWORD = "T@n!um"
HOST = "172.16.31.128"
PORT = "444"

# Logging conrols
LOGLEVEL = 2
DEBUGFORMAT = False

import sys, tempfile
sys.path.append(PYTAN_LIB_PATH)

import pytan
handler = pytan.Handler(
    username=USERNAME,
    password=PASSWORD,
    host=HOST,
    port=PORT,
    loglevel=LOGLEVEL,
    debugformat=DEBUGFORMAT,
)

print handler

# setup the arguments for the handler method
kwargs = {}
kwargs["objtype"] = u'userrole'
kwargs["id"] = 1

# call the handler with the get method, passing in kwargs for arguments
response = handler.get(**kwargs)

print ""
print "Type of response: ", type(response)

print ""
print "print of response:"
print response

print ""
print "length of response (number of objects returned): "
print len(response)

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

print out
```

Output from Python Code

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

Type of response:  <class 'taniumpy.object_types.user_role_list.UserRoleList'>

print of response:
UserRoleList, len: 1

length of response (number of objects returned):
1

print the first object returned in JSON format:
{
  "_type": "role",
  "description": "Administrators can perform all functions in the system, including creating other u
  "id": 1,
  "name": "Administrator",
  "permissions": {
    "_type": "permissions",
    "permission": "admin"
  }
}
```

Get leader clients

Get all clients that are Leader status

Example Python Code

```
# Path to lib directory which contains pytan package
PYTAN_LIB_PATH = '../lib'

# connection info for Tanium Server
USERNAME = "Tanium User"
PASSWORD = "T@n!um"
HOST = "172.16.31.128"
PORT = "444"

# Logging conrols
LOGLEVEL = 2
DEBUGFORMAT = False

import sys, tempfile
sys.path.append(PYTAN_LIB_PATH)

import pytan
handler = pytan.Handler(
    username=USERNAME,
    password=PASSWORD,
    host=HOST,
    port=PORT,
    loglevel=LOGLEVEL,
    debugformat=DEBUGFORMAT,
)

print handler

# setup the arguments for the handler method
kwargs = {}
kwargs["objtype"] = u'client'
kwargs["status"] = u'Leader'

# call the handler with the get method, passing in kwargs for arguments
response = handler.get(**kwargs)

print ""
print "Type of response: ", type(response)

print ""
print "print of response:"
print response

print ""
print "length of response (number of objects returned): "
print len(response)

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

print out
```

Output from Python Code

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

Type of response:  <class 'taniumpy.object_types.system_status_list.SystemStatusList'>

print of response:
SystemStatusList, len: 1

length of response (number of objects returned):
1

print the first object returned in JSON format:
{
  "_type": "client_status",
  "cache_row_id": 2,
  "computer_id": "3508795802",
  "full_version": "6.0.314.1190",
  "host_name": "WIN-A12SC6N6T7Q",
  "ipaddress_client": "172.16.31.145",
  "ipaddress_server": "172.16.31.145",
  "last_registration": "2014-12-08T21:26:21",
  "port_number": 17472,
  "protocol_version": 314,
  "receive_state": "Previous Only",
  "send_state": "Backward Only",
  "status": "Leader"
}
```

Get setting by name

Get a system setting by name

Example Python Code

```
# Path to lib directory which contains pytan package
PYTAN_LIB_PATH = '../lib'

# connection info for Tanium Server
USERNAME = "Tanium User"
PASSWORD = "T@n!um"
HOST = "172.16.31.128"
```

```python
PORT = "444"

# Logging conrols
LOGLEVEL = 2
DEBUGFORMAT = False

import sys, tempfile
sys.path.append(PYTAN_LIB_PATH)

import pytan
handler = pytan.Handler(
    username=USERNAME,
    password=PASSWORD,
    host=HOST,
    port=PORT,
    loglevel=LOGLEVEL,
    debugformat=DEBUGFORMAT,
)

print handler

# setup the arguments for the handler method
kwargs = {}
kwargs["objtype"] = u'setting'
kwargs["name"] = u'control_address'

# call the handler with the get method, passing in kwargs for arguments
response = handler.get(**kwargs)

print ""
print "Type of response: ", type(response)

print ""
print "print of response:"
print response

print ""
print "length of response (number of objects returned): "
print len(response)

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

print out
```

Output from Python Code

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

Type of response:  <class 'taniumpy.object_types.system_settings_list.SystemSettingsList'>

print of response:
```

```
SystemSettingsList, len: 1

length of response (number of objects returned):
1

print the first object returned in JSON format:
{
  "_type": "system_setting",
  "default_value": "512:17473:127.0.0.1",
  "hidden_flag": 0,
  "id": 57,
  "name": "control_address",
  "read_only_flag": 0,
  "setting_type": "Server",
  "value": "512:17473:127.0.0.1",
  "value_type": "Text"
}
```

Get user by name

Get a user by name

Example Python Code

```python
# Path to lib directory which contains pytan package
PYTAN_LIB_PATH = '../lib'

# connection info for Tanium Server
USERNAME = "Tanium User"
PASSWORD = "T@n!um"
HOST = "172.16.31.128"
PORT = "444"

# Logging conrols
LOGLEVEL = 2
DEBUGFORMAT = False

import sys, tempfile
sys.path.append(PYTAN_LIB_PATH)

import pytan
handler = pytan.Handler(
    username=USERNAME,
    password=PASSWORD,
    host=HOST,
    port=PORT,
    loglevel=LOGLEVEL,
    debugformat=DEBUGFORMAT,
)

print handler

# setup the arguments for the handler method
kwargs = {}
kwargs["objtype"] = u'user'
```

```python
kwargs["name"] = u'Tanium User'

# call the handler with the get method, passing in kwargs for arguments
response = handler.get(**kwargs)

print ""
print "Type of response: ", type(response)

print ""
print "print of response:"
print response

print ""
print "length of response (number of objects returned): "
print len(response)

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

print out
```

Output from Python Code

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

Type of response:  <class 'taniumpy.object_types.user_list.UserList'>

print of response:
UserList, len: 1

length of response (number of objects returned):
1

print the first object returned in JSON format:
{
  "_type": "user",
  "deleted_flag": 0,
  "group_id": 0,
  "id": 2,
  "last_login": "2014-12-08T21:26:37",
  "name": "Tanium User",
  "permissions": {
    "_type": "permissions",
    "permission": "admin"
  },
  "roles": {
    "_type": "roles",
    "role": [
      {
..trimmed for brevity..
```

Get sensor by id

Get a sensor by id

Example Python Code

```python
# Path to lib directory which contains pytan package
PYTAN_LIB_PATH = '../lib'

# connection info for Tanium Server
USERNAME = "Tanium User"
PASSWORD = "T@n!um"
HOST = "172.16.31.128"
PORT = "444"

# Logging conrols
LOGLEVEL = 2
DEBUGFORMAT = False

import sys, tempfile
sys.path.append(PYTAN_LIB_PATH)

import pytan
handler = pytan.Handler(
    username=USERNAME,
    password=PASSWORD,
    host=HOST,
    port=PORT,
    loglevel=LOGLEVEL,
    debugformat=DEBUGFORMAT,
)

print handler

# setup the arguments for the handler method
kwargs = {}
kwargs["objtype"] = u'sensor'
kwargs["id"] = 1

# call the handler with the get method, passing in kwargs for arguments
response = handler.get(**kwargs)

print ""
print "Type of response: ", type(response)

print ""
print "print of response:"
print response

print ""
print "length of response (number of objects returned): "
print len(response)

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

print out
```

Output from Python Code

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

Type of response:  <class 'taniumpy.object_types.sensor_list.SensorList'>

print of response:
SensorList, len: 1

length of response (number of objects returned):
1

print the first object returned in JSON format:
{
  "_type": "sensor",
  "category": "Reserved",
  "description": "The recorded state of each action a client has taken recently in the form of id:st
  "exclude_from_parse_flag": 1,
  "hash": 1792443391,
  "hidden_flag": 0,
  "id": 1,
  "ignore_case_flag": 1,
  "max_age_seconds": 3600,
  "name": "Action Statuses",
  "queries": {
    "_type": "queries",
    "query": [
      {
..trimmed for brevity..
```

Get sensor by mixed

Get multiple sensors by id, name, and hash

Example Python Code

```
# Path to lib directory which contains pytan package
PYTAN_LIB_PATH = '../lib'

# connection info for Tanium Server
USERNAME = "Tanium User"
PASSWORD = "T@n!um"
HOST = "172.16.31.128"
PORT = "444"

# Logging conrols
LOGLEVEL = 2
DEBUGFORMAT = False

```

```python
import sys, tempfile
sys.path.append(PYTAN_LIB_PATH)

import pytan
handler = pytan.Handler(
    username=USERNAME,
    password=PASSWORD,
    host=HOST,
    port=PORT,
    loglevel=LOGLEVEL,
    debugformat=DEBUGFORMAT,
)

print handler

# setup the arguments for the handler method
kwargs = {}
kwargs["objtype"] = u'sensor'
kwargs["hash"] = [u'322086833']
kwargs["name"] = [u'Computer Name']
kwargs["id"] = [1, 2]

# call the handler with the get method, passing in kwargs for arguments
response = handler.get(**kwargs)

print ""
print "Type of response: ", type(response)

print ""
print "print of response:"
print response

print ""
print "length of response (number of objects returned): "
print len(response)

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

print out
```

Output from Python Code

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

Type of response:  <class 'taniumpy.object_types.sensor_list.SensorList'>

print of response:
SensorList, len: 4

length of response (number of objects returned):
4
```

```

print the first object returned in JSON format:
{
  "_type": "sensor",
  "category": "Reserved",
  "description": "The recorded state of each download a client has made recently in the form of hash
  "exclude_from_parse_flag": 0,
  "hash": 322086833,
  "hidden_flag": 0,
  "id": 4,
  "ignore_case_flag": 1,
  "max_age_seconds": 900,
  "name": "Download Statuses",
  "queries": {
    "_type": "queries",
    "query": [
      {
..trimmed for brevity..
```

Get whitelisted url by id

Get a whitelisted url by id

Example Python Code

```python
# Path to lib directory which contains pytan package
PYTAN_LIB_PATH = '../lib'

# connection info for Tanium Server
USERNAME = "Tanium User"
PASSWORD = "T@n!um"
HOST = "172.16.31.128"
PORT = "444"

# Logging conrols
LOGLEVEL = 2
DEBUGFORMAT = False

import sys, tempfile
sys.path.append(PYTAN_LIB_PATH)

import pytan
handler = pytan.Handler(
    username=USERNAME,
    password=PASSWORD,
    host=HOST,
    port=PORT,
    loglevel=LOGLEVEL,
    debugformat=DEBUGFORMAT,
)

print handler

# setup the arguments for the handler method
kwargs = {}
```

```
kwargs["objtype"] = u'whitelisted_url'
kwargs["id"] = 1

# call the handler with the get method, passing in kwargs for arguments
response = handler.get(**kwargs)

print ""
print "Type of response: ", type(response)

print ""
print "print of response:"
print response

print ""
print "length of response (number of objects returned): "
print len(response)

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

print out
```

Output from Python Code

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

Type of response:  <class 'taniumpy.object_types.white_listed_url_list.WhiteListedUrlList'>

print of response:
WhiteListedUrlList, len: 1

length of response (number of objects returned):
1

print the first object returned in JSON format:
{
  "_type": "white_listed_url",
  "download_seconds": 86400,
  "id": 1,
  "url_regex": "test1"
}
```

Get group by name

Get a group by name

Example Python Code

```python
# Path to lib directory which contains pytan package
PYTAN_LIB_PATH = '../lib'

# connection info for Tanium Server
USERNAME = "Tanium User"
PASSWORD = "T@n!um"
HOST = "172.16.31.128"
PORT = "444"

# Logging conrols
LOGLEVEL = 2
DEBUGFORMAT = False

import sys, tempfile
sys.path.append(PYTAN_LIB_PATH)

import pytan
handler = pytan.Handler(
    username=USERNAME,
    password=PASSWORD,
    host=HOST,
    port=PORT,
    loglevel=LOGLEVEL,
    debugformat=DEBUGFORMAT,
)

print handler

# setup the arguments for the handler method
kwargs = {}
kwargs["objtype"] = u'group'
kwargs["name"] = u'All Computers'

# call the handler with the get method, passing in kwargs for arguments
response = handler.get(**kwargs)

print ""
print "Type of response: ", type(response)

print ""
print "print of response:"
print response

print ""
print "length of response (number of objects returned): "
print len(response)

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

print out
```

Output from Python Code

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

Type of response:  <class 'taniumpy.object_types.group_list.GroupList'>

print of response:
GroupList, len: 1

length of response (number of objects returned):
1

print the first object returned in JSON format:
{
  "_type": "group",
  "and_flag": 0,
  "deleted_flag": 0,
  "filters": {
    "_type": "filters",
    "filter": []
  },
  "id": 1,
  "name": "All Computers",
  "not_flag": 0,
  "sub_groups": {
    "_type": "groups",
    "group": []
  },
..trimmed for brevity..
```

Get sensor by hash

Get a sensor by hash

Example Python Code

```python
# Path to lib directory which contains pytan package
PYTAN_LIB_PATH = '../lib'

# connection info for Tanium Server
USERNAME = "Tanium User"
PASSWORD = "T@n!um"
HOST = "172.16.31.128"
PORT = "444"

# Logging conrols
LOGLEVEL = 2
DEBUGFORMAT = False

import sys, tempfile
sys.path.append(PYTAN_LIB_PATH)

import pytan
handler = pytan.Handler(
    username=USERNAME,
    password=PASSWORD,
    host=HOST,
    port=PORT,
    loglevel=LOGLEVEL,
    debugformat=DEBUGFORMAT,
)

print handler

# setup the arguments for the handler method
kwargs = {}
kwargs["objtype"] = u'sensor'
kwargs["hash"] = u'322086833'

# call the handler with the get method, passing in kwargs for arguments
response = handler.get(**kwargs)

print ""
print "Type of response: ", type(response)

print ""
print "print of response:"
print response

print ""
print "length of response (number of objects returned): "
print len(response)

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

print out
```

Output from Python Code

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

Type of response:  <class 'taniumpy.object_types.sensor_list.SensorList'>

print of response:
SensorList, len: 1

length of response (number of objects returned):
1

print the first object returned in JSON format:
{
  "_type": "sensor",
  "category": "Reserved",
  "description": "The recorded state of each download a client has made recently in the form of hash
  "exclude_from_parse_flag": 0,
  "hash": 322086833,
  "hidden_flag": 0,
  "id": 4,
  "ignore_case_flag": 1,
  "max_age_seconds": 900,
  "name": "Download Statuses",
  "queries": {
    "_type": "queries",
    "query": [
      {
..trimmed for brevity..
```

Get package by name

Get a package by name

Example Python Code

```python
# Path to lib directory which contains pytan package
PYTAN_LIB_PATH = '../lib'

# connection info for Tanium Server
USERNAME = "Tanium User"
PASSWORD = "T@n!um"
HOST = "172.16.31.128"
PORT = "444"

# Logging conrols
LOGLEVEL = 2
DEBUGFORMAT = False

import sys, tempfile
sys.path.append(PYTAN_LIB_PATH)

import pytan
handler = pytan.Handler(
    username=USERNAME,
    password=PASSWORD,
    host=HOST,
    port=PORT,
    loglevel=LOGLEVEL,
    debugformat=DEBUGFORMAT,
)

print handler

# setup the arguments for the handler method
kwargs = {}
kwargs["objtype"] = u'package'
kwargs["name"] = u'Distribute Patch Tools'

# call the handler with the get method, passing in kwargs for arguments
response = handler.get(**kwargs)

print ""
print "Type of response: ", type(response)

print ""
print "print of response:"
print response

print ""
print "length of response (number of objects returned): "
print len(response)

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

print out
```

Output from Python Code

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

Type of response:  <class 'taniumpy.object_types.package_spec_list.PackageSpecList'>

print of response:
PackageSpecList, len: 1

length of response (number of objects returned):
1

print the first object returned in JSON format:
{
  "_type": "package_spec",
  "available_time": "2014-12-08T19:25:53",
  "command": "cmd /c cscript //T:1800 copy-to-tanium-dir.vbs \"Tools\"",
  "command_timeout": 1800,
  "creation_time": "2014-12-08T19:21:06",
  "deleted_flag": 0,
  "display_name": "Distribute Patch Tools",
  "expire_seconds": 2400,
  "files": {
    "_type": "package_files",
    "file": [
      {
        "_type": "file",
        "bytes_downloaded": 3041,
..trimmed for brevity..
```

Get sensor by names

Get multiple sensors by name

Example Python Code

```
# Path to lib directory which contains pytan package
PYTAN_LIB_PATH = '../lib'

# connection info for Tanium Server
USERNAME = "Tanium User"
PASSWORD = "T@n!um"
HOST = "172.16.31.128"
PORT = "444"

# Logging conrols
LOGLEVEL = 2
DEBUGFORMAT = False

import sys, tempfile
sys.path.append(PYTAN_LIB_PATH)

import pytan
handler = pytan.Handler(
    username=USERNAME,
    password=PASSWORD,
    host=HOST,
    port=PORT,
    loglevel=LOGLEVEL,
    debugformat=DEBUGFORMAT,
)

print handler

# setup the arguments for the handler method
kwargs = {}
kwargs["objtype"] = u'sensor'
kwargs["name"] = [u'Computer Name', u'Action Statuses']

# call the handler with the get method, passing in kwargs for arguments
response = handler.get(**kwargs)

print ""
print "Type of response: ", type(response)

print ""
print "print of response:"
print response

print ""
print "length of response (number of objects returned): "
print len(response)

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

print out
```

Output from Python Code

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

Type of response:  <class 'taniumpy.object_types.sensor_list.SensorList'>

print of response:
SensorList, len: 2

length of response (number of objects returned):
2

print the first object returned in JSON format:
{
  "_type": "sensor",
  "category": "Reserved",
  "description": "The assigned name of the client machine.\nExample: workstation-1.company.com",
  "exclude_from_parse_flag": 0,
  "hash": 3409330187,
  "hidden_flag": 0,
  "id": 3,
  "ignore_case_flag": 1,
  "max_age_seconds": 86400,
  "name": "Computer Name",
  "queries": {
    "_type": "queries",
    "query": [
      {
..trimmed for brevity..
```

Get saved question by name

Get saved question by name

Example Python Code

```python
# Path to lib directory which contains pytan package
PYTAN_LIB_PATH = '../lib'

# connection info for Tanium Server
USERNAME = "Tanium User"
PASSWORD = "T@n!um"
HOST = "172.16.31.128"
PORT = "444"

# Logging conrols
LOGLEVEL = 2
DEBUGFORMAT = False

import sys, tempfile
sys.path.append(PYTAN_LIB_PATH)

import pytan
handler = pytan.Handler(
    username=USERNAME,
    password=PASSWORD,
    host=HOST,
    port=PORT,
    loglevel=LOGLEVEL,
    debugformat=DEBUGFORMAT,
)

print handler

# setup the arguments for the handler method
kwargs = {}
kwargs["objtype"] = u'saved_question'
kwargs["name"] = u'Installed Applications'

# call the handler with the get method, passing in kwargs for arguments
response = handler.get(**kwargs)

print ""
print "Type of response: ", type(response)

print ""
print "print of response:"
print response

print ""
print "length of response (number of objects returned): "
print len(response)

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

print out
```

Output from Python Code

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

Type of response:  <class 'taniumpy.object_types.saved_question_list.SavedQuestionList'>

print of response:
SavedQuestionList, len: 1

length of response (number of objects returned):
1

print the first object returned in JSON format:
{
  "_type": "saved_question",
  "action_tracking_flag": 0,
  "archive_enabled_flag": 0,
  "archive_owner": {
    "_type": "user",
    "id": 1,
    "name": "Jim Olsen"
  },
  "expire_seconds": 600,
  "hidden_flag": 0,
  "id": 92,
  "issue_seconds": 120,
  "issue_seconds_never_flag": 0,
  "keep_seconds": 3600,
..trimmed for brevity..
```

Get user by id

Get a user by id

Example Python Code

```
# Path to lib directory which contains pytan package
PYTAN_LIB_PATH = '../lib'

# connection info for Tanium Server
USERNAME = "Tanium User"
PASSWORD = "T@n!um"
HOST = "172.16.31.128"
PORT = "444"

# Logging conrols
LOGLEVEL = 2
DEBUGFORMAT = False

import sys, tempfile
sys.path.append(PYTAN_LIB_PATH)

import pytan
handler = pytan.Handler(
    username=USERNAME,
    password=PASSWORD,
    host=HOST,
    port=PORT,
    loglevel=LOGLEVEL,
    debugformat=DEBUGFORMAT,
)

print handler

# setup the arguments for the handler method
kwargs = {}
kwargs["objtype"] = u'user'
kwargs["id"] = 1

# call the handler with the get method, passing in kwargs for arguments
response = handler.get(**kwargs)

print ""
print "Type of response: ", type(response)

print ""
print "print of response:"
print response

print ""
print "length of response (number of objects returned): "
print len(response)

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

print out
```

Output from Python Code

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

Type of response:  <class 'taniumpy.object_types.user_list.UserList'>

print of response:
UserList, len: 1

length of response (number of objects returned):
1

print the first object returned in JSON format:
{
  "_type": "user",
  "deleted_flag": 0,
  "group_id": 0,
  "id": 1,
  "last_login": "2014-12-08T19:28:09",
  "name": "Jim Olsen",
  "permissions": {
    "_type": "permissions",
    "permission": "admin"
  },
  "roles": {
    "_type": "roles",
    "role": [
      {
..trimmed for brevity..
```

Get sensor by name

Get a sensor by name

Example Python Code

```python
# Path to lib directory which contains pytan package
PYTAN_LIB_PATH = '../lib'

# connection info for Tanium Server
USERNAME = "Tanium User"
PASSWORD = "T@n!um"
HOST = "172.16.31.128"
PORT = "444"

# Logging conrols
LOGLEVEL = 2
DEBUGFORMAT = False

import sys, tempfile
sys.path.append(PYTAN_LIB_PATH)

import pytan
handler = pytan.Handler(
    username=USERNAME,
    password=PASSWORD,
    host=HOST,
    port=PORT,
    loglevel=LOGLEVEL,
    debugformat=DEBUGFORMAT,
)

print handler

# setup the arguments for the handler method
kwargs = {}
kwargs["objtype"] = u'sensor'
kwargs["name"] = u'Computer Name'

# call the handler with the get method, passing in kwargs for arguments
response = handler.get(**kwargs)

print ""
print "Type of response: ", type(response)

print ""
print "print of response:"
print response

print ""
print "length of response (number of objects returned): "
print len(response)

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

print out
```

Output from Python Code

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

Type of response:  <class 'taniumpy.object_types.sensor_list.SensorList'>

print of response:
SensorList, len: 1

length of response (number of objects returned):
1

print the first object returned in JSON format:
{
  "_type": "sensor",
  "category": "Reserved",
  "description": "The assigned name of the client machine.\nExample: workstation-1.company.com",
  "exclude_from_parse_flag": 0,
  "hash": 3409330187,
  "hidden_flag": 0,
  "id": 3,
  "ignore_case_flag": 1,
  "max_age_seconds": 86400,
  "name": "Computer Name",
  "queries": {
    "_type": "queries",
    "query": [
      {
..trimmed for brevity..
```

Get saved action by name

Get a saved action by name

Example Python Code

```python
# Path to lib directory which contains pytan package
PYTAN_LIB_PATH = '../lib'

# connection info for Tanium Server
USERNAME = "Tanium User"
PASSWORD = "T@n!um"
HOST = "172.16.31.128"
PORT = "444"

# Logging conrols
LOGLEVEL = 2
DEBUGFORMAT = False

import sys, tempfile
sys.path.append(PYTAN_LIB_PATH)

import pytan
handler = pytan.Handler(
    username=USERNAME,
    password=PASSWORD,
    host=HOST,
    port=PORT,
    loglevel=LOGLEVEL,
    debugformat=DEBUGFORMAT,
)

print handler

# setup the arguments for the handler method
kwargs = {}
kwargs["objtype"] = u'saved_action'
kwargs["name"] = u'Distribute Tanium Standard Utilities'

# call the handler with the get method, passing in kwargs for arguments
response = handler.get(**kwargs)

print ""
print "Type of response: ", type(response)

print ""
print "print of response:"
print response

print ""
print "length of response (number of objects returned): "
print len(response)

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

print out
```

Output from Python Code

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

Type of response:  <class 'taniumpy.object_types.saved_action_list.SavedActionList'>

print of response:
SavedActionList, len: 1

length of response (number of objects returned):
1

print the first object returned in JSON format:
{
  "_type": "saved_action",
  "action_group_id": 0,
  "comment": "Distributes the Hardware Tools used for hardware identification.",
  "creation_time": "2014-12-08T19:22:36",
  "distribute_seconds": 0,
  "end_time": "Never",
  "expire_seconds": 660,
  "id": 14,
  "issue_count": 0,
  "issue_seconds": 86400,
  "last_action": {
    "_type": "action",
    "id": 4294967295,
    "start_time": "Never"
..trimmed for brevity..
```

Get all users

Get all users

Example Python Code

```python
# Path to lib directory which contains pytan package
PYTAN_LIB_PATH = '../lib'

# connection info for Tanium Server
USERNAME = "Tanium User"
PASSWORD = "T@n!um"
HOST = "172.16.31.128"
PORT = "444"

# Logging conrols
LOGLEVEL = 2
DEBUGFORMAT = False

import sys, tempfile
sys.path.append(PYTAN_LIB_PATH)

import pytan
handler = pytan.Handler(
    username=USERNAME,
    password=PASSWORD,
    host=HOST,
    port=PORT,
    loglevel=LOGLEVEL,
    debugformat=DEBUGFORMAT,
)

print handler

# setup the arguments for the handler method
kwargs = {}
kwargs["objtype"] = u'user'

# call the handler with the get_all method, passing in kwargs for arguments
response = handler.get_all(**kwargs)

print ""
print "Type of response: ", type(response)

print ""
print "print of response:"
print response

print ""
print "length of response (number of objects returned): "
print len(response)

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

print out
```

Output from Python Code

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

Type of response:  <class 'taniumpy.object_types.user_list.UserList'>

print of response:
UserList, len: 6

length of response (number of objects returned):
6

print the first object returned in JSON format:
{
  "_type": "user",
  "deleted_flag": 0,
  "group_id": 0,
  "id": 1,
  "last_login": "2014-12-08T19:28:09",
  "name": "Jim Olsen",
  "permissions": {
    "_type": "permissions",
    "permission": "admin"
  },
  "roles": {
    "_type": "roles",
    "role": [
      {
..trimmed for brevity..
```

Get all saved actions

Get all saved actions

Example Python Code

```
# Path to lib directory which contains pytan package
PYTAN_LIB_PATH = '../lib'

# connection info for Tanium Server
USERNAME = "Tanium User"
PASSWORD = "T@n!um"
HOST = "172.16.31.128"
PORT = "444"

# Logging conrols
LOGLEVEL = 2
DEBUGFORMAT = False

import sys, tempfile
sys.path.append(PYTAN_LIB_PATH)

import pytan
handler = pytan.Handler(
    username=USERNAME,
    password=PASSWORD,
    host=HOST,
    port=PORT,
    loglevel=LOGLEVEL,
    debugformat=DEBUGFORMAT,
)

print handler

# setup the arguments for the handler method
kwargs = {}
kwargs["objtype"] = u'saved_action'

# call the handler with the get_all method, passing in kwargs for arguments
response = handler.get_all(**kwargs)

print ""
print "Type of response: ", type(response)

print ""
print "print of response:"
print response

print ""
print "length of response (number of objects returned): "
print len(response)

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

print out
```

Output from Python Code

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

Type of response:  <class 'taniumpy.object_types.saved_action_list.SavedActionList'>

print of response:
SavedActionList, len: 38

length of response (number of objects returned):
38

print the first object returned in JSON format:
{
  "_type": "saved_action",
  "action_group_id": 0,
  "cache_row_id": 0,
  "comment": "Scans for unmanaged assets on the network.",
  "creation_time": "2014-12-08T19:22:33",
  "distribute_seconds": 600,
  "end_time": "Never",
  "expire_seconds": 1800,
  "id": 1,
  "issue_count": 3,
  "issue_seconds": 3600,
  "last_action": {
    "_type": "action",
    "id": 45,
..trimmed for brevity..
```

Get all settings

Get all system settings

Example Python Code

```python
# Path to lib directory which contains pytan package
PYTAN_LIB_PATH = '../lib'

# connection info for Tanium Server
USERNAME = "Tanium User"
PASSWORD = "T@n!um"
HOST = "172.16.31.128"
PORT = "444"

# Logging conrols
LOGLEVEL = 2
DEBUGFORMAT = False

import sys, tempfile
sys.path.append(PYTAN_LIB_PATH)

import pytan
handler = pytan.Handler(
    username=USERNAME,
    password=PASSWORD,
    host=HOST,
    port=PORT,
    loglevel=LOGLEVEL,
    debugformat=DEBUGFORMAT,
)

print handler

# setup the arguments for the handler method
kwargs = {}
kwargs["objtype"] = u'setting'

# call the handler with the get_all method, passing in kwargs for arguments
response = handler.get_all(**kwargs)

print ""
print "Type of response: ", type(response)

print ""
print "print of response:"
print response

print ""
print "length of response (number of objects returned): "
print len(response)

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

print out
```

Output from Python Code

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

Type of response:  <class 'taniumpy.object_types.system_settings_list.SystemSettingsList'>

print of response:
SystemSettingsList, len: 88

length of response (number of objects returned):
88

print the first object returned in JSON format:
{
  "_type": "system_setting",
  "cache_row_id": 0,
  "default_value": "0",
  "hidden_flag": 0,
  "id": 1,
  "name": "load_initial_content",
  "read_only_flag": 0,
  "setting_type": "Server",
  "value": "0",
  "value_type": "Numeric"
}
```

Get all saved questions

Get all saved questions

Example Python Code

```
# Path to lib directory which contains pytan package
PYTAN_LIB_PATH = '../lib'

# connection info for Tanium Server
USERNAME = "Tanium User"
PASSWORD = "T@n!um"
HOST = "172.16.31.128"
PORT = "444"

# Logging conrols
LOGLEVEL = 2
DEBUGFORMAT = False

import sys, tempfile
sys.path.append(PYTAN_LIB_PATH)

import pytan
handler = pytan.Handler(
    username=USERNAME,
    password=PASSWORD,
    host=HOST,
    port=PORT,
    loglevel=LOGLEVEL,
    debugformat=DEBUGFORMAT,
)

print handler

# setup the arguments for the handler method
kwargs = {}
kwargs["objtype"] = u'saved_question'

# call the handler with the get_all method, passing in kwargs for arguments
response = handler.get_all(**kwargs)

print ""
print "Type of response: ", type(response)

print ""
print "print of response:"
print response

print ""
print "length of response (number of objects returned): "
print len(response)

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

print out
```

Output from Python Code

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

Type of response:  <class 'taniumpy.object_types.saved_question_list.SavedQuestionList'>

print of response:
SavedQuestionList, len: 176

length of response (number of objects returned):
176

print the first object returned in JSON format:
{
  "_type": "saved_question",
  "action_tracking_flag": 0,
  "archive_enabled_flag": 0,
  "archive_owner": {
    "_type": "user",
    "id": 1,
    "name": "Jim Olsen"
  },
  "cache_row_id": 0,
  "expire_seconds": 600,
  "hidden_flag": 0,
  "id": 1,
  "issue_seconds": 120,
  "issue_seconds_never_flag": 0,
..trimmed for brevity..
```

Get all userroless

Get all user roles

Example Python Code

```
# Path to lib directory which contains pytan package
PYTAN_LIB_PATH = '../lib'

# connection info for Tanium Server
USERNAME = "Tanium User"
PASSWORD = "T@n!um"
HOST = "172.16.31.128"
```

```
PORT = "444"

# Logging conrols
LOGLEVEL = 2
DEBUGFORMAT = False

import sys, tempfile
sys.path.append(PYTAN_LIB_PATH)

import pytan
handler = pytan.Handler(
    username=USERNAME,
    password=PASSWORD,
    host=HOST,
    port=PORT,
    loglevel=LOGLEVEL,
    debugformat=DEBUGFORMAT,
)

print handler

# setup the arguments for the handler method
kwargs = {}
kwargs["objtype"] = u'userrole'

# call the handler with the get_all method, passing in kwargs for arguments
response = handler.get_all(**kwargs)

print ""
print "Type of response: ", type(response)

print ""
print "print of response:"
print response

print ""
print "length of response (number of objects returned): "
print len(response)

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

print out
```

Output from Python Code

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

Type of response:  <class 'taniumpy.object_types.user_role_list.UserRoleList'>

print of response:
UserRoleList, len: 9
```

```

length of response (number of objects returned):
9

print the first object returned in JSON format:
{
  "_type": "role",
  "description": "Administrators can perform all functions in the system, including creating other u
  "id": 1,
  "name": "Administrator",
  "permissions": {
    "_type": "permissions",
    "permission": "admin"
  }
}
```

Get all questions

Get all questions

Example Python Code

```
# Path to lib directory which contains pytan package
PYTAN_LIB_PATH = '../lib'

# connection info for Tanium Server
USERNAME = "Tanium User"
PASSWORD = "T@n!um"
HOST = "172.16.31.128"
PORT = "444"

# Logging conrols
LOGLEVEL = 2
DEBUGFORMAT = False

import sys, tempfile
sys.path.append(PYTAN_LIB_PATH)

import pytan
handler = pytan.Handler(
    username=USERNAME,
    password=PASSWORD,
    host=HOST,
    port=PORT,
    loglevel=LOGLEVEL,
    debugformat=DEBUGFORMAT,
)

print handler

# setup the arguments for the handler method
kwargs = {}
kwargs["objtype"] = u'question'

# call the handler with the get_all method, passing in kwargs for arguments
```

```python
response = handler.get_all(**kwargs)

print ""
print "Type of response: ", type(response)

print ""
print "print of response:"
print response

print ""
print "length of response (number of objects returned): "
print len(response)

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

print out
```

Output from Python Code

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

Type of response:  <class 'taniumpy.object_types.question_list.QuestionList'>

print of response:
QuestionList, len: 255

length of response (number of objects returned):
255

print the first object returned in JSON format:
{
  "_type": "question",
  "action_tracking_flag": 0,
  "cache_row_id": 0,
  "context_group": {
    "_type": "group",
    "id": 0
  },
  "expiration": "2014-12-08T19:30:12",
  "expire_seconds": 600,
  "hidden_flag": 0,
  "id": 1,
  "management_rights_group": {
    "_type": "group",
    "id": 0
..trimmed for brevity..
```

Get all groups

Get all groups

Example Python Code

```python
# Path to lib directory which contains pytan package
PYTAN_LIB_PATH = '../lib'

# connection info for Tanium Server
USERNAME = "Tanium User"
PASSWORD = "T@n!um"
HOST = "172.16.31.128"
PORT = "444"

# Logging conrols
LOGLEVEL = 2
DEBUGFORMAT = False

import sys, tempfile
sys.path.append(PYTAN_LIB_PATH)

import pytan
handler = pytan.Handler(
    username=USERNAME,
    password=PASSWORD,
    host=HOST,
    port=PORT,
    loglevel=LOGLEVEL,
    debugformat=DEBUGFORMAT,
)

print handler

# setup the arguments for the handler method
kwargs = {}
kwargs["objtype"] = u'group'

# call the handler with the get_all method, passing in kwargs for arguments
response = handler.get_all(**kwargs)

print ""
print "Type of response: ", type(response)

print ""
print "print of response:"
print response

print ""
print "length of response (number of objects returned): "
print len(response)

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

print out
```

Output from Python Code

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

Type of response:  <class 'taniumpy.object_types.group_list.GroupList'>

print of response:
GroupList, len: 5

length of response (number of objects returned):
5

print the first object returned in JSON format:
{
  "_type": "group",
  "and_flag": 0,
  "deleted_flag": 0,
  "filters": {
    "_type": "filters",
    "filter": []
  },
  "id": 1,
  "name": "All Computers",
  "not_flag": 0,
  "sub_groups": {
    "_type": "groups",
    "group": []
  },
..trimmed for brevity..
```

Get all sensors

Get all sensors

Example Python Code

```python
# Path to lib directory which contains pytan package
PYTAN_LIB_PATH = '../lib'

# connection info for Tanium Server
USERNAME = "Tanium User"
PASSWORD = "T@n!um"
HOST = "172.16.31.128"
PORT = "444"

# Logging conrols
LOGLEVEL = 2
DEBUGFORMAT = False

import sys, tempfile
```

```
sys.path.append(PYTAN_LIB_PATH)

import pytan
handler = pytan.Handler(
    username=USERNAME,
    password=PASSWORD,
    host=HOST,
    port=PORT,
    loglevel=LOGLEVEL,
    debugformat=DEBUGFORMAT,
)

print handler

# setup the arguments for the handler method
kwargs = {}
kwargs["objtype"] = u'sensor'

# call the handler with the get_all method, passing in kwargs for arguments
response = handler.get_all(**kwargs)

print ""
print "Type of response: ", type(response)

print ""
print "print of response:"
print response

print ""
print "length of response (number of objects returned): "
print len(response)

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

print out
```

Output from Python Code

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

Type of response:  <class 'taniumpy.object_types.sensor_list.SensorList'>

print of response:
SensorList, len: 536

length of response (number of objects returned):
536

print the first object returned in JSON format:
{
  "_type": "sensor",
  "cache_row_id": 0,
  "category": "Reserved",
  "description": "The recorded state of each action a client has taken recently in the form of id:st
  "exclude_from_parse_flag": 1,
  "hash": 1792443391,
  "hidden_flag": 0,
  "id": 1,
  "ignore_case_flag": 1,
  "max_age_seconds": 3600,
  "name": "Action Statuses",
  "queries": {
    "_type": "queries",
    "query": [
..trimmed for brevity..
```

Get all whitelisted urls

Get all whitelisted urls

Example Python Code

```python
# Path to lib directory which contains pytan package
PYTAN_LIB_PATH = '../lib'

# connection info for Tanium Server
USERNAME = "Tanium User"
PASSWORD = "T@n!um"
HOST = "172.16.31.128"
PORT = "444"

# Logging conrols
LOGLEVEL = 2
DEBUGFORMAT = False

import sys, tempfile
sys.path.append(PYTAN_LIB_PATH)

import pytan
handler = pytan.Handler(
    username=USERNAME,
    password=PASSWORD,
    host=HOST,
    port=PORT,
    loglevel=LOGLEVEL,
    debugformat=DEBUGFORMAT,
)

print handler

# setup the arguments for the handler method
kwargs = {}
kwargs["objtype"] = u'whitelisted_url'

# call the handler with the get_all method, passing in kwargs for arguments
response = handler.get_all(**kwargs)
```

```

print ""
print "Type of response: ", type(response)

print ""
print "print of response:"
print response

print ""
print "length of response (number of objects returned): "
print len(response)

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

print out
```

Output from Python Code

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

Type of response:  <class 'taniumpy.object_types.white_listed_url_list.WhiteListedUrlList'>

print of response:
WhiteListedUrlList, len: 46

length of response (number of objects returned):
46

print the first object returned in JSON format:
{
  "_type": "white_listed_url",
  "download_seconds": 86400,
  "id": 1,
  "url_regex": "test1"
}
```

Get all clients

Get all clients

Example Python Code

```
# Path to lib directory which contains pytan package
PYTAN_LIB_PATH = '../lib'

# connection info for Tanium Server
USERNAME = "Tanium User"
PASSWORD = "T@n!um"
```

```
HOST = "172.16.31.128"
PORT = "444"

# Logging conrols
LOGLEVEL = 2
DEBUGFORMAT = False

import sys, tempfile
sys.path.append(PYTAN_LIB_PATH)

import pytan
handler = pytan.Handler(
    username=USERNAME,
    password=PASSWORD,
    host=HOST,
    port=PORT,
    loglevel=LOGLEVEL,
    debugformat=DEBUGFORMAT,
)

print handler

# setup the arguments for the handler method
kwargs = {}
kwargs["objtype"] = u'client'

# call the handler with the get_all method, passing in kwargs for arguments
response = handler.get_all(**kwargs)

print ""
print "Type of response: ", type(response)

print ""
print "print of response:"
print response

print ""
print "length of response (number of objects returned): "
print len(response)

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

print out
```

Output from Python Code

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

Type of response:  <class 'taniumpy.object_types.system_status_list.SystemStatusList'>

print of response:
```

```
SystemStatusList, len: 6

length of response (number of objects returned):
6

print the first object returned in JSON format:
{
  "_type": "client_status",
  "cache_row_id": 0,
  "computer_id": "660621737",
  "full_version": "5.1.314.7724",
  "host_name": "Casus-Belli.local",
  "ipaddress_client": "172.16.31.1",
  "ipaddress_server": "172.16.31.1",
  "last_registration": "2014-12-08T21:26:16",
  "port_number": 17472,
  "protocol_version": 314,
  "send_state": "Forward Only",
  "status": "Leader, Slow Link Behind"
}
```

Get all packages

Get all packages

Example Python Code

```python
# Path to lib directory which contains pytan package
PYTAN_LIB_PATH = '../lib'

# connection info for Tanium Server
USERNAME = "Tanium User"
PASSWORD = "T@n!um"
HOST = "172.16.31.128"
PORT = "444"

# Logging conrols
LOGLEVEL = 2
DEBUGFORMAT = False

import sys, tempfile
sys.path.append(PYTAN_LIB_PATH)

import pytan
handler = pytan.Handler(
    username=USERNAME,
    password=PASSWORD,
    host=HOST,
    port=PORT,
    loglevel=LOGLEVEL,
    debugformat=DEBUGFORMAT,
)

print handler

```

```
# setup the arguments for the handler method
kwargs = {}
kwargs["objtype"] = u'package'

# call the handler with the get_all method, passing in kwargs for arguments
response = handler.get_all(**kwargs)

print ""
print "Type of response: ", type(response)

print ""
print "print of response:"
print response

print ""
print "length of response (number of objects returned): "
print len(response)

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

print out
```

Output from Python Code

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

Type of response:  <class 'taniumpy.object_types.package_spec_list.PackageSpecList'>

print of response:
PackageSpecList, len: 92

length of response (number of objects returned):
92

print the first object returned in JSON format:
{
  "_type": "package_spec",
  "available_time": "2014-12-08T19:21:15",
  "cache_row_id": 0,
  "command": "cmd /c cscript //T:900 java-installer.vbs /KillAppsUsingJava:Yes /RebootIfNeeded:Yes /
  "command_timeout": 900,
  "creation_time": "2014-12-08T19:20:46",
  "deleted_flag": 0,
  "display_name": "Update Java 64-bit - Kill / Reboot",
  "expire_seconds": 1500,
  "files": {
    "_type": "package_files",
    "file": [
      {
        "_type": "file",
..trimmed for brevity..
```

Get all actions

Get all actions

Example Python Code

```
# Path to lib directory which contains pytan package
PYTAN_LIB_PATH = '../lib'

# connection info for Tanium Server
USERNAME = "Tanium User"
PASSWORD = "T@n!um"
HOST = "172.16.31.128"
PORT = "444"

# Logging conrols
LOGLEVEL = 2
DEBUGFORMAT = False

import sys, tempfile
sys.path.append(PYTAN_LIB_PATH)

import pytan
handler = pytan.Handler(
    username=USERNAME,
    password=PASSWORD,
    host=HOST,
    port=PORT,
    loglevel=LOGLEVEL,
    debugformat=DEBUGFORMAT,
)

print handler

# setup the arguments for the handler method
kwargs = {}
kwargs["objtype"] = u'action'

# call the handler with the get_all method, passing in kwargs for arguments
response = handler.get_all(**kwargs)

print ""
print "Type of response: ", type(response)

print ""
print "print of response:"
print response

print ""
print "length of response (number of objects returned): "
print len(response)

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

print out
```

Output from Python Code

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

Type of response:  <class 'taniumpy.object_types.action_list.ActionList'>

print of response:
ActionList, len: 30

length of response (number of objects returned):
30

print the first object returned in JSON format:
{
  "_type": "action",
  "action_group": {
    "_type": "group",
    "id": 0,
    "name": "Default"
  },
  "cache_row_id": 0,
  "comment": "Scans for unmanaged assets on the network.",
  "creation_time": "2014-12-08T19:26:36",
  "distribute_seconds": 600,
  "expiration_time": "2014-12-08T20:16:36",
  "expire_seconds": 3000,
  "history_saved_question": {
    "_type": "saved_question",
..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

```
# Path to lib directory which contains pytan package
PYTAN_LIB_PATH = '../lib'

# connection info for Tanium Server
USERNAME = "Tanium User"
PASSWORD = "T@n!um"
HOST = "172.16.31.128"
```

```python
PORT = "444"

# Logging conrols
LOGLEVEL = 2
DEBUGFORMAT = False

import sys, tempfile
sys.path.append(PYTAN_LIB_PATH)

import pytan
handler = pytan.Handler(
    username=USERNAME,
    password=PASSWORD,
    host=HOST,
    port=PORT,
    loglevel=LOGLEVEL,
    debugformat=DEBUGFORMAT,
)

print handler

# setup the arguments for the handler method
kwargs = {}
kwargs["objtype"] = u'action'
kwargs["name"] = u'Distribute Tanium Standard Utilities'


# call the handler with the get method, passing in kwargs for arguments
# this should throw an exception: pytan.utils.HandlerError
import traceback
try:
    handler.get(**kwargs)
except Exception as e:
    traceback.print_exc(file=sys.stdout)
```

Output from Python Code

```
Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
Traceback (most recent call last):
  File "<string>", line 39, in <module>
  File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 1562, in get
    raise HandlerError(err(objtype, api_attrs))
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

```python
# Path to lib directory which contains pytan package
PYTAN_LIB_PATH = '../lib'

# connection info for Tanium Server
```

```
USERNAME = "Tanium User"
PASSWORD = "T@n!um"
HOST = "172.16.31.128"
PORT = "444"

# Logging conrols
LOGLEVEL = 2
DEBUGFORMAT = False

import sys, tempfile
sys.path.append(PYTAN_LIB_PATH)

import pytan
handler = pytan.Handler(
    username=USERNAME,
    password=PASSWORD,
    host=HOST,
    port=PORT,
    loglevel=LOGLEVEL,
    debugformat=DEBUGFORMAT,
)

print handler

# setup the arguments for the handler method
kwargs = {}
kwargs["objtype"] = u'question'
kwargs["name"] = u'dweedle'


# call the handler with the get method, passing in kwargs for arguments
# this should throw an exception: pytan.utils.HandlerError
import traceback
try:
    handler.get(**kwargs)
except Exception as e:
    traceback.print_exc(file=sys.stdout)
```

Output from Python Code

```
Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
Traceback (most recent call last):
  File "<string>", line 39, in <module>
  File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 1562, in get
    raise HandlerError(err(objtype, api_attrs))
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

```
# Path to lib directory which contains pytan package
PYTAN_LIB_PATH = '../lib'

# connection info for Tanium Server
USERNAME = "Tanium User"
PASSWORD = "T@n!um"
HOST = "172.16.31.128"
PORT = "444"

# Logging conrols
LOGLEVEL = 2
DEBUGFORMAT = False

import sys, tempfile
sys.path.append(PYTAN_LIB_PATH)

import pytan
handler = pytan.Handler(
    username=USERNAME,
    password=PASSWORD,
    host=HOST,
    port=PORT,
    loglevel=LOGLEVEL,
    debugformat=DEBUGFORMAT,
)

print handler

# setup the arguments for the handler method
kwargs = {}
kwargs["run"] = True
kwargs["package"] = u'Distribute Tanium Standard Utilities'

# call the handler with the deploy_action_human method, passing in kwargs for arguments
response = handler.deploy_action_human(**kwargs)
import pprint, io

print ""
print "Type of response: ", type(response)

print ""
print "Pretty print of response:"
print pprint.pformat(response)

print ""
print "Print of action object: "
print response['action_object']

# create an IO stream to store CSV results to
out = io.BytesIO()

# if results were returned (i.e. get_results=True was one of the kwargs passed in):
if response['action_results']:
    # call the write_csv() method to convert response to CSV and store it in out
    response['action_results'].write_csv(out, response['action_results'])

    print ""
    print "CSV Results of response: "
    print out.getvalue()
```

Output from Python Code

```
Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
2014-12-08 16:26:39,899 INFO     question_progress: Results 0% (Get Online = "True" from all machine
2014-12-08 16:26:44,920 INFO     question_progress: Results 50% (Get Online = "True" from all machin
2014-12-08 16:26:49,936 INFO     question_progress: Results 100% (Get Online = "True" from all machi
2014-12-08 16:26:50,011 INFO     action_progress: Action Results Passed: 0% (API Deploy Distribute T
2014-12-08 16:26:51,045 INFO     action_progress: Action Results Passed: 0% (API Deploy Distribute T
2014-12-08 16:26:52,079 INFO     action_progress: Action Results Passed: 0% (API Deploy Distribute T
2014-12-08 16:26:53,112 INFO     action_progress: Action Results Passed: 0% (API Deploy Distribute T
2014-12-08 16:26:54,152 INFO     action_progress: Action Results Passed: 0% (API Deploy Distribute T
2014-12-08 16:26:55,185 INFO     action_progress: Action Results Passed: 0% (API Deploy Distribute T
2014-12-08 16:26:56,219 INFO     action_progress: Action Results Passed: 0% (API Deploy Distribute T
2014-12-08 16:26:57,251 INFO     action_progress: Action Results Passed: 0% (API Deploy Distribute T
2014-12-08 16:26:58,286 INFO     action_progress: Action Results Passed: 0% (API Deploy Distribute T
2014-12-08 16:26:59,329 INFO     action_progress: Action Results Passed: 0% (API Deploy Distribute T
2014-12-08 16:27:00,372 INFO     action_progress: Action Results Passed: 0% (API Deploy Distribute T
2014-12-08 16:27:01,482 INFO     action_progress: Action Results Passed: 0% (API Deploy Distribute T
2014-12-08 16:27:02,523 INFO     action_progress: Action Results Passed: 0% (API Deploy Distribute T
2014-12-08 16:27:03,559 INFO     action_progress: Action Results Passed: 0% (API Deploy Distribute T
2014-12-08 16:27:04,603 INFO     action_progress: Action Results Passed: 0% (API Deploy Distribute T
2014-12-08 16:27:05,647 INFO     action_progress: Action Results Passed: 0% (API Deploy Distribute T
2014-12-08 16:27:06,688 INFO     action_progress: Action Results Passed: 0% (API Deploy Distribute T
2014-12-08 16:27:07,731 INFO     action_progress: Action Results Passed: 0% (API Deploy Distribute T
2014-12-08 16:27:08,770 INFO     action_progress: Action Results Passed: 0% (API Deploy Distribute T
2014-12-08 16:27:09,813 INFO     action_progress: Action Results Passed: 33% (API Deploy Distribute
2014-12-08 16:27:10,851 INFO     action_progress: Action Results Passed: 33% (API Deploy Distribute
2014-12-08 16:27:11,889 INFO     action_progress: Action Results Passed: 50% (API Deploy Distribute
2014-12-08 16:27:12,925 INFO     action_progress: Action Results Passed: 50% (API Deploy Distribute
2014-12-08 16:27:13,968 INFO     action_progress: Action Results Passed: 50% (API Deploy Distribute
2014-12-08 16:27:15,005 INFO     action_progress: Action Results Passed: 83% (API Deploy Distribute
2014-12-08 16:27:16,041 INFO     action_progress: Action Results Passed: 100% (API Deploy Distribute
2014-12-08 16:27:16,077 INFO     action_progress: Action Results Completed: 100% (API Deploy Distrib
2014-12-08 16:27:16,077 INFO     action_progress: API Deploy Distribute Tanium Standard Utilities Re
    Running Count: 0
    Success Count: 6
    Failed Count: 0
    Unknown Count: 0
    Finished Count: 6
    Total Count: 6
    Finished Count must equal: 6

Type of response:  <type 'dict'>

Pretty print of response:
{'action_object': <taniumpy.object_types.action.Action object at 0x102967d10>,
 'action_progress_human': 'API Deploy Distribute Tanium Standard Utilities Result Counts:\n\tRunning
 'action_progress_map': {'Completed.': ['Casus-Belli.local',
                                        'jtanium1.localdomain',
                                        'ubuntu.(none)',
                                        'localhost.(none)',
                                        'Jims-Mac.local',
                                        'WIN-A12SC6N6T7Q']},
 'action_results': <taniumpy.object_types.result_set.ResultSet object at 0x1025a2050>,
 'pre_action_question_results': {'question_object': <taniumpy.object_types.question.Question object
```

```
                                    'question_results': <taniumpy.object_types.result_set.ResultSet obj

Print of action object:
Action, name: 'API Deploy Distribute Tanium Standard Utilities'

CSV Results of response:
Action Statuses,Computer Name
46:Completed.,Casus-Belli.local
46:Completed.,jtanium1.localdomain
46:Completed.,ubuntu.(none)
46:Completed.,localhost.(none)
46:Completed.,Jims-Mac.local
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

```python
# Path to lib directory which contains pytan package
PYTAN_LIB_PATH = '../lib'

# connection info for Tanium Server
USERNAME = "Tanium User"
PASSWORD = "T@n!um"
HOST = "172.16.31.128"
PORT = "444"

# Logging conrols
LOGLEVEL = 2
DEBUGFORMAT = False

import sys, tempfile
sys.path.append(PYTAN_LIB_PATH)

import pytan
handler = pytan.Handler(
    username=USERNAME,
    password=PASSWORD,
    host=HOST,
    port=PORT,
    loglevel=LOGLEVEL,
    debugformat=DEBUGFORMAT,
)

print handler

# setup the arguments for the handler method
kwargs = {}
kwargs["get_results"] = False
kwargs["run"] = True
kwargs["package"] = u'Distribute Tanium Standard Utilities'

```

```
# call the handler with the deploy_action_human method, passing in kwargs for arguments
response = handler.deploy_action_human(**kwargs)
import pprint, io

print ""
print "Type of response: ", type(response)

print ""
print "Pretty print of response:"
print pprint.pformat(response)

print ""
print "Print of action object: "
print response['action_object']

# create an IO stream to store CSV results to
out = io.BytesIO()

# if results were returned (i.e. get_results=True was one of the kwargs passed in):
if response['action_results']:
    # call the write_csv() method to convert response to CSV and store it in out
    response['action_results'].write_csv(out, response['action_results'])

    print ""
    print "CSV Results of response: "
    print out.getvalue()
```

Output from Python Code

```
Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
2014-12-08 16:27:16,199 INFO     question_progress: Results 0% (Get Online = "True" from all machine
2014-12-08 16:27:21,218 INFO     question_progress: Results 100% (Get Online = "True" from all machi

Type of response:  <type 'dict'>

Pretty print of response:
{'action_object': <taniumpy.object_types.action.Action object at 0x10211d590>,
 'action_progress_human': None,
 'action_progress_map': None,
 'action_results': None,
 'pre_action_question_results': {'question_object': <taniumpy.object_types.question.Question object
                                 'question_results': <taniumpy.object_types.result_set.ResultSet obj

Print of action object:
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

```
# Path to lib directory which contains pytan package
PYTAN_LIB_PATH = '../lib'
```

```

# connection info for Tanium Server
USERNAME = "Tanium User"
PASSWORD = "T@n!um"
HOST = "172.16.31.128"
PORT = "444"

# Logging conrols
LOGLEVEL = 2
DEBUGFORMAT = False

import sys, tempfile
sys.path.append(PYTAN_LIB_PATH)

import pytan
handler = pytan.Handler(
    username=USERNAME,
    password=PASSWORD,
    host=HOST,
    port=PORT,
    loglevel=LOGLEVEL,
    debugformat=DEBUGFORMAT,
)

print handler

# setup the arguments for the handler method
kwargs = {}
kwargs["run"] = True
kwargs["action_filters"] = u'Operating System, that contains:Windows'
kwargs["package"] = u'Distribute Tanium Standard Utilities'

# call the handler with the deploy_action_human method, passing in kwargs for arguments
response = handler.deploy_action_human(**kwargs)
import pprint, io

print ""
print "Type of response: ", type(response)

print ""
print "Pretty print of response:"
print pprint.pformat(response)

print ""
print "Print of action object: "
print response['action_object']

# create an IO stream to store CSV results to
out = io.BytesIO()

# if results were returned (i.e. get_results=True was one of the kwargs passed in):
if response['action_results']:
    # call the write_csv() method to convert response to CSV and store it in out
    response['action_results'].write_csv(out, response['action_results'])

    print ""
    print "CSV Results of response: "
    print out.getvalue()
```

Output from Python Code

```
Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
2014-12-08 16:27:21,364 INFO     question_progress: Results 0% (Get Online = "True" from all machine
2014-12-08 16:27:26,389 INFO     question_progress: Results 0% (Get Online = "True" from all machine
2014-12-08 16:27:31,409 INFO     question_progress: Results 100% (Get Online = "True" from all machi
2014-12-08 16:27:31,496 INFO     action_progress: Action Results Passed: 0% (API Deploy Distribute T
2014-12-08 16:27:32,528 INFO     action_progress: Action Results Passed: 0% (API Deploy Distribute T
2014-12-08 16:27:33,561 INFO     action_progress: Action Results Passed: 0% (API Deploy Distribute T
2014-12-08 16:27:34,607 INFO     action_progress: Action Results Passed: 0% (API Deploy Distribute T
2014-12-08 16:27:35,649 INFO     action_progress: Action Results Passed: 0% (API Deploy Distribute T
2014-12-08 16:27:36,707 INFO     action_progress: Action Results Passed: 0% (API Deploy Distribute T
2014-12-08 16:27:37,764 INFO     action_progress: Action Results Passed: 0% (API Deploy Distribute T
2014-12-08 16:27:38,800 INFO     action_progress: Action Results Passed: 0% (API Deploy Distribute T
2014-12-08 16:27:39,830 INFO     action_progress: Action Results Passed: 0% (API Deploy Distribute T
2014-12-08 16:27:40,867 INFO     action_progress: Action Results Passed: 0% (API Deploy Distribute T
2014-12-08 16:27:41,904 INFO     action_progress: Action Results Passed: 0% (API Deploy Distribute T
2014-12-08 16:27:42,942 INFO     action_progress: Action Results Passed: 0% (API Deploy Distribute T
2014-12-08 16:27:43,986 INFO     action_progress: Action Results Passed: 0% (API Deploy Distribute T
2014-12-08 16:27:45,091 INFO     action_progress: Action Results Passed: 0% (API Deploy Distribute T
2014-12-08 16:27:46,143 INFO     action_progress: Action Results Passed: 0% (API Deploy Distribute T
2014-12-08 16:27:47,186 INFO     action_progress: Action Results Passed: 0% (API Deploy Distribute T
2014-12-08 16:27:48,222 INFO     action_progress: Action Results Passed: 0% (API Deploy Distribute T
2014-12-08 16:27:49,267 INFO     action_progress: Action Results Passed: 0% (API Deploy Distribute T
2014-12-08 16:27:50,316 INFO     action_progress: Action Results Passed: 0% (API Deploy Distribute T
2014-12-08 16:27:51,363 INFO     action_progress: Action Results Passed: 0% (API Deploy Distribute T
2014-12-08 16:27:52,475 INFO     action_progress: Action Results Passed: 0% (API Deploy Distribute T
2014-12-08 16:27:53,522 INFO     action_progress: Action Results Passed: 100% (API Deploy Distribute
2014-12-08 16:27:53,563 INFO     action_progress: Action Results Completed: 50% (API Deploy Distribu
2014-12-08 16:27:54,610 INFO     action_progress: Action Results Completed: 50% (API Deploy Distribu
2014-12-08 16:27:55,648 INFO     action_progress: Action Results Completed: 50% (API Deploy Distribu
2014-12-08 16:27:56,697 INFO     action_progress: Action Results Completed: 50% (API Deploy Distribu
2014-12-08 16:27:58,074 INFO     action_progress: Action Results Completed: 100% (API Deploy Distrib
2014-12-08 16:27:58,074 INFO     action_progress: API Deploy Distribute Tanium Standard Utilities Re
    Running Count: 0
    Success Count: 2
    Failed Count: 0
    Unknown Count: 0
    Finished Count: 2
    Total Count: 2
    Finished Count must equal: 2

Type of response:  <type 'dict'>

Pretty print of response:
{'action_object': <taniumpy.object_types.action.Action object at 0x102120410>,
 'action_progress_human': 'API Deploy Distribute Tanium Standard Utilities Result Counts:\n\tRunning
 'action_progress_map': {'Completed.': ['jtanium1.localdomain',
                                        'WIN-A12SC6N6T7Q']},
 'action_results': <taniumpy.object_types.result_set.ResultSet object at 0x102a047d0>,
 'pre_action_question_results': {'question_object': <taniumpy.object_types.question.Question object
                                 'question_results': <taniumpy.object_types.result_set.ResultSet obj

Print of action object:
Action, name: 'API Deploy Distribute Tanium Standard Utilities'
```

```

CSV Results of response:
Action Statuses,Computer Name
48:Completed.,jtanium1.localdomain
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

```
# Path to lib directory which contains pytan package
PYTAN_LIB_PATH = '../lib'

# connection info for Tanium Server
USERNAME = "Tanium User"
PASSWORD = "T@n!um"
HOST = "172.16.31.128"
PORT = "444"

# Logging conrols
LOGLEVEL = 2
DEBUGFORMAT = False

import sys, tempfile
sys.path.append(PYTAN_LIB_PATH)

import pytan
handler = pytan.Handler(
    username=USERNAME,
    password=PASSWORD,
    host=HOST,
    port=PORT,
    loglevel=LOGLEVEL,
    debugformat=DEBUGFORMAT,
)

print handler

# setup the arguments for the handler method
kwargs = {}
kwargs["run"] = True
kwargs["action_filters"] = u'Operating System, that contains:Windows'
kwargs["package"] = u'Custom Tagging - Add Tags{$1=tag_should_be_added,$2=tag_should_be_ignore}'

# call the handler with the deploy_action_human method, passing in kwargs for arguments
response = handler.deploy_action_human(**kwargs)
import pprint, io

print ""
print "Type of response: ", type(response)

print ""
print "Pretty print of response:"
print pprint.pformat(response)

print ""
print "Print of action object: "
print response['action_object']

# create an IO stream to store CSV results to
out = io.BytesIO()

# if results were returned (i.e. get_results=True was one of the kwargs passed in):
if response['action_results']:
    # call the write_csv() method to convert response to CSV and store it in out
    response['action_results'].write_csv(out, response['action_results'])

    print ""
    print "CSV Results of response: "
    print out.getvalue()
```

Output from Python Code

```
Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
2014-12-08 16:27:58,284 INFO     question_progress: Results 0% (Get Online = "True" from all machine
2014-12-08 16:28:03,310 INFO     question_progress: Results 100% (Get Online = "True" from all machi
2014-12-08 16:28:03,416 INFO     action_progress: Action Results Passed: 0% (API Deploy Custom Taggi
2014-12-08 16:28:04,454 INFO     action_progress: Action Results Passed: 0% (API Deploy Custom Taggi
2014-12-08 16:28:05,489 INFO     action_progress: Action Results Passed: 0% (API Deploy Custom Taggi
2014-12-08 16:28:06,521 INFO     action_progress: Action Results Passed: 0% (API Deploy Custom Taggi
2014-12-08 16:28:07,561 INFO     action_progress: Action Results Passed: 0% (API Deploy Custom Taggi
2014-12-08 16:28:08,602 INFO     action_progress: Action Results Passed: 0% (API Deploy Custom Taggi
2014-12-08 16:28:09,633 INFO     action_progress: Action Results Passed: 0% (API Deploy Custom Taggi
2014-12-08 16:28:10,667 INFO     action_progress: Action Results Passed: 0% (API Deploy Custom Taggi
2014-12-08 16:28:11,704 INFO     action_progress: Action Results Passed: 0% (API Deploy Custom Taggi
2014-12-08 16:28:12,738 INFO     action_progress: Action Results Passed: 0% (API Deploy Custom Taggi
2014-12-08 16:28:13,773 INFO     action_progress: Action Results Passed: 100% (API Deploy Custom Tag
2014-12-08 16:28:13,805 INFO     action_progress: Action Results Completed: 0% (API Deploy Custom Ta
2014-12-08 16:28:14,848 INFO     action_progress: Action Results Completed: 0% (API Deploy Custom Ta
2014-12-08 16:28:15,890 INFO     action_progress: Action Results Completed: 0% (API Deploy Custom Ta
2014-12-08 16:28:16,935 INFO     action_progress: Action Results Completed: 0% (API Deploy Custom Ta
2014-12-08 16:28:17,970 INFO     action_progress: Action Results Completed: 0% (API Deploy Custom Ta
2014-12-08 16:28:19,006 INFO     action_progress: Action Results Completed: 100% (API Deploy Custom
2014-12-08 16:28:19,006 INFO     action_progress: API Deploy Custom Tagging - Add Tags Result Counts
    Running Count: 0
    Success Count: 2
    Failed Count: 0
    Unknown Count: 0
    Finished Count: 2
    Total Count: 2
    Finished Count must equal: 2

Type of response:  <type 'dict'>

Pretty print of response:
{'action_object': <taniumpy.object_types.action.Action object at 0x1029798d0>,
 'action_progress_human': 'API Deploy Custom Tagging - Add Tags Result Counts:\n\tRunning Count: 0\n
 'action_progress_map': {'Completed.': ['jtanium1.localdomain',
                                                  'WIN-A12SC6N6T7Q']},
 'action_results': <taniumpy.object_types.result_set.ResultSet object at 0x102a05fd0>,
 'pre_action_question_results': {'question_object': <taniumpy.object_types.question.Question object
                                 'question_results': <taniumpy.object_types.result_set.ResultSet obj

Print of action object:
Action, name: 'API Deploy Custom Tagging - Add Tags'

CSV Results of response:
Action Statuses,Computer Name
49:Completed.,jtanium1.localdomain
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

```python
# Path to lib directory which contains pytan package
PYTAN_LIB_PATH = '../lib'

# connection info for Tanium Server
USERNAME = "Tanium User"
PASSWORD = "T@n!um"
HOST = "172.16.31.128"
PORT = "444"

# Logging conrols
LOGLEVEL = 2
DEBUGFORMAT = False

import sys, tempfile
sys.path.append(PYTAN_LIB_PATH)

import pytan
handler = pytan.Handler(
    username=USERNAME,
    password=PASSWORD,
    host=HOST,
    port=PORT,
    loglevel=LOGLEVEL,
    debugformat=DEBUGFORMAT,
)

print handler

# setup the arguments for the handler method
kwargs = {}
kwargs['report_dir'] = tempfile.gettempdir()
kwargs["package"] = u'Distribute Tanium Standard Utilities'

```

```

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

Output from Python Code

```
Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
2014-12-08 16:28:19,131 INFO     question_progress: Results 0% (Get Computer Name and Online = "True
2014-12-08 16:28:24,148 INFO     question_progress: Results 67% (Get Computer Name and Online = "Tru
2014-12-08 16:28:29,164 INFO     question_progress: Results 100% (Get Computer Name and Online = "Tr
2014-12-08 16:28:29,185 INFO     handler: Report file '/var/folders/dk/vjr1r_c53yx6k6gzp2bbt_c40000g
Traceback (most recent call last):
  File "<string>", line 39, in <module>
  File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 1172, in deploy_action_human
    **kwargs
  File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 1003, in deploy_action
    raise RunFalse(m(report_path, len(result)))
RunFalse: 'Run' is not True!!
View and verify the contents of /var/folders/dk/vjr1r_c53yx6k6gzp2bbt_c40000gn/T/VERIFY_BEFORE_DEPLO
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

```
# Path to lib directory which contains pytan package
PYTAN_LIB_PATH = '../lib'

# connection info for Tanium Server
USERNAME = "Tanium User"
PASSWORD = "T@n!um"
HOST = "172.16.31.128"
PORT = "444"

# Logging conrols
LOGLEVEL = 2
DEBUGFORMAT = False

import sys, tempfile
sys.path.append(PYTAN_LIB_PATH)

import pytan
handler = pytan.Handler(
    username=USERNAME,
    password=PASSWORD,
    host=HOST,
    port=PORT,
    loglevel=LOGLEVEL,
    debugformat=DEBUGFORMAT,
)

print handler

# setup the arguments for the handler method
kwargs = {}
kwargs['report_dir'] = tempfile.gettempdir()
kwargs["package_help"] = True


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

Output from Python Code

```
Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
Traceback (most recent call last):
  File "<string>", line 39, in <module>
  File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 1136, in deploy_action_human
    raise PytanHelp(utils.help_package())
PytanHelp:
Package Help
============

Supplying package defines what package will be deployed as part of the
action.

A package string is a human string that describes, at a minimum, a
package. It can also optionally define a selector for the package,
and/or parameters for the package. A package must be provided as a string.

Examples for package
---------------------------------

Supplying a package:

    'Distribute Tanium Standard Utilities'

Supplying a package by id:

    'id:1'

Supplying a package by hash:

    'hash:123456789'

Supplying a package by name:

    'name:Distribute Tanium Standard Utilities'

```

```
Package Parameters
------------------

Supplying parameters to a package can control the arguments
that are supplied to a package, if that package takes any arguments.

Package parameters must be surrounded with curly braces '{}',
and must have a key and value specified that is separated by
an equals '='. Multiple parameters must be seperated by
a comma ','. The key should match up to a valid parameter key
for the package in question.

If a parameter is supplied and the package doesn't have a
corresponding key name, it will be ignored. If the package has
parameters and a parameter is NOT supplied then an exception
will be raised, printing out the JSON of the missing paramater
for the package in question.

Examples for package with parameters
------------------------------------

Supplying a package with a single parameter '$1':

    'Package With Params{$1=value1}'

Supplying a package with two parameters, '$1' and '$2':

    'Package With Params{$1=value1,$2=value2}'
```

Invalid deploy action package

Deploy an action using a non-existing package.

Example Python Code

```python
# Path to lib directory which contains pytan package
PYTAN_LIB_PATH = '../lib'

# connection info for Tanium Server
USERNAME = "Tanium User"
PASSWORD = "T@n!um"
HOST = "172.16.31.128"
PORT = "444"

# Logging conrols
LOGLEVEL = 2
DEBUGFORMAT = False

import sys, tempfile
sys.path.append(PYTAN_LIB_PATH)

import pytan
handler = pytan.Handler(
    username=USERNAME,
    password=PASSWORD,
    host=HOST,
    port=PORT,
    loglevel=LOGLEVEL,
    debugformat=DEBUGFORMAT,
)

print handler

# setup the arguments for the handler method
kwargs = {}
kwargs['report_dir'] = tempfile.gettempdir()
kwargs["run"] = True
kwargs["package"] = u'Invalid Package'


# call the handler with the deploy_action_human method, passing in kwargs for arguments
# this should throw an exception: pytan.utils.HandlerError
import traceback
try:
    handler.deploy_action_human(**kwargs)
except Exception as e:
    traceback.print_exc(file=sys.stdout)
```

Output from Python Code

```
Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
Traceback (most recent call last):
  File "<string>", line 40, in <module>
  File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 1172, in deploy_action_human
    **kwargs
  File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 963, in deploy_action
    package_def = self._get_package_def(package_def)
  File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 1796, in _get_package_def
    d['package_obj'] = self.get('package', **def_search)[0]
  File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 1572, in get
    return self._get_single(obj_map, **kwargs)
  File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 1756, in _get_single
    for x in self._single_find(obj_map, k, v, **kwargs):
  File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 1766, in _single_find
    obj_ret = self._find(api_obj_single, **kwargs)
  File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 1694, in _find
    raise HandlerError(err(search_str))
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

```
# Path to lib directory which contains pytan package
PYTAN_LIB_PATH = '../lib'

# connection info for Tanium Server
```

```python
USERNAME = "Tanium User"
PASSWORD = "T@n!um"
HOST = "172.16.31.128"
PORT = "444"

# Logging conrols
LOGLEVEL = 2
DEBUGFORMAT = False

import sys, tempfile
sys.path.append(PYTAN_LIB_PATH)

import pytan
handler = pytan.Handler(
    username=USERNAME,
    password=PASSWORD,
    host=HOST,
    port=PORT,
    loglevel=LOGLEVEL,
    debugformat=DEBUGFORMAT,
)

print handler

# setup the arguments for the handler method
kwargs = {}
kwargs['report_dir'] = tempfile.gettempdir()
kwargs["options_help"] = True


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

Output from Python Code

```
Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
Traceback (most recent call last):
  File "<string>", line 39, in <module>
  File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 1142, in deploy_action_human
    raise PytanHelp(utils.help_options())
PytanHelp:
Options Help
============

Options are used for controlling how filters act. When options are
used as part of a sensor string, they change how the filters
supplied as part of that sensor operate. When options are used for
whole question options, they change how all of the question filters
operate.

When options are supplied for a sensor string, they must be
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'
```

```
        Help: Use 'or' for all of the filters supplied
        Usable on: group
        Example for sensor: "Sensor1, opt:or"
        Example for question: "or"

    '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"
        Example for question: "or"
```

Invalid deploy action empty package

Deploy an action using an empty package string.

Example Python Code

```python
# Path to lib directory which contains pytan package
PYTAN_LIB_PATH = '../lib'

# connection info for Tanium Server
USERNAME = "Tanium User"
PASSWORD = "T@n!um"
HOST = "172.16.31.128"
PORT = "444"

# Logging conrols
LOGLEVEL = 2
DEBUGFORMAT = False

import sys, tempfile
sys.path.append(PYTAN_LIB_PATH)

import pytan
handler = pytan.Handler(
    username=USERNAME,
    password=PASSWORD,
    host=HOST,
    port=PORT,
    loglevel=LOGLEVEL,
    debugformat=DEBUGFORMAT,
)

print handler

# setup the arguments for the handler method
kwargs = {}
kwargs['report_dir'] = tempfile.gettempdir()
kwargs["run"] = True
kwargs["package"] = u''


# call the handler with the deploy_action_human method, passing in kwargs for arguments
# this should throw an exception: pytan.utils.HumanParserError
import traceback
try:
    handler.deploy_action_human(**kwargs)
except Exception as e:
    traceback.print_exc(file=sys.stdout)
```

Output from Python Code

```
Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
Traceback (most recent call last):
  File "<string>", line 40, in <module>
  File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 1166, in deploy_action_human
    package_def = utils.dehumanize_package(package)
  File "/Users/jolsen/gh/pytan/lib/pytan/utils.py", line 1334, in dehumanize_package
    raise HumanParserError(err(package))
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

```python
# Path to lib directory which contains pytan package
PYTAN_LIB_PATH = '../lib'

# connection info for Tanium Server
USERNAME = "Tanium User"
PASSWORD = "T@n!um"
HOST = "172.16.31.128"
PORT = "444"

# Logging conrols
LOGLEVEL = 2
DEBUGFORMAT = False

import sys, tempfile
sys.path.append(PYTAN_LIB_PATH)

import pytan
handler = pytan.Handler(
    username=USERNAME,
    password=PASSWORD,
    host=HOST,
    port=PORT,
    loglevel=LOGLEVEL,
    debugformat=DEBUGFORMAT,
)

print handler

# setup the arguments for the handler method
kwargs = {}
kwargs['report_dir'] = tempfile.gettempdir()
kwargs["filters_help"] = True


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

Output from Python Code

```
Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
Traceback (most recent call last):
  File "<string>", line 39, in <module>
  File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 1139, in deploy_action_human
    raise PytanHelp(utils.help_filters())
PytanHelp:
Filters Help
============

Filters are used generously throughout pytan. When used as part of a
sensor string, they control what data is shown for the columns that
the sensor returns. When filters are used for whole question filters,
they control what rows will be returned. They are used by Groups to
define group membership, deploy actions to determine which machines
should have the action deployed to it, and more.

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

Valid Filters
-------------

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

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

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

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

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

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

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

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

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

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

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

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

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

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

    'not lessequal'
        Help: Filter for not less than or equal to VALUE
        Example: "Sensor1, that not lessequal:VALUE"

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

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

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

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

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

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

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

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

    '=>'
```

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

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

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

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

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

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

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

    '='
        Help: Filter for equals to VALUE
        Example: "Sensor1, that =:VALUE"

    'equal'
        Help: Filter for equals to VALUE
        Example: "Sensor1, that equal:VALUE"

    'equals'
        Help: Filter for equals to VALUE
        Example: "Sensor1, that equals:VALUE"

    'eq'
        Help: Filter for equals to VALUE
        Example: "Sensor1, that eq:VALUE"

    '!='
        Help: Filter for not equals to VALUE
        Example: "Sensor1, that !=:VALUE"

    'not equal'
        Help: Filter for not equals to VALUE
        Example: "Sensor1, that not equal:VALUE"

    'notequal'
        Help: Filter for not equals to VALUE
        Example: "Sensor1, that notequal:VALUE"

    'not equals'
        Help: Filter for not equals to VALUE
        Example: "Sensor1, that not equals:VALUE"

    'notequals'
        Help: Filter for not equals to VALUE
        Example: "Sensor1, that notequals:VALUE"

    'ne'
        Help: Filter for not equals to VALUE
        Example: "Sensor1, that ne:VALUE"

    'contains'
        Help: Filter for contains VALUE (adds .* before and after VALUE)
        Example: "Sensor1, that contains:VALUE"

    'does not contain'
        Help: Filter for does not contain VALUE (adds .* before and after VALUE)
        Example: "Sensor1, that does not contain:VALUE"

    'doesnotcontain'
        Help: Filter for does not contain VALUE (adds .* before and after VALUE)
        Example: "Sensor1, that doesnotcontain:VALUE"

    'not contains'
        Help: Filter for does not contain VALUE (adds .* before and after VALUE)
        Example: "Sensor1, that not contains:VALUE"

    'notcontains'
        Help: Filter for does not contain VALUE (adds .* before and after VALUE)
        Example: "Sensor1, that notcontains:VALUE"

    'starts with'
        Help: Filter for starts with VALUE (adds .* after VALUE)
        Example: "Sensor1, that starts with:VALUE"

    'startswith'
        Help: Filter for starts with VALUE (adds .* after VALUE)
        Example: "Sensor1, that startswith:VALUE"

    'does not start with'
        Help: Filter for does not start with VALUE (adds .* after VALUE)
        Example: "Sensor1, that does not start with:VALUE"

    'doesnotstartwith'
        Help: Filter for does not start with VALUE (adds .* after VALUE)
        Example: "Sensor1, that doesnotstartwith:VALUE"

    'not starts with'
        Help: Filter for does not start with VALUE (adds .* after VALUE)
        Example: "Sensor1, that not starts with:VALUE"

    'notstartswith'
        Help: Filter for does not start with VALUE (adds .* after VALUE)
        Example: "Sensor1, that notstartswith:VALUE"

    'ends with'
        Help: Filter for ends with VALUE (adds .* before VALUE)
        Example: "Sensor1, that ends with:VALUE"

    'endswith'
```

```
        Help: Filter for ends with VALUE (adds .* before VALUE)
        Example: "Sensor1, that endswith:VALUE"

    'does not end with'
        Help: Filter for does bit end with VALUE (adds .* before VALUE)
        Example: "Sensor1, that does not end with:VALUE"

    'doesnotendwith'
        Help: Filter for does bit end with VALUE (adds .* before VALUE)
        Example: "Sensor1, that doesnotendwith:VALUE"

    'not ends with'
        Help: Filter for does bit end with VALUE (adds .* before VALUE)
        Example: "Sensor1, that not ends with:VALUE"

    'notstartswith'
        Help: Filter for does bit end with VALUE (adds .* before VALUE)
        Example: "Sensor1, that notstartswith:VALUE"

    'is not'
        Help: Filter for non regular expression match for VALUE
        Example: "Sensor1, that is not:VALUE"

    'not regex'
        Help: Filter for non regular expression match for VALUE
        Example: "Sensor1, that not regex:VALUE"

    'notregex'
        Help: Filter for non regular expression match for VALUE
        Example: "Sensor1, that notregex:VALUE"

    'not regex match'
        Help: Filter for non regular expression match for VALUE
        Example: "Sensor1, that not regex match:VALUE"

    'notregexmatch'
        Help: Filter for non regular expression match for VALUE
        Example: "Sensor1, that notregexmatch:VALUE"

    'nre'
        Help: Filter for non regular expression match for VALUE
        Example: "Sensor1, that nre:VALUE"

    'is'
        Help: Filter for regular expression match for VALUE
        Example: "Sensor1, that is:VALUE"

    'regex'
        Help: Filter for regular expression match for VALUE
        Example: "Sensor1, that regex:VALUE"

    'regex match'
        Help: Filter for regular expression match for VALUE
        Example: "Sensor1, that regex match:VALUE"

    'regexmatch'
        Help: Filter for regular expression match for VALUE
        Example: "Sensor1, that regexmatch:VALUE"

    're'
        Help: Filter for regular expression match for VALUE
        Example: "Sensor1, that re:VALUE"

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

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

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

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

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

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

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

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

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

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

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

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

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

    'not less equal'
```

```
            Help: Filter for not less than or equal to VALUE
            Example: "Sensor1, that not less equal:VALUE"

        'not lessequal'
            Help: Filter for not less than or equal to VALUE
            Example: "Sensor1, that not lessequal:VALUE"

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

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

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

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

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

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

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

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

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

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

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

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

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

```

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

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

    '='
        Help: Filter for equals to VALUE
        Example: "Sensor1, that =:VALUE"

    'equal'
        Help: Filter for equals to VALUE
        Example: "Sensor1, that equal:VALUE"

    'equals'
        Help: Filter for equals to VALUE
        Example: "Sensor1, that equals:VALUE"

    'eq'
        Help: Filter for equals to VALUE
        Example: "Sensor1, that eq:VALUE"

    '!='
        Help: Filter for not equals to VALUE
        Example: "Sensor1, that !=:VALUE"

    'not equal'
        Help: Filter for not equals to VALUE
        Example: "Sensor1, that not equal:VALUE"

    'notequal'
        Help: Filter for not equals to VALUE
        Example: "Sensor1, that notequal:VALUE"

    'not equals'
        Help: Filter for not equals to VALUE
        Example: "Sensor1, that not equals:VALUE"

    'notequals'
        Help: Filter for not equals to VALUE
        Example: "Sensor1, that notequals:VALUE"

    'ne'
        Help: Filter for not equals to VALUE
        Example: "Sensor1, that ne:VALUE"

    'contains'
        Help: Filter for contains VALUE (adds .* before and after VALUE)
        Example: "Sensor1, that contains:VALUE"

    'does not contain'
        Help: Filter for does not contain VALUE (adds .* before and after VALUE)
        Example: "Sensor1, that does not contain:VALUE"

    'doesnotcontain'
```

```
            Help: Filter for does not contain VALUE (adds .* before and after VALUE)
            Example: "Sensor1, that doesnotcontain:VALUE"

        'not contains'
            Help: Filter for does not contain VALUE (adds .* before and after VALUE)
            Example: "Sensor1, that not contains:VALUE"

        'notcontains'
            Help: Filter for does not contain VALUE (adds .* before and after VALUE)
            Example: "Sensor1, that notcontains:VALUE"

        'starts with'
            Help: Filter for starts with VALUE (adds .* after VALUE)
            Example: "Sensor1, that starts with:VALUE"

        'startswith'
            Help: Filter for starts with VALUE (adds .* after VALUE)
            Example: "Sensor1, that startswith:VALUE"

        'does not start with'
            Help: Filter for does not start with VALUE (adds .* after VALUE)
            Example: "Sensor1, that does not start with:VALUE"

        'doesnotstartwith'
            Help: Filter for does not start with VALUE (adds .* after VALUE)
            Example: "Sensor1, that doesnotstartwith:VALUE"

        'not starts with'
            Help: Filter for does not start with VALUE (adds .* after VALUE)
            Example: "Sensor1, that not starts with:VALUE"

        'notstartswith'
            Help: Filter for does not start with VALUE (adds .* after VALUE)
            Example: "Sensor1, that notstartswith:VALUE"

        'ends with'
            Help: Filter for ends with VALUE (adds .* before VALUE)
            Example: "Sensor1, that ends with:VALUE"

        'endswith'
            Help: Filter for ends with VALUE (adds .* before VALUE)
            Example: "Sensor1, that endswith:VALUE"

        'does not end with'
            Help: Filter for does bit end with VALUE (adds .* before VALUE)
            Example: "Sensor1, that does not end with:VALUE"

        'doesnotendwith'
            Help: Filter for does bit end with VALUE (adds .* before VALUE)
            Example: "Sensor1, that doesnotendwith:VALUE"

        'not ends with'
            Help: Filter for does bit end with VALUE (adds .* before VALUE)
            Example: "Sensor1, that not ends with:VALUE"

        'notstartswith'
            Help: Filter for does bit end with VALUE (adds .* before VALUE)
            Example: "Sensor1, that notstartswith:VALUE"
```

```

    'is not'
        Help: Filter for non regular expression match for VALUE
        Example: "Sensor1, that is not:VALUE"

    'not regex'
        Help: Filter for non regular expression match for VALUE
        Example: "Sensor1, that not regex:VALUE"

    'notregex'
        Help: Filter for non regular expression match for VALUE
        Example: "Sensor1, that notregex:VALUE"

    'not regex match'
        Help: Filter for non regular expression match for VALUE
        Example: "Sensor1, that not regex match:VALUE"

    'notregexmatch'
        Help: Filter for non regular expression match for VALUE
        Example: "Sensor1, that notregexmatch:VALUE"

    'nre'
        Help: Filter for non regular expression match for VALUE
        Example: "Sensor1, that nre:VALUE"

    'is'
        Help: Filter for regular expression match for VALUE
        Example: "Sensor1, that is:VALUE"

    'regex'
        Help: Filter for regular expression match for VALUE
        Example: "Sensor1, that regex:VALUE"

    'regex match'
        Help: Filter for regular expression match for VALUE
        Example: "Sensor1, that regex match:VALUE"

    'regexmatch'
        Help: Filter for regular expression match for VALUE
        Example: "Sensor1, that regexmatch:VALUE"

    're'
        Help: Filter for regular expression match for VALUE
        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

```python
# Path to lib directory which contains pytan package
PYTAN_LIB_PATH = '../lib'

# connection info for Tanium Server
```

```
USERNAME = "Tanium User"
PASSWORD = "T@n!um"
HOST = "172.16.31.128"
PORT = "444"

# Logging conrols
LOGLEVEL = 2
DEBUGFORMAT = False

import sys, tempfile
sys.path.append(PYTAN_LIB_PATH)

import pytan
handler = pytan.Handler(
    username=USERNAME,
    password=PASSWORD,
    host=HOST,
    port=PORT,
    loglevel=LOGLEVEL,
    debugformat=DEBUGFORMAT,
)

print handler

# setup the arguments for the handler method
kwargs = {}
kwargs['report_dir'] = tempfile.gettempdir()
kwargs["run"] = True
kwargs["package"] = u'Custom Tagging - Add Tags'


# call the handler with the deploy_action_human method, passing in kwargs for arguments
# this should throw an exception: pytan.utils.HandlerError
import traceback
try:
    handler.deploy_action_human(**kwargs)
except Exception as e:
    traceback.print_exc(file=sys.stdout)
```

Output from Python Code

```
Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
2014-12-08 16:28:29,332 INFO     question_progress: Results 0% (Get Online = "True" from all machine
2014-12-08 16:28:34,353 INFO     question_progress: Results 50% (Get Online = "True" from all machin
2014-12-08 16:28:39,370 INFO     question_progress: Results 83% (Get Online = "True" from all machin
2014-12-08 16:28:44,391 INFO     question_progress: Results 100% (Get Online = "True" from all machi
Traceback (most recent call last):
  File "<string>", line 40, in <module>
  File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 1172, in deploy_action_human
    **kwargs
  File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 1026, in deploy_action
    empty_ok=False,
  File "/Users/jolsen/gh/pytan/lib/pytan/utils.py", line 2147, in build_param_objlist
    raise HandlerError(err(obj_name, p_key, jsonify(obj_param)))
HandlerError: PackageSpec, name: 'Custom Tagging - Add Tags' parameter key '$1' requires a value, pa
{
  "defaultValue": "",
  "helpString": "Enter tags space-delimited.",
  "key": "$1",
  "label": "Add tags (space-delimited)",
  "maxChars": 0,
  "model": "com.tanium.components.parameters::TextInputParameter",
  "parameterType": "com.tanium.components.parameters::TextInputParameter",
  "promptText": "e.g. PCI DMZ Decomm",
  "restrict": null,
  "validationExpressions": [
    {
      "expression": "\\S",
      "flags": "",
      "helpString": "You must enter a value",
      "model": "com.tanium.models::ValidationExpression",
      "parameterType": "com.tanium.models::ValidationExpression"
    }
  ],
  "value": ""
}
```

pytan API Valid Create Object Examples

Create user

Create a user called API Test User

Example Python Code

```
# Path to lib directory which contains pytan package
PYTAN_LIB_PATH = '../lib'

# connection info for Tanium Server
USERNAME = "Tanium User"
PASSWORD = "T@n!um"
HOST = "172.16.31.128"
PORT = "444"

# Logging conrols
LOGLEVEL = 2
DEBUGFORMAT = False

import sys, tempfile
sys.path.append(PYTAN_LIB_PATH)

import pytan
handler = pytan.Handler(
    username=USERNAME,
    password=PASSWORD,
    host=HOST,
    port=PORT,
    loglevel=LOGLEVEL,
    debugformat=DEBUGFORMAT,
)

print handler
```

```python

# setup the arguments for the delete method (to remove the package in case it exists)
delete_kwargs = {}
delete_kwargs["objtype"] = 'user'
delete_kwargs["name"] = 'API Test User'


# setup the arguments for the handler method
kwargs = {}
kwargs["username"] = u'API Test User'
kwargs["rolename"] = u'Administrator'
kwargs["properties"] = [[u'property1', u'value1']]

# delete the object in case it already exists
try:
    handler.delete(**delete_kwargs)
except Exception as e:
    print e

# call the handler with the create_user method, passing in kwargs for arguments
response = handler.create_user(**kwargs)


print ""
print "Type of response: ", type(response)

print ""
print "print of response:"
print response

print ""
print "print the object returned in JSON format:"
print response.to_json(response)

# delete the object, we are done with it now
try:
    handler.delete(**delete_kwargs)
except Exception as e:
    print e
```

Output from Python Code

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

Type of response:  <class 'taniumpy.object_types.user.User'>

print of response:
User, name: 'API Test User'

print the object returned in JSON format:
{
  "_type": "user",
  "deleted_flag": 0,
  "group_id": 0,
  "id": 16,
  "last_login": "2001-01-01T00:00:00",
  "metadata": {
    "_type": "metadata",
    "item": [
      {
        "_type": "item",
        "admin_flag": 0,
        "name": "TConsole.User.Property.property1",
        "value": "value1"
      }
    ]
  },
  "name": "API Test User",
  "permissions": {
    "_type": "permissions",
    "permission": "admin"
  },
  "roles": {
    "_type": "roles",
    "role": [
      {
        "_type": "role",
        "description": "Administrators can perform all functions in the system, including creating c
        "id": 1,
        "name": "Administrator",
        "permissions": {
          "_type": "permissions",
          "permission": "admin"
        }
      }
    ]
  }
}
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

```python
# Path to lib directory which contains pytan package
PYTAN_LIB_PATH = '../lib'

# connection info for Tanium Server
USERNAME = "Tanium User"
PASSWORD = "T@n!um"
HOST = "172.16.31.128"
PORT = "444"

# Logging conrols
LOGLEVEL = 2
DEBUGFORMAT = False

import sys, tempfile
```

```
sys.path.append(PYTAN_LIB_PATH)

import pytan
handler = pytan.Handler(
    username=USERNAME,
    password=PASSWORD,
    host=HOST,
    port=PORT,
    loglevel=LOGLEVEL,
    debugformat=DEBUGFORMAT,
)

print handler

# setup the arguments for the delete method (to remove the package in case it exists)
delete_kwargs = {}
delete_kwargs["objtype"] = 'package'
delete_kwargs["name"] = 'package49'


# setup the arguments for the handler method
kwargs = {}
kwargs["expire_seconds"] = 1500
kwargs["display_name"] = u'package49 API test'
kwargs["name"] = u'package49'
kwargs["parameters_json_file"] = u'../doc/example_of_all_package_parameters.json'
kwargs["verify_expire_seconds"] = 3600
kwargs["command"] = u'package49 $1 $2 $3 $4 $5 $6 $7 $8'
kwargs["file_urls"] = [u'3600::testing.vbs||https://content.tanium.com/files/initialcontent/bundles/
kwargs["verify_filter_options"] = [u'and']
kwargs["verify_filters"] = [u'Custom Tags, that contains:tag']
kwargs["command_timeout_seconds"] = 9999

# delete the object in case it already exists
try:
    handler.delete(**delete_kwargs)
except Exception as e:
    print e

# call the handler with the create_package method, passing in kwargs for arguments
response = handler.create_package(**kwargs)


print ""
print "Type of response: ", type(response)

print ""
print "print of response:"
print response

print ""
print "print the object returned in JSON format:"
print response.to_json(response)

# delete the object, we are done with it now
try:
    handler.delete(**delete_kwargs)
except Exception as e:
    print e
```

Output from Python Code

```
Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
No results found searching for PackageSpec, name: 'package49'!!
2014-12-08 16:28:44,534 INFO     handler: New package 'package49' created with ID 107, command: 'pac

Type of response:  <class 'taniumpy.object_types.package_spec.PackageSpec'>

print of response:
PackageSpec, name: 'package49'

print the object returned in JSON format:
{
  "_type": "package_spec",
  "available_time": "1900-01-01T00:00:00",
  "command": "package49 $1 $2 $3 $4 $5 $6 $7 $8",
  "command_timeout": 9999,
  "creation_time": "2014-12-08T21:28:44",
  "deleted_flag": 0,
  "display_name": "package49 API test",
  "expire_seconds": 1500,
  "files": {
    "_type": "package_files",
    "file": [
      {
        "_type": "file",
        "bytes_downloaded": 0,
        "bytes_total": 0,
        "cache_status": "UNCACHED",
        "download_seconds": 3600,
        "id": 235,
        "name": "testing.vbs",
        "size": 0,
        "source": "https://content.tanium.com/files/initialcontent/bundles/2014-10-01_11-32-15-7844/
        "status": 0
      }
    ]
  },
  "hidden_flag": 0,
  "id": 107,
  "last_modified_by": "Tanium User",
  "last_update": "2014-12-08T21:28:44",
  "modification_time": "2014-12-08T21:28:44",
  "name": "package49",
  "parameter_definition": "{\"parameterType\": \"com.tanium.components.parameters::ParametersArray\"
  "source_id": 0,
  "verify_group_id": 396
}
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

```python
# Path to lib directory which contains pytan package
PYTAN_LIB_PATH = '../lib'

# connection info for Tanium Server
USERNAME = "Tanium User"
PASSWORD = "T@n!um"
HOST = "172.16.31.128"
PORT = "444"

# Logging conrols
LOGLEVEL = 2
DEBUGFORMAT = False

import sys, tempfile
sys.path.append(PYTAN_LIB_PATH)

import pytan
handler = pytan.Handler(
    username=USERNAME,
    password=PASSWORD,
    host=HOST,
    port=PORT,
    loglevel=LOGLEVEL,
    debugformat=DEBUGFORMAT,
)

print handler

# setup the arguments for the delete method (to remove the package in case it exists)
delete_kwargs = {}
delete_kwargs["objtype"] = 'group'
delete_kwargs["name"] = 'All Windows Computers API Test'


# setup the arguments for the handler method
kwargs = {}
kwargs["groupname"] = u'All Windows Computers API Test'
kwargs["filters"] = [u'Operating System, that contains:Windows']
kwargs["filter_options"] = [u'and']

# delete the object in case it already exists
try:
    handler.delete(**delete_kwargs)
except Exception as e:
    print e

# call the handler with the create_group method, passing in kwargs for arguments
response = handler.create_group(**kwargs)


print ""
print "Type of response: ", type(response)

print ""
print "print of response:"
print response

print ""
print "print the object returned in JSON format:"
print response.to_json(response)

# delete the object, we are done with it now
try:
    handler.delete(**delete_kwargs)
except Exception as e:
    print e
```

Output from Python Code

```
Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
No results found searching for Group, name: 'All Windows Computers API Test'!!
2014-12-08 16:28:44,605 INFO     handler: New group 'All Windows Computers API Test' created with ID

Type of response:  <class 'taniumpy.object_types.group.Group'>

print of response:
Group, name: 'All Windows Computers API Test'

print the object returned in JSON format:
{
  "_type": "group",
  "and_flag": 1,
  "deleted_flag": 1,
  "filters": {
    "_type": "filters",
    "filter": [
      {
        "_type": "filter",
        "all_times_flag": 0,
        "all_values_flag": 0,
        "delimiter_index": 0,
        "ignore_case_flag": 1,
        "max_age_seconds": 0,
        "not_flag": 0,
        "operator": "RegexMatch",
        "sensor": {
          "_type": "sensor",
          "hash": 45421433
        },
        "substring_flag": 0,
        "substring_length": 0,
        "substring_start": 0,
        "utf8_flag": 0,
        "value": ".*Windows.*",
        "value_type": "String"
      }
    ]
  },
  "id": 397,
  "name": "All Windows Computers API Test",
  "not_flag": 0,
  "sub_groups": {
    "_type": "groups",
    "group": []
  },
  "text": " Operating System contains \"Windows\"",
  "type": 0
}
2014-12-08 16:28:44,619 INFO     handler: Deleted 'Group, id: 397'
```

Create whitelisted url

Create a whitelisted url

Example Python Code

```python
# Path to lib directory which contains pytan package
PYTAN_LIB_PATH = '../lib'

# connection info for Tanium Server
USERNAME = "Tanium User"
PASSWORD = "T@n!um"
HOST = "172.16.31.128"
PORT = "444"

# Logging conrols
LOGLEVEL = 2
DEBUGFORMAT = False

import sys, tempfile
sys.path.append(PYTAN_LIB_PATH)

import pytan
handler = pytan.Handler(
    username=USERNAME,
    password=PASSWORD,
    host=HOST,
    port=PORT,
    loglevel=LOGLEVEL,
    debugformat=DEBUGFORMAT,
)

print handler

# setup the arguments for the delete method (to remove the package in case it exists)
delete_kwargs = {}
delete_kwargs["objtype"] = 'whitelisted_url'
delete_kwargs["url_regex"] = 'regex:http://test.com/.*API_Test.*URL'


# setup the arguments for the handler method
kwargs = {}
kwargs["url"] = u'http://test.com/.*API_Test.*URL'
kwargs["regex"] = True
kwargs["properties"] = [[u'property1', u'value1']]
kwargs["download_seconds"] = 3600

# delete the object in case it already exists
try:
    handler.delete(**delete_kwargs)
except Exception as e:
    print e

# call the handler with the create_whitelisted_url method, passing in kwargs for arguments
response = handler.create_whitelisted_url(**kwargs)


print ""
print "Type of response: ", type(response)

print ""
print "print of response:"
print response

print ""
print "print the object returned in JSON format:"
print response.to_json(response)

# delete the object, we are done with it now
try:
    handler.delete(**delete_kwargs)
except Exception as e:
    print e
```

Output from Python Code

```
Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
No results found searching for whitelisted_url with {'url_regex': 'regex:http://test.com/.*API_Test.
2014-12-08 16:28:44,663 INFO     handler: New Whitelisted URL 'regex:http://test.com/.*API_Test.*URL

Type of response:  <class 'taniumpy.object_types.white_listed_url.WhiteListedUrl'>

print of response:
WhiteListedUrl, id: 52

print the object returned in JSON format:
{
  "_type": "white_listed_url",
  "download_seconds": 3600,
  "id": 52,
  "metadata": {
    "_type": "metadata",
    "item": [
      {
        "_type": "item",
        "admin_flag": 0,
        "name": "TConsole.WhitelistedURL.property1",
        "value": "value1"
      }
    ]
  },
  "url_regex": "regex:http://test.com/.*API_Test.*URL"
}
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

```python
# Path to lib directory which contains pytan package
PYTAN_LIB_PATH = '../lib'

# connection info for Tanium Server
USERNAME = "Tanium User"
PASSWORD = "T@n!um"
HOST = "172.16.31.128"
PORT = "444"

# Logging conrols
LOGLEVEL = 2
DEBUGFORMAT = False

import sys, tempfile
sys.path.append(PYTAN_LIB_PATH)

import pytan
handler = pytan.Handler(
    username=USERNAME,
    password=PASSWORD,
    host=HOST,
    port=PORT,
    loglevel=LOGLEVEL,
    debugformat=DEBUGFORMAT,
)

print handler

# setup the arguments for the handler method
kwargs = {}


# call the handler with the create_sensor method, passing in kwargs for arguments
# this should throw an exception: pytan.utils.HandlerError
import traceback
try:
    handler.create_sensor(**kwargs)
except Exception as e:
    traceback.print_exc(file=sys.stdout)
```

Output from Python Code

```
Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
Traceback (most recent call last):
  File "<string>", line 37, in <module>
  File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 537, in create_sensor
    raise HandlerError(m)
```

```
HandlerError: Sensor creation not supported via PyTan as of yet, too complex
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

```python
# Path to lib directory which contains pytan package
PYTAN_LIB_PATH = '../lib'

# connection info for Tanium Server
USERNAME = "Tanium User"
PASSWORD = "T@n!um"
HOST = "172.16.31.128"
PORT = "444"

# Logging conrols
LOGLEVEL = 2
DEBUGFORMAT = False

import sys, tempfile
sys.path.append(PYTAN_LIB_PATH)

import pytan
handler = pytan.Handler(
    username=USERNAME,
    password=PASSWORD,
    host=HOST,
    port=PORT,
    loglevel=LOGLEVEL,
    debugformat=DEBUGFORMAT,
)

print handler

# set the attribute name and value we want to add to the original object (if any)
attr_name = "name"
attr_add = " API TEST"

# delete object before creating it?
delete = True

# setup the arguments for getting an object to export as json file
get_kwargs = {}
get_kwargs["objtype"] = u'package'
get_kwargs["id"] = 31


# get objects to use as an export to JSON file
```

```python
orig_objs = handler.get(**get_kwargs)

# if attr_name and attr_add exists, modify the orig_objs to add attr_add to the attribute
# attr_name
if attr_name:
    for x in orig_objs:
        new_attr = getattr(x, attr_name)
        new_attr += attr_add
        setattr(x, attr_name, new_attr)
        if delete:
            # delete the object in case it already exists
            del_kwargs = {}
            del_kwargs[attr_name] = new_attr
            del_kwargs['objtype'] = u'package'
            try:
                handler.delete(**del_kwargs)
            except Exception as e:
                print e

# export orig_objs to a json file
json_file, results = handler.export_to_report_file(
    obj=orig_objs,
    export_format='json',
    report_dir=tempfile.gettempdir(),
)

# create the object from the exported JSON file
create_kwargs = {'objtype': u'package', 'json_file': json_file}
response = handler.create_from_json(**create_kwargs)


print ""
print "Type of response: ", type(response)

print ""
print "print of response:"
print response

print ""
print "print the object returned in JSON format:"
print response.to_json(response)
```

Output from Python Code

```
Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
2014-12-08 16:28:44,729 INFO     handler: Deleted 'PackageSpec, id: 101'
2014-12-08 16:28:44,730 INFO     handler: Report file '/var/folders/dk/vjr1r_c53yx6k6gzp2bbt_c40000g
2014-12-08 16:28:44,761 INFO     handler: New PackageSpec, name: 'Custom Tagging - Add Tags API TEST

Type of response:  <class 'taniumpy.object_types.package_spec_list.PackageSpecList'>

print of response:
PackageSpecList, len: 1

print the object returned in JSON format:
{
  "_type": "package_specs",
  "package_spec": [
    {
      "_type": "package_spec",
      "available_time": "1900-01-01T00:00:00",
      "command": "cmd /c cscript //T:60 add-tags.vbs \"$1\"",
      "command_timeout": 60,
      "creation_time": "2014-12-08T21:28:44",
      "deleted_flag": 0,
      "display_name": "Custom Tagging - Add Tags",
      "expire_seconds": 660,
      "files": {
        "_type": "package_files",
        "file": [
          {
            "_type": "file",
            "bytes_downloaded": 1972,
            "bytes_total": 1972,
            "cache_status": "CACHED",
            "download_seconds": 0,
            "download_start_time": "2014-12-08T19:23:55",
            "hash": "55aa6c54d82282ad2d41390e49f7b9939c582e14fa5cfca1b7b7fb9264261182",
            "id": 71,
            "last_download_progress_time": "2014-12-08T19:24:06",
            "name": "add-tags.vbs",
            "size": 1972,
            "source": "https://content.tanium.com/files/initialcontent/bundles/2014-11-05_12-56-07-8
            "status": 200
          }
        ]
      },
      "hidden_flag": 0,
      "id": 108,
      "last_modified_by": "Tanium User",
      "last_update": "2014-12-08T21:28:44",
      "metadata": {
        "_type": "metadata",
        "item": [
          {
            "_type": "item",
            "admin_flag": 0,
            "name": "defined",
            "value": "Tanium"
          },
          {
            "_type": "item",
            "admin_flag": 0,
            "name": "category",
            "value": "Tanium"
          }
        ]
      },
      "modification_time": "2014-12-08T21:28:44",
      "name": "Custom Tagging - Add Tags API TEST",
      "parameter_definition": "{\"parameters\":[{\"restrict\":null,\"validationExpressions\":[{\"hel
      "source_id": 0,
      "verify_group_id": 0
    }
  ]
```

```
}
```

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

```python
# Path to lib directory which contains pytan package
PYTAN_LIB_PATH = '../lib'

# connection info for Tanium Server
USERNAME = "Tanium User"
PASSWORD = "T@n!um"
HOST = "172.16.31.128"
PORT = "444"

# Logging conrols
LOGLEVEL = 2
DEBUGFORMAT = False

import sys, tempfile
sys.path.append(PYTAN_LIB_PATH)

import pytan
handler = pytan.Handler(
    username=USERNAME,
    password=PASSWORD,
    host=HOST,
    port=PORT,
    loglevel=LOGLEVEL,
    debugformat=DEBUGFORMAT,
)

print handler

# set the attribute name and value we want to add to the original object (if any)
attr_name = "name"
attr_add = " API TEST"

# delete object before creating it?
delete = True

# setup the arguments for getting an object to export as json file
get_kwargs = {}
get_kwargs["objtype"] = u'user'
get_kwargs["id"] = 1


# get objects to use as an export to JSON file
orig_objs = handler.get(**get_kwargs)

# if attr_name and attr_add exists, modify the orig_objs to add attr_add to the attribute
# attr_name
```

```
if attr_name:
    for x in orig_objs:
        new_attr = getattr(x, attr_name)
        new_attr += attr_add
        setattr(x, attr_name, new_attr)
        if delete:
            # delete the object in case it already exists
            del_kwargs = {}
            del_kwargs[attr_name] = new_attr
            del_kwargs['objtype'] = u'user'
            try:
                handler.delete(**del_kwargs)
            except Exception as e:
                print e

# export orig_objs to a json file
json_file, results = handler.export_to_report_file(
    obj=orig_objs,
    export_format='json',
    report_dir=tempfile.gettempdir(),
)

# create the object from the exported JSON file
create_kwargs = {'objtype': u'user', 'json_file': json_file}
response = handler.create_from_json(**create_kwargs)


print ""
print "Type of response: ", type(response)

print ""
print "print of response:"
print response

print ""
print "print the object returned in JSON format:"
print response.to_json(response)
```

Output from Python Code

```
Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
2014-12-08 16:28:44,799 INFO     handler: Deleted "User, name: 'Jim Olsen API TEST'"
2014-12-08 16:28:44,800 INFO     handler: Report file '/var/folders/dk/vjr1r_c53yx6k6gzp2bbt_c40000g
2014-12-08 16:28:44,816 INFO     handler: New User, name: 'Jim Olsen API TEST' (ID: 17) created succ

Type of response:  <class 'taniumpy.object_types.user_list.UserList'>

print of response:
UserList, len: 1

print the object returned in JSON format:
{
  "_type": "users",
  "user": [
    {
      "_type": "user",
      "deleted_flag": 0,
      "group_id": 0,
      "id": 17,
      "last_login": "2001-01-01T00:00:00",
      "name": "Jim Olsen API TEST",
      "permissions": {
        "_type": "permissions",
        "permission": "admin"
      },
      "roles": {
        "_type": "roles",
        "role": [
          {
            "_type": "role",
            "description": "Administrators can perform all functions in the system, including creati
            "id": 1,
            "name": "Administrator",
            "permissions": {
              "_type": "permissions",
              "permission": "admin"
            }
          }
        ]
      }
    }
  ]
}
```

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

```
# Path to lib directory which contains pytan package
PYTAN_LIB_PATH = '../lib'

# connection info for Tanium Server
USERNAME = "Tanium User"
PASSWORD = "T@n!um"
HOST = "172.16.31.128"
PORT = "444"

# Logging conrols
LOGLEVEL = 2
DEBUGFORMAT = False

import sys, tempfile
sys.path.append(PYTAN_LIB_PATH)

import pytan
handler = pytan.Handler(
    username=USERNAME,
    password=PASSWORD,
    host=HOST,
    port=PORT,
    loglevel=LOGLEVEL,
    debugformat=DEBUGFORMAT,
)

print handler

# set the attribute name and value we want to add to the original object (if any)
attr_name = "name"
attr_add = " API TEST"

# delete object before creating it?
delete = True

# setup the arguments for getting an object to export as json file
get_kwargs = {}
get_kwargs["objtype"] = u'saved_question'
get_kwargs["id"] = 1


# get objects to use as an export to JSON file
orig_objs = handler.get(**get_kwargs)

# if attr_name and attr_add exists, modify the orig_objs to add attr_add to the attribute
# attr_name
if attr_name:
    for x in orig_objs:
        new_attr = getattr(x, attr_name)
        new_attr += attr_add
        setattr(x, attr_name, new_attr)
        if delete:
            # delete the object in case it already exists
            del_kwargs = {}
            del_kwargs[attr_name] = new_attr
            del_kwargs['objtype'] = u'saved_question'
            try:
                handler.delete(**del_kwargs)
            except Exception as e:
                print e

# export orig_objs to a json file
json_file, results = handler.export_to_report_file(
    obj=orig_objs,
    export_format='json',
    report_dir=tempfile.gettempdir(),
)

# create the object from the exported JSON file
create_kwargs = {'objtype': u'saved_question', 'json_file': json_file}
response = handler.create_from_json(**create_kwargs)


print ""
print "Type of response: ", type(response)

print ""
print "print of response:"
print response

print ""
print "print the object returned in JSON format:"
print response.to_json(response)
```

Output from Python Code

```
Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
2014-12-08 16:28:44,876 INFO     handler: Deleted 'SavedQuestion, id: 178'
2014-12-08 16:28:44,877 INFO     handler: Report file '/var/folders/dk/vjr1r_c53yx6k6gzp2bbt_c40000g
2014-12-08 16:28:44,909 INFO     handler: New SavedQuestion, name: 'Run Unmanaged Asset Scan on All

Type of response:  <class 'taniumpy.object_types.saved_question_list.SavedQuestionList'>

print of response:
SavedQuestionList, len: 1

print the object returned in JSON format:
{
  "_type": "saved_questions",
  "saved_question": [
    {
      "_type": "saved_question",
      "action_tracking_flag": 0,
      "archive_enabled_flag": 0,
      "archive_owner": {
        "_type": "user"
      },
      "expire_seconds": 600,
      "hidden_flag": 0,
      "id": 180,
      "issue_seconds": 120,
      "issue_seconds_never_flag": 0,
      "keep_seconds": 0,
      "mod_time": "2000-01-01T00:00:00",
      "most_recent_question_id": 315,
      "name": "Run Unmanaged Asset Scan on All Machines API TEST",
      "packages": {
        "_type": "package_specs",
        "package_spec": []
      },
      "public_flag": 1,
      "query_text": "Get Is Windows from all machines",
      "question": {
        "_type": "question",
        "action_tracking_flag": 0,
        "expiration": "2014-12-08T20:32:42",
        "expire_seconds": 0,
        "force_computer_id_flag": 0,
        "hidden_flag": 0,
        "id": 315,
        "management_rights_group": {
          "_type": "group",
          "id": 0
        },
        "query_text": "Get Is Windows from all machines",
        "saved_question": {
          "_type": "saved_question",
          "id": 1
        },
        "selects": {
          "_type": "selects",
          "select": [
            {
              "_type": "select",
              "filter": {
                "_type": "filter",
                "all_times_flag": 0,
                "all_values_flag": 0,
                "delimiter_index": 0,
                "end_time": "2001-01-01T00:00:00",
                "ignore_case_flag": 1,
                "max_age_seconds": 0,
                "not_flag": 0,
                "operator": "Less",
                "start_time": "2001-01-01T00:00:00",
                "substring_flag": 0,
                "substring_length": 0,
                "substring_start": 0,
                "utf8_flag": 0,
                "value_type": "String"
              },
              "sensor": {
                "_type": "sensor",
                "category": "Operating System",
                "creation_time": "2014-12-08T19:20:40",
                "delimiter": ",",
                "description": "Returns whether the machine runs Windows.  True if so, False if not.
                "exclude_from_parse_flag": 0,
                "hash": 2721439124,
                "hidden_flag": 0,
                "id": 35,
                "ignore_case_flag": 1,
                "last_modified_by": "Jim Olsen",
                "max_age_seconds": 86400,
                "metadata": {
                  "_type": "metadata",
                  "item": [
                    {
                      "_type": "item",
                      "admin_flag": 0,
                      "name": "defined",
                      "value": "Tanium"
                    }
                  ]
                },
                "modification_time": "2014-12-08T19:20:40",
                "name": "Is Windows",
                "queries": {
                  "_type": "queries",
                  "query": [
                    {
                      "_type": "query",
                      "platform": "Windows",
                      "script": "&#039;==========================================\n&#039; Is Windows\n
                      "script_type": "VBScript"
                    },
                    {
                      "_type": "query",
                      "platform": "Linux",
                      "script": "#!/bin/bash\necho False\n",
                      "script_type": "UnixShell"
                    },
                    {
                      "_type": "query",
                      "platform": "Mac",
                      "script": "#!/bin/bash\necho False\n",
                      "script_type": "UnixShell"
                    }
                  ]
                },
                "source_id": 0,
                "string_count": 3,
                "value_type": "String"
              }
            }
          ]
        },
        "skip_lock_flag": 0,
        "user": {
          "_type": "user",
          "id": 1,
          "name": "Jim Olsen"
        }
      },
      "row_count_flag": 0,
      "sort_column": 0,
      "user": {
        "_type": "user",
        "id": 2,
        "name": "Tanium User"
      }
    }
  ]
}
```

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

```
# Path to lib directory which contains pytan package
PYTAN_LIB_PATH = '../lib'

# connection info for Tanium Server
USERNAME = "Tanium User"
PASSWORD = "T@n!um"
```

```
HOST = "172.16.31.128"
PORT = "444"

# Logging conrols
LOGLEVEL = 2
DEBUGFORMAT = False

import sys, tempfile
sys.path.append(PYTAN_LIB_PATH)

import pytan
handler = pytan.Handler(
    username=USERNAME,
    password=PASSWORD,
    host=HOST,
    port=PORT,
    loglevel=LOGLEVEL,
    debugformat=DEBUGFORMAT,
)

print handler

# set the attribute name and value we want to add to the original object (if any)
attr_name = ""
attr_add = ""

# delete object before creating it?
delete = False

# setup the arguments for getting an object to export as json file
get_kwargs = {}
get_kwargs["objtype"] = u'action'
get_kwargs["id"] = 1


# get objects to use as an export to JSON file
orig_objs = handler.get(**get_kwargs)

# if attr_name and attr_add exists, modify the orig_objs to add attr_add to the attribute
# attr_name
if attr_name:
    for x in orig_objs:
        new_attr = getattr(x, attr_name)
        new_attr += attr_add
        setattr(x, attr_name, new_attr)
        if delete:
            # delete the object in case it already exists
            del_kwargs = {}
            del_kwargs[attr_name] = new_attr
            del_kwargs['objtype'] = u'action'
            try:
                handler.delete(**del_kwargs)
            except Exception as e:
                print e

# export orig_objs to a json file
json_file, results = handler.export_to_report_file(
    obj=orig_objs,
```

```python
    export_format='json',
    report_dir=tempfile.gettempdir(),
)

# create the object from the exported JSON file
create_kwargs = {'objtype': u'action', 'json_file': json_file}
response = handler.create_from_json(**create_kwargs)


print ""
print "Type of response: ", type(response)

print ""
print "print of response:"
print response

print ""
print "print the object returned in JSON format:"
print response.to_json(response)
```

Output from Python Code

```
Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
2014-12-08 16:28:44,933 INFO     handler: Report file '/var/folders/dk/vjr1r_c53yx6k6gzp2bbt_c40000g
2014-12-08 16:28:45,017 INFO     handler: New Action, name: 'Unmanaged Asset Tracking - Run Scan' (I

Type of response:  <class 'taniumpy.object_types.action_list.ActionList'>

print of response:
ActionList, len: 1

print the object returned in JSON format:
{
  "_type": "actions",
  "action": [
    {
      "_type": "action",
      "action_group": {
        "_type": "group",
        "id": 0,
        "name": "Default"
      },
      "comment": "Scans for unmanaged assets on the network.",
      "creation_time": "2014-12-08T21:28:45",
      "distribute_seconds": 600,
      "expiration_time": "2014-12-08T22:18:45",
      "expire_seconds": 3000,
      "history_saved_question": {
        "_type": "saved_question",
        "id": 173
      },
      "id": 50,
      "name": "Unmanaged Asset Tracking - Run Scan",
      "package_spec": {
        "_type": "package_spec",
        "command": "cmd /c start /B cscript //T:3600 ..\\..\\Tools\\run-ua-scan.vbs /RANDOM_WAIT_TIM
        "id": 6,
        "name": "Run Unmanaged Asset Scanner"
      },
      "saved_action": {
        "_type": "saved_action",
        "id": 43
      },
      "skip_lock_flag": 0,
      "start_time": "2014-12-08T21:28:45",
      "status": "Active",
      "stopped_flag": 0,
      "target_group": {
        "_type": "group",
        "id": 65,
        "name": "Default"
      },
      "user": {
        "_type": "user",
        "group_id": 0,
        "id": 2,
        "last_login": "2014-12-08T21:28:45",
        "name": "Tanium User"
      }
    }
  ]
}
```

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

```python
# Path to lib directory which contains pytan package
PYTAN_LIB_PATH = '../lib'

# connection info for Tanium Server
USERNAME = "Tanium User"
PASSWORD = "T@n!um"
HOST = "172.16.31.128"
PORT = "444"

# Logging conrols
LOGLEVEL = 2
DEBUGFORMAT = False

import sys, tempfile
sys.path.append(PYTAN_LIB_PATH)

import pytan
handler = pytan.Handler(
    username=USERNAME,
    password=PASSWORD,
    host=HOST,
    port=PORT,
    loglevel=LOGLEVEL,
    debugformat=DEBUGFORMAT,
)

print handler

# set the attribute name and value we want to add to the original object (if any)
attr_name = "name"
attr_add = " API TEST"

# delete object before creating it?
delete = True

# setup the arguments for getting an object to export as json file
get_kwargs = {}
get_kwargs["objtype"] = u'sensor'
get_kwargs["id"] = 381


# get objects to use as an export to JSON file
orig_objs = handler.get(**get_kwargs)

# if attr_name and attr_add exists, modify the orig_objs to add attr_add to the attribute
# attr_name
if attr_name:
    for x in orig_objs:
        new_attr = getattr(x, attr_name)
        new_attr += attr_add
        setattr(x, attr_name, new_attr)
        if delete:
            # delete the object in case it already exists
            del_kwargs = {}
            del_kwargs[attr_name] = new_attr
            del_kwargs['objtype'] = u'sensor'
            try:
                handler.delete(**del_kwargs)
            except Exception as e:
                print e

# export orig_objs to a json file
json_file, results = handler.export_to_report_file(
    obj=orig_objs,
    export_format='json',
    report_dir=tempfile.gettempdir(),
)

# create the object from the exported JSON file
create_kwargs = {'objtype': u'sensor', 'json_file': json_file}
response = handler.create_from_json(**create_kwargs)


print ""
print "Type of response: ", type(response)

print ""
print "print of response:"
print response

print ""
print "print the object returned in JSON format:"
print response.to_json(response)
```

Output from Python Code

```
Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
2014-12-08 16:28:45,059 INFO     handler: Deleted 'Sensor, id: 827'
2014-12-08 16:28:45,060 INFO     handler: Report file '/var/folders/dk/vjr1r_c53yx6k6gzp2bbt_c40000g
2014-12-08 16:28:45,102 INFO     handler: New Sensor, name: 'Folder Name Search with RegEx Match API

Type of response:  <class 'taniumpy.object_types.sensor_list.SensorList'>

print of response:
SensorList, len: 1

print the object returned in JSON format:
{
  "_type": "sensors",
  "sensor": [
    {
      "_type": "sensor",
      "category": "File System",
      "creation_time": "2014-12-08T21:28:45",
      "delimiter": ",",
      "description": "Finds the specified folder and provides the full path if the folder exists on
      "exclude_from_parse_flag": 1,
      "hash": 839342978,
      "hidden_flag": 0,
      "id": 830,
      "ignore_case_flag": 1,
      "last_modified_by": "Tanium User",
      "max_age_seconds": 600,
      "metadata": {
        "_type": "metadata",
        "item": [
          {
            "_type": "item",
            "admin_flag": 0,
            "name": "defined",
            "value": "McAfee"
          }
        ]
      },
      "modification_time": "2014-12-08T21:28:45",
      "name": "Folder Name Search with RegEx Match API TEST",
      "parameter_definition": "{\"parameters\":[{\"restrict\":null,\"validationExpressions\":[{\"hel
      "queries": {
        "_type": "queries",
        "query": [
          {
            "_type": "query",
            "platform": "Windows",
            "script": "&amp;#039;=========================================\n&amp;#039; Folder Name Se
            "script_type": "VBScript"
          },
          {
            "_type": "query",
            "platform": "Linux",
            "script": "#!/bin/bash\n#||dirname||||regexp||||casesensitive||||global||\necho Windows
            "script_type": "UnixShell"
          },
          {
            "_type": "query",
            "platform": "Mac",
            "script": "#!/bin/bash\n#||dirname||||regexp||||casesensitive||||global||\necho Windows
            "script_type": "UnixShell"
          }
        ]
      },
      "source_id": 0,
      "string_count": 0,
      "value_type": "String"
    }
  ]
}
```

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

```
# Path to lib directory which contains pytan package
PYTAN_LIB_PATH = '../lib'

# connection info for Tanium Server
USERNAME = "Tanium User"
PASSWORD = "T@n!um"
HOST = "172.16.31.128"
PORT = "444"

# Logging conrols
LOGLEVEL = 2
DEBUGFORMAT = False

import sys, tempfile
sys.path.append(PYTAN_LIB_PATH)

import pytan
handler = pytan.Handler(
    username=USERNAME,
    password=PASSWORD,
    host=HOST,
    port=PORT,
    loglevel=LOGLEVEL,
    debugformat=DEBUGFORMAT,
)

print handler
```

```

# set the attribute name and value we want to add to the original object (if any)
attr_name = ""
attr_add = ""

# delete object before creating it?
delete = False

# setup the arguments for getting an object to export as json file
get_kwargs = {}
get_kwargs["objtype"] = u'question'
get_kwargs["id"] = 1


# get objects to use as an export to JSON file
orig_objs = handler.get(**get_kwargs)

# if attr_name and attr_add exists, modify the orig_objs to add attr_add to the attribute
# attr_name
if attr_name:
    for x in orig_objs:
        new_attr = getattr(x, attr_name)
        new_attr += attr_add
        setattr(x, attr_name, new_attr)
        if delete:
            # delete the object in case it already exists
            del_kwargs = {}
            del_kwargs[attr_name] = new_attr
            del_kwargs['objtype'] = u'question'
            try:
                handler.delete(**del_kwargs)
            except Exception as e:
                print e

# export orig_objs to a json file
json_file, results = handler.export_to_report_file(
    obj=orig_objs,
    export_format='json',
    report_dir=tempfile.gettempdir(),
)

# create the object from the exported JSON file
create_kwargs = {'objtype': u'question', 'json_file': json_file}
response = handler.create_from_json(**create_kwargs)


print ""
print "Type of response: ", type(response)

print ""
print "print of response:"
print response

print ""
print "print the object returned in JSON format:"
print response.to_json(response)
```

Output from Python Code

```
Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
2014-12-08 16:28:45,123 INFO     handler: Report file '/var/folders/dk/vjr1r_c53yx6k6gzp2bbt_c40000g
2014-12-08 16:28:45,159 INFO     handler: New Question, id: 437 (ID: 437) created successfully!

Type of response:  <class 'taniumpy.object_types.question_list.QuestionList'>

print of response:
QuestionList, len: 1

print the object returned in JSON format:
{
  "_type": "questions",
  "question": [
    {
      "_type": "question",
      "action_tracking_flag": 0,
      "context_group": {
        "_type": "group",
        "id": 0
      },
      "expiration": "2014-12-08T21:38:45",
      "expire_seconds": 0,
      "force_computer_id_flag": 1,
      "hidden_flag": 0,
      "id": 437,
      "management_rights_group": {
        "_type": "group",
        "id": 0
      },
      "query_text": "Get Action Statuses matches \"Nil\" from all machines",
      "saved_question": {
        "_type": "saved_question",
        "id": 0
      },
      "selects": {
        "_type": "selects",
        "select": [
          {
            "_type": "select",
            "filter": {
              "_type": "filter",
              "all_times_flag": 0,
              "all_values_flag": 1,
              "delimiter_index": 0,
              "end_time": "2001-01-01T00:00:00",
              "ignore_case_flag": 1,
              "max_age_seconds": 0,
              "not_flag": 0,
              "operator": "RegexMatch",
              "start_time": "2001-01-01T00:00:00",
              "substring_flag": 0,
              "substring_length": 0,
              "substring_start": 0,
              "utf8_flag": 0,
              "value": "Nil",
              "value_type": "String"
            },
            "sensor": {
              "_type": "sensor",
              "category": "Reserved",
              "description": "The recorded state of each action a client has taken recently in the f
              "exclude_from_parse_flag": 1,
              "hash": 1792443391,
              "hidden_flag": 0,
              "id": 1,
              "ignore_case_flag": 1,
              "max_age_seconds": 3600,
              "name": "Action Statuses",
              "queries": {
                "_type": "queries",
                "query": [
                  {
                    "_type": "query",
                    "platform": "Windows",
                    "script": "Reserved",
                    "script_type": "WMIQuery"
                  }
                ]
              },
              "source_id": 0,
              "string_count": 3540,
              "value_type": "String"
            }
          }
        ]
      },
      "skip_lock_flag": 0,
      "user": {
        "_type": "user",
        "id": 2,
        "name": "Tanium User"
      }
    }
  ]
}
```

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

```
# Path to lib directory which contains pytan package
PYTAN_LIB_PATH = '../lib'

# connection info for Tanium Server
USERNAME = "Tanium User"
PASSWORD = "T@n!um"
HOST = "172.16.31.128"
PORT = "444"
```

```

# Logging conrols
LOGLEVEL = 2
DEBUGFORMAT = False

import sys, tempfile
sys.path.append(PYTAN_LIB_PATH)

import pytan
handler = pytan.Handler(
    username=USERNAME,
    password=PASSWORD,
    host=HOST,
    port=PORT,
    loglevel=LOGLEVEL,
    debugformat=DEBUGFORMAT,
)

print handler

# set the attribute name and value we want to add to the original object (if any)
attr_name = "url_regex"
attr_add = " API TEST"

# delete object before creating it?
delete = True

# setup the arguments for getting an object to export as json file
get_kwargs = {}
get_kwargs["objtype"] = u'whitelisted_url'
get_kwargs["url_regex"] = u'test1'


# get objects to use as an export to JSON file
orig_objs = handler.get(**get_kwargs)

# if attr_name and attr_add exists, modify the orig_objs to add attr_add to the attribute
# attr_name
if attr_name:
    for x in orig_objs:
        new_attr = getattr(x, attr_name)
        new_attr += attr_add
        setattr(x, attr_name, new_attr)
        if delete:
            # delete the object in case it already exists
            del_kwargs = {}
            del_kwargs[attr_name] = new_attr
            del_kwargs['objtype'] = u'whitelisted_url'
            try:
                handler.delete(**del_kwargs)
            except Exception as e:
                print e

# export orig_objs to a json file
json_file, results = handler.export_to_report_file(
    obj=orig_objs,
    export_format='json',
    report_dir=tempfile.gettempdir(),
)

# create the object from the exported JSON file
create_kwargs = {'objtype': u'whitelisted_url', 'json_file': json_file}
response = handler.create_from_json(**create_kwargs)


print ""
print "Type of response: ", type(response)

print ""
print "print of response:"
print response

print ""
print "print the object returned in JSON format:"
print response.to_json(response)
```

Output from Python Code

```
Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
2014-12-08 16:28:45,212 INFO     handler: Deleted 'WhiteListedUrl, id: 27'
2014-12-08 16:28:45,213 INFO     handler: Report file '/var/folders/dk/vjr1r_c53yx6k6gzp2bbt_c40000g
2014-12-08 16:28:45,226 INFO     handler: New WhiteListedUrl, id: 53 (ID: 53) created successfully!

Type of response:  <class 'taniumpy.object_types.white_listed_url_list.WhiteListedUrlList'>

print of response:
WhiteListedUrlList, len: 1

print the object returned in JSON format:
{
  "_type": "white_listed_urls",
  "white_listed_url": [
    {
      "_type": "white_listed_url",
      "download_seconds": 86400,
      "id": 53,
      "url_regex": "test1 API TEST"
    }
  ]
}
```

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

```
# Path to lib directory which contains pytan package
PYTAN_LIB_PATH = '../lib'

```

```
# connection info for Tanium Server
USERNAME = "Tanium User"
PASSWORD = "T@n!um"
HOST = "172.16.31.128"
PORT = "444"

# Logging conrols
LOGLEVEL = 2
DEBUGFORMAT = False

import sys, tempfile
sys.path.append(PYTAN_LIB_PATH)

import pytan
handler = pytan.Handler(
    username=USERNAME,
    password=PASSWORD,
    host=HOST,
    port=PORT,
    loglevel=LOGLEVEL,
    debugformat=DEBUGFORMAT,
)

print handler

# set the attribute name and value we want to add to the original object (if any)
attr_name = "name"
attr_add = " API TEST"

# delete object before creating it?
delete = True

# setup the arguments for getting an object to export as json file
get_kwargs = {}
get_kwargs["objtype"] = u'group'
get_kwargs["name"] = u'All Computers'


# get objects to use as an export to JSON file
orig_objs = handler.get(**get_kwargs)

# if attr_name and attr_add exists, modify the orig_objs to add attr_add to the attribute
# attr_name
if attr_name:
    for x in orig_objs:
        new_attr = getattr(x, attr_name)
        new_attr += attr_add
        setattr(x, attr_name, new_attr)
        if delete:
            # delete the object in case it already exists
            del_kwargs = {}
            del_kwargs[attr_name] = new_attr
            del_kwargs['objtype'] = u'group'
            try:
                handler.delete(**del_kwargs)
            except Exception as e:
                print e

```

```
# export orig_objs to a json file
json_file, results = handler.export_to_report_file(
    obj=orig_objs,
    export_format='json',
    report_dir=tempfile.gettempdir(),
)

# create the object from the exported JSON file
create_kwargs = {'objtype': u'group', 'json_file': json_file}
response = handler.create_from_json(**create_kwargs)


print ""
print "Type of response: ", type(response)

print ""
print "print of response:"
print response

print ""
print "print the object returned in JSON format:"
print response.to_json(response)
```

Output from Python Code

```
Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
2014-12-08 16:28:45,264 INFO     handler: Deleted 'Group, id: 311'
2014-12-08 16:28:45,265 INFO     handler: Report file '/var/folders/dk/vjr1r_c53yx6k6gzp2bbt_c40000g
2014-12-08 16:28:45,289 INFO     handler: New Group, name: 'All Computers API TEST' (ID: 398) create

Type of response:  <class 'taniumpy.object_types.group_list.GroupList'>

print of response:
GroupList, len: 1

print the object returned in JSON format:
{
  "_type": "groups",
  "group": [
    {
      "_type": "group",
      "and_flag": 0,
      "deleted_flag": 1,
      "filters": {
        "_type": "filters",
        "filter": []
      },
      "id": 398,
      "name": "All Computers API TEST",
      "not_flag": 0,
      "sub_groups": {
        "_type": "groups",
        "group": []
      },
      "type": 0
    }
  ]
}
```

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

```python
# Path to lib directory which contains pytan package
PYTAN_LIB_PATH = '../lib'

# connection info for Tanium Server
USERNAME = "Tanium User"
PASSWORD = "T@n!um"
HOST = "172.16.31.128"
PORT = "444"

# Logging conrols
LOGLEVEL = 2
DEBUGFORMAT = False

import sys, tempfile
sys.path.append(PYTAN_LIB_PATH)

import pytan
handler = pytan.Handler(
    username=USERNAME,
    password=PASSWORD,
    host=HOST,
    port=PORT,
    loglevel=LOGLEVEL,
    debugformat=DEBUGFORMAT,
)

print handler

# setup the arguments for getting an object to export as json file
get_kwargs = {}
get_kwargs["objtype"] = u'saved_action'
get_kwargs["name"] = u'Distribute Tanium Standard Utilities'

# get objects to use as an export to JSON file
orig_objs = handler.get(**get_kwargs)

# export orig_objs to a json file
json_file, results = handler.export_to_report_file(
    obj=orig_objs,
    export_format='json',
    report_dir=tempfile.gettempdir(),
)

# call the handler with the create_from_json method, passing in kwargs for arguments
# this should throw an exception: pytan.utils.HandlerError
```

```python
import traceback

# create the object from the exported JSON file
create_kwargs = {'objtype': u'saved_action', 'json_file': json_file}
try:
    response = handler.create_from_json(**create_kwargs)
except Exception as e:
    traceback.print_exc(file=sys.stdout)
```

Output from Python Code

```
Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
2014-12-08 16:28:45,310 INFO     handler: Report file '/var/folders/dk/vjr1r_c53yx6k6gzp2bbt_c40000g
Traceback (most recent call last):
  File "<string>", line 51, in <module>
  File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 484, in create_from_json
    raise HandlerError(m(objtype, json_createable))
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

```python
# Path to lib directory which contains pytan package
PYTAN_LIB_PATH = '../lib'

# connection info for Tanium Server
USERNAME = "Tanium User"
PASSWORD = "T@n!um"
HOST = "172.16.31.128"
PORT = "444"

# Logging conrols
LOGLEVEL = 2
DEBUGFORMAT = False

import sys, tempfile
sys.path.append(PYTAN_LIB_PATH)

import pytan
handler = pytan.Handler(
    username=USERNAME,
    password=PASSWORD,
    host=HOST,
    port=PORT,
    loglevel=LOGLEVEL,
    debugformat=DEBUGFORMAT,
)

print handler

# setup the arguments for getting an object to export as json file
```

```python
get_kwargs = {}
get_kwargs["objtype"] = u'client'
get_kwargs["status"] = u'Leader'

# get objects to use as an export to JSON file
orig_objs = handler.get(**get_kwargs)

# export orig_objs to a json file
json_file, results = handler.export_to_report_file(
    obj=orig_objs,
    export_format='json',
    report_dir=tempfile.gettempdir(),
)

# call the handler with the create_from_json method, passing in kwargs for arguments
# this should throw an exception: pytan.utils.HandlerError
import traceback

# create the object from the exported JSON file
create_kwargs = {'objtype': u'client', 'json_file': json_file}
try:
    response = handler.create_from_json(**create_kwargs)
except Exception as e:
    traceback.print_exc(file=sys.stdout)
```

Output from Python Code

```
Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
2014-12-08 16:28:45,334 INFO     handler: Report file '/var/folders/dk/vjr1r_c53yx6k6gzp2bbt_c40000g
Traceback (most recent call last):
  File "<string>", line 51, in <module>
  File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 484, in create_from_json
    raise HandlerError(m(objtype, json_createable))
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

```python
# Path to lib directory which contains pytan package
PYTAN_LIB_PATH = '../lib'

# connection info for Tanium Server
USERNAME = "Tanium User"
PASSWORD = "T@n!um"
HOST = "172.16.31.128"
PORT = "444"

# Logging conrols
LOGLEVEL = 2
DEBUGFORMAT = False

```

```
import sys, tempfile
sys.path.append(PYTAN_LIB_PATH)

import pytan
handler = pytan.Handler(
    username=USERNAME,
    password=PASSWORD,
    host=HOST,
    port=PORT,
    loglevel=LOGLEVEL,
    debugformat=DEBUGFORMAT,
)

print handler

# setup the arguments for getting an object to export as json file
get_kwargs = {}
get_kwargs["objtype"] = u'userrole'
get_kwargs["name"] = u'Administrator'

# get objects to use as an export to JSON file
orig_objs = handler.get(**get_kwargs)

# export orig_objs to a json file
json_file, results = handler.export_to_report_file(
    obj=orig_objs,
    export_format='json',
    report_dir=tempfile.gettempdir(),
)

# call the handler with the create_from_json method, passing in kwargs for arguments
# this should throw an exception: pytan.utils.HandlerError
import traceback

# create the object from the exported JSON file
create_kwargs = {'objtype': u'userrole', 'json_file': json_file}
try:
    response = handler.create_from_json(**create_kwargs)
except Exception as e:
    traceback.print_exc(file=sys.stdout)
```

Output from Python Code

```
Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
2014-12-08 16:28:45,367 INFO     handler: Report file '/var/folders/dk/vjr1r_c53yx6k6gzp2bbt_c40000g
Traceback (most recent call last):
  File "<string>", line 51, in <module>
  File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 484, in create_from_json
    raise HandlerError(m(objtype, json_createable))
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

```python
# Path to lib directory which contains pytan package
PYTAN_LIB_PATH = '../lib'

# connection info for Tanium Server
USERNAME = "Tanium User"
PASSWORD = "T@n!um"
HOST = "172.16.31.128"
PORT = "444"

# Logging conrols
LOGLEVEL = 2
DEBUGFORMAT = False

import sys, tempfile
sys.path.append(PYTAN_LIB_PATH)

import pytan
handler = pytan.Handler(
    username=USERNAME,
    password=PASSWORD,
    host=HOST,
    port=PORT,
    loglevel=LOGLEVEL,
    debugformat=DEBUGFORMAT,
)

print handler

# setup the arguments for getting an object to export as json file
get_kwargs = {}
get_kwargs["objtype"] = u'setting'
get_kwargs["id"] = 1

# get objects to use as an export to JSON file
orig_objs = handler.get(**get_kwargs)

# export orig_objs to a json file
json_file, results = handler.export_to_report_file(
    obj=orig_objs,
    export_format='json',
    report_dir=tempfile.gettempdir(),
)

# call the handler with the create_from_json method, passing in kwargs for arguments
# this should throw an exception: pytan.utils.HandlerError
import traceback

# create the object from the exported JSON file
create_kwargs = {'objtype': u'setting', 'json_file': json_file}
try:
    response = handler.create_from_json(**create_kwargs)
except Exception as e:
    traceback.print_exc(file=sys.stdout)
```

Output from Python Code

```
Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
2014-12-08 16:28:45,385 INFO     handler: Report file '/var/folders/dk/vjr1r_c53yx6k6gzp2bbt_c40000g
Traceback (most recent call last):
  File "<string>", line 51, in <module>
  File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 484, in create_from_json
    raise HandlerError(m(objtype, json_createable))
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

```
# Path to lib directory which contains pytan package
PYTAN_LIB_PATH = '../lib'

# connection info for Tanium Server
USERNAME = "Tanium User"
PASSWORD = "T@n!um"
HOST = "172.16.31.128"
PORT = "444"

# Logging conrols
LOGLEVEL = 2
DEBUGFORMAT = False

import sys, tempfile
sys.path.append(PYTAN_LIB_PATH)

import pytan
handler = pytan.Handler(
    username=USERNAME,
    password=PASSWORD,
    host=HOST,
    port=PORT,
    loglevel=LOGLEVEL,
    debugformat=DEBUGFORMAT,
)

print handler

# setup the export_obj kwargs for later
export_kwargs = {}
export_kwargs["export_format"] = u'csv'

# ask the question that will provide the resultset that we want to use
ask_kwargs = {
    'qtype': 'manual_human',
    'sensors': [
        "Computer Name", "IP Route Details", "IP Address",
        'Folder Name Search with RegEx Match{dirname=Program Files,regex=.*Shared.*}',
    ],
}
response = handler.ask(**ask_kwargs)

# export the object to a string
# (we could just as easily export to a file using export_to_report_file)
export_kwargs['obj'] = response['question_results']
export_str = handler.export_obj(**export_kwargs)


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

print out
```

Output from Python Code

```
Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
2014-12-08 16:28:45,552 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:28:50,583 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:28:55,610 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:29:00,640 INFO     question_progress: Results 17% (Get Computer Name and IP Route Deta
2014-12-08 16:29:05,668 INFO     question_progress: Results 17% (Get Computer Name and IP Route Deta
2014-12-08 16:29:10,699 INFO     question_progress: Results 50% (Get Computer Name and IP Route Deta
2014-12-08 16:29:15,728 INFO     question_progress: Results 67% (Get Computer Name and IP Route Deta
2014-12-08 16:29:20,762 INFO     question_progress: Results 83% (Get Computer Name and IP Route Deta
2014-12-08 16:29:25,794 INFO     question_progress: Results 100% (Get Computer Name and IP Route Det

print the export_str returned from export_obj():
Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
2014-12-08 16:28:45,385 INFO     handler: Report file '/var/folders/dk/vjr1r_c53yx6k6gzp2bbt_c40000g
Traceback (most recent call last):
  File "<string>", line 51, in <module>
  File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 484, in create_from_json
    raise HandlerError(m(objtype, json_createable))
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

```
# Path to lib directory which contains pytan package
PYTAN_LIB_PATH = '../lib'

# connection info for Tanium Server
USERNAME = "Tanium User"
PASSWORD = "T@n!um"
HOST = "172.16.31.128"
PORT = "444"
```

```python

# Logging conrols
LOGLEVEL = 2
DEBUGFORMAT = False

import sys, tempfile
sys.path.append(PYTAN_LIB_PATH)

import pytan
handler = pytan.Handler(
    username=USERNAME,
    password=PASSWORD,
    host=HOST,
    port=PORT,
    loglevel=LOGLEVEL,
    debugformat=DEBUGFORMAT,
)

print handler

# setup the export_obj kwargs for later
export_kwargs = {}
export_kwargs["export_format"] = u'csv'
export_kwargs["expand_grouped_columns"] = False

# ask the question that will provide the resultset that we want to use
ask_kwargs = {
    'qtype': 'manual_human',
    'sensors': [
        "Computer Name", "IP Route Details", "IP Address",
        'Folder Name Search with RegEx Match{dirname=Program Files,regex=.*Shared.*}',
    ],
}
response = handler.ask(**ask_kwargs)

# export the object to a string
# (we could just as easily export to a file using export_to_report_file)
export_kwargs['obj'] = response['question_results']
export_str = handler.export_obj(**export_kwargs)


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

print out
```

Output from Python Code

```
Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
2014-12-08 16:29:25,976 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:29:31,003 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:29:36,035 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:29:41,062 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:29:46,088 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:29:51,126 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:29:56,153 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:30:01,183 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:30:06,213 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:30:11,240 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:30:16,269 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:30:21,306 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:30:26,331 INFO     question_progress: Results 33% (Get Computer Name and IP Route Deta
2014-12-08 16:30:31,362 INFO     question_progress: Results 100% (Get Computer Name and IP Route Det

print the export_str returned from export_obj():
Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
2014-12-08 16:28:45,552 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:28:50,583 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:28:55,610 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:29:00,640 INFO     question_progress: Results 17% (Get Computer Name and IP Route Deta
2014-12-08 16:29:05,668 INFO     question_progress: Results 17% (Get Computer Name and IP Route Deta
2014-12-08 16:29:10,699 INFO     question_progress: Results 50% (Get Computer Name and IP Route Deta
2014-12-08 16:29:15,728 INFO     question_progress: Results 67% (Get Computer Name and IP Route Deta
2014-12-08 16:29:20,762 INFO     question_progress: Results 83% (Get Computer Name and IP Route Deta
2014-12-08 16:29:25,794 INFO     question_progress: Results 100% (Get Computer Name and IP Route Det

print the export_str returned from export_obj():
Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
2014-12-08 16:28:45,385 INFO     handler: Report file '/var/folders/dk/vjr1r_c53yx6k6gzp2bbt_c40000g
Traceback (most recent call last):
..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

```python
# Path to lib directory which contains pytan package
PYTAN_LIB_PATH = '../lib'

# connection info for Tanium Server
USERNAME = "Tanium User"
PASSWORD = "T@n!um"
HOST = "172.16.31.128"
PORT = "444"

# Logging conrols
LOGLEVEL = 2
DEBUGFORMAT = False

import sys, tempfile
sys.path.append(PYTAN_LIB_PATH)

import pytan
handler = pytan.Handler(
    username=USERNAME,
    password=PASSWORD,
    host=HOST,
    port=PORT,
    loglevel=LOGLEVEL,
    debugformat=DEBUGFORMAT,
)

print handler

# setup the export_obj kwargs for later
export_kwargs = {}
export_kwargs["export_format"] = u'csv'
export_kwargs["expand_grouped_columns"] = True

# ask the question that will provide the resultset that we want to use
ask_kwargs = {
    'qtype': 'manual_human',
    'sensors': [
        "Computer Name", "IP Route Details", "IP Address",
        'Folder Name Search with RegEx Match{dirname=Program Files,regex=.*Shared.*}',
    ],
}
response = handler.ask(**ask_kwargs)

# export the object to a string
# (we could just as easily export to a file using export_to_report_file)
export_kwargs['obj'] = response['question_results']
export_str = handler.export_obj(**export_kwargs)


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

print out
```

Output from Python Code

```
Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
2014-12-08 16:30:31,572 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:30:36,599 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:30:41,628 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:30:46,653 INFO     question_progress: Results 17% (Get Computer Name and IP Route Deta
2014-12-08 16:30:51,682 INFO     question_progress: Results 50% (Get Computer Name and IP Route Deta
2014-12-08 16:30:56,710 INFO     question_progress: Results 100% (Get Computer Name and IP Route Det

print the export_str returned from export_obj():
Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
2014-12-08 16:29:25,976 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:29:31,003 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:29:36,035 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:29:41,062 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:29:46,088 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:29:51,126 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:29:56,153 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:30:01,183 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:30:06,213 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:30:11,240 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:30:16,269 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:30:21,306 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:30:26,331 INFO     question_progress: Results 33% (Get Computer Name and IP Route Deta
2014-12-08 16:30:31,362 INFO     question_progress: Results 100% (Get Computer Name and IP Route Det
..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

```python
# Path to lib directory which contains pytan package
PYTAN_LIB_PATH = '../lib'

# connection info for Tanium Server
USERNAME = "Tanium User"
PASSWORD = "T@n!um"
HOST = "172.16.31.128"
PORT = "444"

# Logging conrols
LOGLEVEL = 2
DEBUGFORMAT = False

import sys, tempfile
sys.path.append(PYTAN_LIB_PATH)

import pytan
handler = pytan.Handler(
    username=USERNAME,
    password=PASSWORD,
    host=HOST,
    port=PORT,
    loglevel=LOGLEVEL,
    debugformat=DEBUGFORMAT,
)

print handler

# setup the export_obj kwargs for later
export_kwargs = {}
export_kwargs["header_sort"] = True
export_kwargs["export_format"] = u'csv'
export_kwargs["header_add_type"] = True
export_kwargs["expand_grouped_columns"] = True
export_kwargs["header_add_sensor"] = True

# ask the question that will provide the resultset that we want to use
ask_kwargs = {
    'qtype': 'manual_human',
    'sensors': [
        "Computer Name", "IP Route Details", "IP Address",
        'Folder Name Search with RegEx Match{dirname=Program Files,regex=.*Shared.*}',
    ],
}
response = handler.ask(**ask_kwargs)

# export the object to a string
# (we could just as easily export to a file using export_to_report_file)
export_kwargs['obj'] = response['question_results']
export_str = handler.export_obj(**export_kwargs)


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

print out
```

Output from Python Code

```
Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
2014-12-08 16:30:56,943 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:31:01,971 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:31:07,002 INFO     question_progress: Results 33% (Get Computer Name and IP Route Deta
2014-12-08 16:31:12,030 INFO     question_progress: Results 67% (Get Computer Name and IP Route Deta
2014-12-08 16:31:17,058 INFO     question_progress: Results 83% (Get Computer Name and IP Route Deta
2014-12-08 16:31:22,085 INFO     question_progress: Results 83% (Get Computer Name and IP Route Deta
2014-12-08 16:31:27,114 INFO     question_progress: Results 83% (Get Computer Name and IP Route Deta
2014-12-08 16:31:32,138 INFO     question_progress: Results 83% (Get Computer Name and IP Route Deta
2014-12-08 16:31:37,164 INFO     question_progress: Results 100% (Get Computer Name and IP Route Det

print the export_str returned from export_obj():
Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
2014-12-08 16:30:31,572 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:30:36,599 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:30:41,628 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:30:46,653 INFO     question_progress: Results 17% (Get Computer Name and IP Route Deta
2014-12-08 16:30:51,682 INFO     question_progress: Results 50% (Get Computer Name and IP Route Deta
2014-12-08 16:30:56,710 INFO     question_progress: Results 100% (Get Computer Name and IP Route Det

print the export_str returned from export_obj():
Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
2014-12-08 16:29:25,976 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:29:31,003 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:29:36,035 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:29:41,062 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:29:46,088 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
..trimmed for brevity..
```

Export resultset json

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

Example Python Code

```python
# Path to lib directory which contains pytan package
PYTAN_LIB_PATH = '../lib'

# connection info for Tanium Server
USERNAME = "Tanium User"
PASSWORD = "T@n!um"
HOST = "172.16.31.128"
PORT = "444"

# Logging conrols
LOGLEVEL = 2
DEBUGFORMAT = False

import sys, tempfile
sys.path.append(PYTAN_LIB_PATH)

import pytan
handler = pytan.Handler(
    username=USERNAME,
    password=PASSWORD,
    host=HOST,
    port=PORT,
    loglevel=LOGLEVEL,
    debugformat=DEBUGFORMAT,
)

print handler

# setup the export_obj kwargs for later
export_kwargs = {}
export_kwargs["export_format"] = u'json'

# ask the question that will provide the resultset that we want to use
ask_kwargs = {
    'qtype': 'manual_human',
    'sensors': [
        "Computer Name", "IP Route Details", "IP Address",
        'Folder Name Search with RegEx Match{dirname=Program Files,regex=.*Shared.*}',
    ],
}
response = handler.ask(**ask_kwargs)

# export the object to a string
# (we could just as easily export to a file using export_to_report_file)
export_kwargs['obj'] = response['question_results']
export_str = handler.export_obj(**export_kwargs)


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

print out
```

Output from Python Code

```
Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
2014-12-08 16:31:37,503 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:31:42,529 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:31:47,559 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:31:52,587 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:31:57,615 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:32:02,640 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:32:07,665 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:32:12,693 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:32:17,719 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:32:22,747 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:32:27,775 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:32:32,805 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:32:37,831 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:32:42,858 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:32:47,887 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:32:52,922 INFO     question_progress: Results 17% (Get Computer Name and IP Route Deta
2014-12-08 16:32:57,948 INFO     question_progress: Results 50% (Get Computer Name and IP Route Deta
2014-12-08 16:33:02,976 INFO     question_progress: Results 100% (Get Computer Name and IP Route Det

print the export_str returned from export_obj():
Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
2014-12-08 16:30:56,943 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:31:01,971 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:31:07,002 INFO     question_progress: Results 33% (Get Computer Name and IP Route Deta
2014-12-08 16:31:12,030 INFO     question_progress: Results 67% (Get Computer Name and IP Route Deta
2014-12-08 16:31:17,058 INFO     question_progress: Results 83% (Get Computer Name and IP Route Deta
2014-12-08 16:31:22,085 INFO     question_progress: Results 83% (Get Computer Name and IP Route Deta
2014-12-08 16:31:27,114 INFO     question_progress: Results 83% (Get Computer Name and IP Route Deta
2014-12-08 16:31:32,138 INFO     question_progress: Results 83% (Get Computer Name and IP Route Deta
2014-12-08 16:31:37,164 INFO     question_progress: Results 100% (Get Computer Name and IP Route Det

print the export_str returned from export_obj():
Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
2014-12-08 16:30:31,572 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:30:36,599 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
..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

```
# Path to lib directory which contains pytan package
PYTAN_LIB_PATH = '../lib'

```

```
# connection info for Tanium Server
USERNAME = "Tanium User"
PASSWORD = "T@n!um"
HOST = "172.16.31.128"
PORT = "444"

# Logging conrols
LOGLEVEL = 2
DEBUGFORMAT = False

import sys, tempfile
sys.path.append(PYTAN_LIB_PATH)

import pytan
handler = pytan.Handler(
    username=USERNAME,
    password=PASSWORD,
    host=HOST,
    port=PORT,
    loglevel=LOGLEVEL,
    debugformat=DEBUGFORMAT,
)

print handler

# setup the export_obj kwargs for later
export_kwargs = {}
export_kwargs["export_format"] = u'csv'
export_kwargs["header_sort"] = []

# ask the question that will provide the resultset that we want to use
ask_kwargs = {
    'qtype': 'manual_human',
    'sensors': [
        "Computer Name", "IP Route Details", "IP Address",
        'Folder Name Search with RegEx Match{dirname=Program Files,regex=.*Shared.*}',
    ],
}
response = handler.ask(**ask_kwargs)

# export the object to a string
# (we could just as easily export to a file using export_to_report_file)
export_kwargs['obj'] = response['question_results']
export_str = handler.export_obj(**export_kwargs)


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

print out
```

Output from Python Code

```
Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
2014-12-08 16:33:03,175 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:33:08,203 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:33:13,228 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:33:18,257 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:33:23,282 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:33:28,308 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:33:33,335 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:33:38,362 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:33:43,390 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:33:48,418 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:33:53,454 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:33:58,481 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:34:03,510 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:34:08,539 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:34:13,572 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:34:18,603 INFO     question_progress: Results 67% (Get Computer Name and IP Route Deta
2014-12-08 16:34:23,630 INFO     question_progress: Results 83% (Get Computer Name and IP Route Deta
2014-12-08 16:34:28,657 INFO     question_progress: Results 83% (Get Computer Name and IP Route Deta
2014-12-08 16:34:33,686 INFO     question_progress: Results 83% (Get Computer Name and IP Route Deta
2014-12-08 16:34:38,716 INFO     question_progress: Results 100% (Get Computer Name and IP Route Det

print the export_str returned from export_obj():
Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
2014-12-08 16:31:37,503 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:31:42,529 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:31:47,559 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:31:52,587 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:31:57,615 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:32:02,640 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:32:07,665 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:32:12,693 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:32:17,719 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:32:22,747 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:32:27,775 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:32:32,805 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:32:37,831 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:32:42,858 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
..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

```python
# Path to lib directory which contains pytan package
PYTAN_LIB_PATH = '../lib'

# connection info for Tanium Server
USERNAME = "Tanium User"
PASSWORD = "T@n!um"
HOST = "172.16.31.128"
PORT = "444"

```

```python
# Logging conrols
LOGLEVEL = 2
DEBUGFORMAT = False

import sys, tempfile
sys.path.append(PYTAN_LIB_PATH)

import pytan
handler = pytan.Handler(
    username=USERNAME,
    password=PASSWORD,
    host=HOST,
    port=PORT,
    loglevel=LOGLEVEL,
    debugformat=DEBUGFORMAT,
)

print handler

# setup the export_obj kwargs for later
export_kwargs = {}
export_kwargs["export_format"] = u'csv'
export_kwargs["header_sort"] = True

# ask the question that will provide the resultset that we want to use
ask_kwargs = {
    'qtype': 'manual_human',
    'sensors': [
        "Computer Name", "IP Route Details", "IP Address",
        'Folder Name Search with RegEx Match{dirname=Program Files,regex=.*Shared.*}',
    ],
}
response = handler.ask(**ask_kwargs)

# export the object to a string
# (we could just as easily export to a file using export_to_report_file)
export_kwargs['obj'] = response['question_results']
export_str = handler.export_obj(**export_kwargs)


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

print out
```

Output from Python Code

```
Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
2014-12-08 16:34:38,921 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:34:43,950 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:34:48,987 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:34:54,015 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:34:59,046 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:35:04,073 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:35:09,098 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:35:14,126 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:35:19,155 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:35:24,182 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:35:29,213 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:35:34,241 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:35:39,272 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:35:44,296 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:35:49,327 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:35:54,353 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:35:59,383 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:36:04,419 INFO     question_progress: Results 50% (Get Computer Name and IP Route Deta
2014-12-08 16:36:09,449 INFO     question_progress: Results 50% (Get Computer Name and IP Route Deta
2014-12-08 16:36:14,473 INFO     question_progress: Results 50% (Get Computer Name and IP Route Deta
2014-12-08 16:36:19,502 INFO     question_progress: Results 67% (Get Computer Name and IP Route Deta
2014-12-08 16:36:24,534 INFO     question_progress: Results 100% (Get Computer Name and IP Route Det

print the export_str returned from export_obj():
Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
2014-12-08 16:33:03,175 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:33:08,203 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:33:13,228 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:33:18,257 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:33:23,282 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:33:28,308 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:33:33,335 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:33:38,362 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:33:43,390 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:33:48,418 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:33:53,454 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:33:58,481 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:34:03,510 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:34:08,539 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
..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

```
# Path to lib directory which contains pytan package
PYTAN_LIB_PATH = '../lib'

# connection info for Tanium Server
USERNAME = "Tanium User"
PASSWORD = "T@n!um"
HOST = "172.16.31.128"
PORT = "444"

# Logging conrols
LOGLEVEL = 2
DEBUGFORMAT = False

```

```
import sys, tempfile
sys.path.append(PYTAN_LIB_PATH)

import pytan
handler = pytan.Handler(
    username=USERNAME,
    password=PASSWORD,
    host=HOST,
    port=PORT,
    loglevel=LOGLEVEL,
    debugformat=DEBUGFORMAT,
)

print handler

# setup the export_obj kwargs for later
export_kwargs = {}
export_kwargs["export_format"] = u'csv'
export_kwargs["header_sort"] = False

# ask the question that will provide the resultset that we want to use
ask_kwargs = {
    'qtype': 'manual_human',
    'sensors': [
        "Computer Name", "IP Route Details", "IP Address",
        'Folder Name Search with RegEx Match{dirname=Program Files,regex=.*Shared.*}',
    ],
}
response = handler.ask(**ask_kwargs)

# export the object to a string
# (we could just as easily export to a file using export_to_report_file)
export_kwargs['obj'] = response['question_results']
export_str = handler.export_obj(**export_kwargs)


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

print out
```

Output from Python Code

```
Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
2014-12-08 16:36:24,732 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:36:29,758 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:36:34,793 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:36:39,821 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:36:44,845 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:36:49,878 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:36:54,905 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:36:59,941 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:37:04,967 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:37:09,995 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:37:15,020 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:37:20,047 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:37:25,076 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:37:30,103 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:37:35,131 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:37:40,159 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:37:45,193 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:37:50,224 INFO     question_progress: Results 50% (Get Computer Name and IP Route Deta
2014-12-08 16:37:55,251 INFO     question_progress: Results 83% (Get Computer Name and IP Route Deta
2014-12-08 16:38:00,284 INFO     question_progress: Results 100% (Get Computer Name and IP Route Det

print the export_str returned from export_obj():
Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
2014-12-08 16:34:38,921 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:34:43,950 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:34:48,987 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:34:54,015 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:34:59,046 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:35:04,073 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:35:09,098 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:35:14,126 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:35:19,155 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:35:24,182 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:35:29,213 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:35:34,241 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:35:39,272 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:35:44,296 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
..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

```python
# Path to lib directory which contains pytan package
PYTAN_LIB_PATH = '../lib'

# connection info for Tanium Server
USERNAME = "Tanium User"
PASSWORD = "T@n!um"
HOST = "172.16.31.128"
PORT = "444"

# Logging conrols
LOGLEVEL = 2
DEBUGFORMAT = False

import sys, tempfile
sys.path.append(PYTAN_LIB_PATH)

import pytan
handler = pytan.Handler(
    username=USERNAME,
    password=PASSWORD,
    host=HOST,
    port=PORT,
    loglevel=LOGLEVEL,
    debugformat=DEBUGFORMAT,
)

print handler

# setup the export_obj kwargs for later
export_kwargs = {}
export_kwargs["export_format"] = u'csv'
export_kwargs["header_sort"] = [u'Computer Name', u'IP Address']

# ask the question that will provide the resultset that we want to use
ask_kwargs = {
    'qtype': 'manual_human',
    'sensors': [
        "Computer Name", "IP Route Details", "IP Address",
        'Folder Name Search with RegEx Match{dirname=Program Files,regex=.*Shared.*}',
    ],
}
response = handler.ask(**ask_kwargs)

# export the object to a string
# (we could just as easily export to a file using export_to_report_file)
export_kwargs['obj'] = response['question_results']
export_str = handler.export_obj(**export_kwargs)


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

print out
```

Output from Python Code

```
Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
2014-12-08 16:38:00,467 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:38:05,497 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:38:10,524 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:38:15,551 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:38:20,578 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:38:25,604 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:38:30,628 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:38:35,653 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:38:40,679 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:38:45,708 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:38:50,735 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:38:55,768 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:39:00,797 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:39:05,823 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:39:10,845 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:39:15,876 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:39:20,904 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:39:25,930 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:39:30,955 INFO     question_progress: Results 83% (Get Computer Name and IP Route Deta
2014-12-08 16:39:35,982 INFO     question_progress: Results 83% (Get Computer Name and IP Route Deta
2014-12-08 16:39:41,010 INFO     question_progress: Results 100% (Get Computer Name and IP Route Det

print the export_str returned from export_obj():
Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
2014-12-08 16:36:24,732 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:36:29,758 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:36:34,793 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:36:39,821 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:36:44,845 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:36:49,878 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:36:54,905 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:36:59,941 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:37:04,967 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:37:09,995 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:37:15,020 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:37:20,047 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:37:25,076 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:37:30,103 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
..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

```python
# Path to lib directory which contains pytan package
PYTAN_LIB_PATH = '../lib'

# connection info for Tanium Server
USERNAME = "Tanium User"
PASSWORD = "T@n!um"
HOST = "172.16.31.128"
PORT = "444"

# Logging conrols
LOGLEVEL = 2
DEBUGFORMAT = False

import sys, tempfile
sys.path.append(PYTAN_LIB_PATH)

import pytan
handler = pytan.Handler(
    username=USERNAME,
    password=PASSWORD,
    host=HOST,
    port=PORT,
    loglevel=LOGLEVEL,
    debugformat=DEBUGFORMAT,
)

print handler

# setup the export_obj kwargs for later
export_kwargs = {}
export_kwargs["export_format"] = u'csv'
export_kwargs["header_add_type"] = False

# ask the question that will provide the resultset that we want to use
ask_kwargs = {
    'qtype': 'manual_human',
    'sensors': [
        "Computer Name", "IP Route Details", "IP Address",
        'Folder Name Search with RegEx Match{dirname=Program Files,regex=.*Shared.*}',
    ],
}
response = handler.ask(**ask_kwargs)

# export the object to a string
# (we could just as easily export to a file using export_to_report_file)
export_kwargs['obj'] = response['question_results']
export_str = handler.export_obj(**export_kwargs)


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

print out
```

Output from Python Code

```
Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
2014-12-08 16:39:41,204 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:39:46,233 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:39:51,260 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:39:56,286 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:40:01,313 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:40:06,340 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:40:11,373 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:40:16,400 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:40:21,427 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:40:26,453 INFO     question_progress: Results 50% (Get Computer Name and IP Route Deta
2014-12-08 16:40:31,481 INFO     question_progress: Results 83% (Get Computer Name and IP Route Deta
2014-12-08 16:40:36,509 INFO     question_progress: Results 83% (Get Computer Name and IP Route Deta
2014-12-08 16:40:41,536 INFO     question_progress: Results 83% (Get Computer Name and IP Route Deta
2014-12-08 16:40:46,562 INFO     question_progress: Results 83% (Get Computer Name and IP Route Deta
2014-12-08 16:40:51,588 INFO     question_progress: Results 83% (Get Computer Name and IP Route Deta
2014-12-08 16:40:56,613 INFO     question_progress: Results 83% (Get Computer Name and IP Route Deta
2014-12-08 16:41:01,643 INFO     question_progress: Results 83% (Get Computer Name and IP Route Deta
2014-12-08 16:41:06,670 INFO     question_progress: Results 83% (Get Computer Name and IP Route Deta
2014-12-08 16:41:11,694 INFO     question_progress: Results 83% (Get Computer Name and IP Route Deta
2014-12-08 16:41:16,719 INFO     question_progress: Results 83% (Get Computer Name and IP Route Deta
2014-12-08 16:41:21,745 INFO     question_progress: Results 83% (Get Computer Name and IP Route Deta
2014-12-08 16:41:26,778 INFO     question_progress: Results 83% (Get Computer Name and IP Route Deta
2014-12-08 16:41:31,805 INFO     question_progress: Results 83% (Get Computer Name and IP Route Deta
2014-12-08 16:41:36,836 INFO     question_progress: Results 83% (Get Computer Name and IP Route Deta
2014-12-08 16:41:41,863 INFO     question_progress: Results 100% (Get Computer Name and IP Route Det

print the export_str returned from export_obj():
Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
2014-12-08 16:38:00,467 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:38:05,497 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:38:10,524 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:38:15,551 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:38:20,578 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:38:25,604 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:38:30,628 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:38:35,653 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:38:40,679 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:38:45,708 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:38:50,735 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:38:55,768 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:39:00,797 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:39:05,823 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
..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

```python
# Path to lib directory which contains pytan package
PYTAN_LIB_PATH = '../lib'

# connection info for Tanium Server
USERNAME = "Tanium User"
PASSWORD = "T@n!um"
HOST = "172.16.31.128"
PORT = "444"

# Logging conrols
LOGLEVEL = 2
DEBUGFORMAT = False

import sys, tempfile
sys.path.append(PYTAN_LIB_PATH)

import pytan
handler = pytan.Handler(
    username=USERNAME,
    password=PASSWORD,
    host=HOST,
    port=PORT,
    loglevel=LOGLEVEL,
    debugformat=DEBUGFORMAT,
)
```

```

print handler

# setup the export_obj kwargs for later
export_kwargs = {}
export_kwargs["export_format"] = u'csv'
export_kwargs["header_add_type"] = True

# ask the question that will provide the resultset that we want to use
ask_kwargs = {
    'qtype': 'manual_human',
    'sensors': [
        "Computer Name", "IP Route Details", "IP Address",
        'Folder Name Search with RegEx Match{dirname=Program Files,regex=.*Shared.*}',
    ],
}
response = handler.ask(**ask_kwargs)

# export the object to a string
# (we could just as easily export to a file using export_to_report_file)
export_kwargs['obj'] = response['question_results']
export_str = handler.export_obj(**export_kwargs)


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

print out
```

Output from Python Code

```
Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
2014-12-08 16:41:42,048 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:41:47,073 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:41:52,100 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:41:57,128 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:42:02,151 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:42:07,176 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:42:12,200 INFO     question_progress: Results 33% (Get Computer Name and IP Route Deta
2014-12-08 16:42:17,226 INFO     question_progress: Results 100% (Get Computer Name and IP Route Det

print the export_str returned from export_obj():
Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
2014-12-08 16:39:41,204 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:39:46,233 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:39:51,260 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:39:56,286 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:40:01,313 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:40:06,340 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:40:11,373 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:40:16,400 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:40:21,427 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:40:26,453 INFO     question_progress: Results 50% (Get Computer Name and IP Route Deta
2014-12-08 16:40:31,481 INFO     question_progress: Results 83% (Get Computer Name and IP Route Deta
2014-12-08 16:40:36,509 INFO     question_progress: Results 83% (Get Computer Name and IP Route Deta
2014-12-08 16:40:41,536 INFO     question_progress: Results 83% (Get Computer Name and IP Route Deta
2014-12-08 16:40:46,562 INFO     question_progress: Results 83% (Get Computer Name and IP Route Deta
..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

```python
# Path to lib directory which contains pytan package
PYTAN_LIB_PATH = '../lib'

# connection info for Tanium Server
USERNAME = "Tanium User"
PASSWORD = "T@n!um"
HOST = "172.16.31.128"
PORT = "444"

# Logging conrols
LOGLEVEL = 2
DEBUGFORMAT = False

import sys, tempfile
sys.path.append(PYTAN_LIB_PATH)

import pytan
handler = pytan.Handler(
    username=USERNAME,
    password=PASSWORD,
    host=HOST,
    port=PORT,
    loglevel=LOGLEVEL,
    debugformat=DEBUGFORMAT,
)

print handler

# setup the export_obj kwargs for later
export_kwargs = {}
export_kwargs["export_format"] = u'csv'
export_kwargs["header_add_sensor"] = False

# ask the question that will provide the resultset that we want to use
ask_kwargs = {
    'qtype': 'manual_human',
    'sensors': [
        "Computer Name", "IP Route Details", "IP Address",
        'Folder Name Search with RegEx Match{dirname=Program Files,regex=.*Shared.*}',
    ],
}
response = handler.ask(**ask_kwargs)

```

```
# export the object to a string
# (we could just as easily export to a file using export_to_report_file)
export_kwargs['obj'] = response['question_results']
export_str = handler.export_obj(**export_kwargs)


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

print out
```

Output from Python Code

```
Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
2014-12-08 16:42:17,402 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:42:22,428 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:42:27,454 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:42:32,483 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:42:37,508 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:42:42,537 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:42:47,566 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:42:52,593 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:42:57,620 INFO     question_progress: Results 33% (Get Computer Name and IP Route Deta
2014-12-08 16:43:02,649 INFO     question_progress: Results 83% (Get Computer Name and IP Route Deta
2014-12-08 16:43:07,674 INFO     question_progress: Results 100% (Get Computer Name and IP Route Det

print the export_str returned from export_obj():
Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
2014-12-08 16:41:42,048 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:41:47,073 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:41:52,100 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:41:57,128 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:42:02,151 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:42:07,176 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:42:12,200 INFO     question_progress: Results 33% (Get Computer Name and IP Route Deta
2014-12-08 16:42:17,226 INFO     question_progress: Results 100% (Get Computer Name and IP Route Det

print the export_str returned from export_obj():
Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
2014-12-08 16:39:41,204 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:39:46,233 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:39:51,260 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
..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

```python
# Path to lib directory which contains pytan package
PYTAN_LIB_PATH = '../lib'

# connection info for Tanium Server
USERNAME = "Tanium User"
PASSWORD = "T@n!um"
HOST = "172.16.31.128"
PORT = "444"

# Logging conrols
LOGLEVEL = 2
DEBUGFORMAT = False

import sys, tempfile
sys.path.append(PYTAN_LIB_PATH)

import pytan
handler = pytan.Handler(
    username=USERNAME,
    password=PASSWORD,
    host=HOST,
    port=PORT,
    loglevel=LOGLEVEL,
    debugformat=DEBUGFORMAT,
)

print handler

# setup the export_obj kwargs for later
export_kwargs = {}
export_kwargs["export_format"] = u'csv'
export_kwargs["header_add_sensor"] = True

# ask the question that will provide the resultset that we want to use
ask_kwargs = {
    'qtype': 'manual_human',
    'sensors': [
        "Computer Name", "IP Route Details", "IP Address",
        'Folder Name Search with RegEx Match{dirname=Program Files,regex=.*Shared.*}',
    ],
}
response = handler.ask(**ask_kwargs)

# export the object to a string
# (we could just as easily export to a file using export_to_report_file)
export_kwargs['obj'] = response['question_results']
export_str = handler.export_obj(**export_kwargs)


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

print out
```

Output from Python Code

```
Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
2014-12-08 16:43:07,859 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:43:12,890 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:43:17,917 INFO     question_progress: Results 67% (Get Computer Name and IP Route Deta
2014-12-08 16:43:22,946 INFO     question_progress: Results 67% (Get Computer Name and IP Route Deta
2014-12-08 16:43:27,974 INFO     question_progress: Results 83% (Get Computer Name and IP Route Deta
2014-12-08 16:43:33,001 INFO     question_progress: Results 100% (Get Computer Name and IP Route Det

print the export_str returned from export_obj():
Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
2014-12-08 16:42:17,402 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:42:22,428 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:42:27,454 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:42:32,483 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:42:37,508 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:42:42,537 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:42:47,566 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:42:52,593 INFO     question_progress: Results 0% (Get Computer Name and IP Route Detai
2014-12-08 16:42:57,620 INFO     question_progress: Results 33% (Get Computer Name and IP Route Deta
2014-12-08 16:43:02,649 INFO     question_progress: Results 83% (Get Computer Name and IP Route Deta
2014-12-08 16:43:07,674 INFO     question_progress: Results 100% (Get Computer Name and IP Route Det

print the export_str returned from export_obj():
Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
..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

```python
# Path to lib directory which contains pytan package
PYTAN_LIB_PATH = '../lib'

# connection info for Tanium Server
USERNAME = "Tanium User"
PASSWORD = "T@n!um"
HOST = "172.16.31.128"
PORT = "444"

# Logging conrols
LOGLEVEL = 2
DEBUGFORMAT = False

import sys, tempfile
sys.path.append(PYTAN_LIB_PATH)

import pytan
handler = pytan.Handler(
    username=USERNAME,
    password=PASSWORD,
    host=HOST,
    port=PORT,
    loglevel=LOGLEVEL,
    debugformat=DEBUGFORMAT,
)

print handler

# setup the export_obj kwargs for later
export_kwargs = {}
export_kwargs["export_format"] = u'csv'
export_kwargs["header_sort"] = [[]]

# ask the question that will provide the resultset that we want to use
ask_kwargs = {
    'qtype': 'manual_human',
    'sensors': [
        "Computer Name"
    ],
}
response = handler.ask(**ask_kwargs)
export_kwargs['obj'] = response['question_results']

# export the object to a string
# this should throw an exception: pytan.utils.HandlerError
import traceback

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

Output from Python Code

```
Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
2014-12-08 16:43:33,188 INFO     question_progress: Results 0% (Get Computer Name from all machines)
2014-12-08 16:43:38,204 INFO     question_progress: Results 67% (Get Computer Name from all machines
2014-12-08 16:43:43,225 INFO     question_progress: Results 100% (Get Computer Name from all machine
Traceback (most recent call last):
  File "<string>", line 49, in <module>
  File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 1402, in export_obj
    utils.check_dictkey(**check_args)
  File "/Users/jolsen/gh/pytan/lib/pytan/utils.py", line 2516, in check_dictkey
    raise HandlerError(err(key, valid_list_types, list_types))
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

```
# Path to lib directory which contains pytan package
PYTAN_LIB_PATH = '../lib'
```

```

# connection info for Tanium Server
USERNAME = "Tanium User"
PASSWORD = "T@n!um"
HOST = "172.16.31.128"
PORT = "444"

# Logging conrols
LOGLEVEL = 2
DEBUGFORMAT = False

import sys, tempfile
sys.path.append(PYTAN_LIB_PATH)

import pytan
handler = pytan.Handler(
    username=USERNAME,
    password=PASSWORD,
    host=HOST,
    port=PORT,
    loglevel=LOGLEVEL,
    debugformat=DEBUGFORMAT,
)

print handler

# setup the export_obj kwargs for later
export_kwargs = {}
export_kwargs["export_format"] = u'csv'
export_kwargs["header_sort"] = u'bad'

# ask the question that will provide the resultset that we want to use
ask_kwargs = {
    'qtype': 'manual_human',
    'sensors': [
        "Computer Name"
    ],
}
response = handler.ask(**ask_kwargs)
export_kwargs['obj'] = response['question_results']

# export the object to a string
# this should throw an exception: pytan.utils.HandlerError
import traceback

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

Output from Python Code

```
Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
2014-12-08 16:43:43,303 INFO     question_progress: Results 0% (Get Computer Name from all machines)
2014-12-08 16:43:48,320 INFO     question_progress: Results 67% (Get Computer Name from all machines
2014-12-08 16:43:53,334 INFO     question_progress: Results 100% (Get Computer Name from all machine
Traceback (most recent call last):
  File "<string>", line 49, in <module>
  File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 1402, in export_obj
    utils.check_dictkey(**check_args)
  File "/Users/jolsen/gh/pytan/lib/pytan/utils.py", line 2509, in check_dictkey
    raise HandlerError(err(key, valid_types, k_type))
HandlerError: 'header_sort' must be one of [<type 'bool'>, <type 'list'>, <type 'tuple'>], you suppl
```

Invalid export resultset csv bad expand type

Export a ResultSet from asking a question using a bad expand_grouped_columns

Example Python Code

```python
# Path to lib directory which contains pytan package
PYTAN_LIB_PATH = '../lib'

# connection info for Tanium Server
USERNAME = "Tanium User"
PASSWORD = "T@n!um"
HOST = "172.16.31.128"
PORT = "444"

# Logging conrols
LOGLEVEL = 2
DEBUGFORMAT = False

import sys, tempfile
sys.path.append(PYTAN_LIB_PATH)

import pytan
handler = pytan.Handler(
    username=USERNAME,
    password=PASSWORD,
    host=HOST,
    port=PORT,
    loglevel=LOGLEVEL,
    debugformat=DEBUGFORMAT,
)

print handler

# setup the export_obj kwargs for later
export_kwargs = {}
export_kwargs["export_format"] = u'csv'
export_kwargs["expand_grouped_columns"] = u'bad'

# ask the question that will provide the resultset that we want to use
ask_kwargs = {
    'qtype': 'manual_human',
    'sensors': [
        "Computer Name"
    ],
}
response = handler.ask(**ask_kwargs)
export_kwargs['obj'] = response['question_results']
```

```python

# export the object to a string
# this should throw an exception: pytan.utils.HandlerError
import traceback

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

Output from Python Code

```
Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
2014-12-08 16:43:53,414 INFO     question_progress: Results 0% (Get Computer Name from all machines)
2014-12-08 16:43:58,431 INFO     question_progress: Results 0% (Get Computer Name from all machines)
2014-12-08 16:44:03,451 INFO     question_progress: Results 100% (Get Computer Name from all machine
Traceback (most recent call last):
  File "<string>", line 49, in <module>
  File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 1402, in export_obj
    utils.check_dictkey(**check_args)
  File "/Users/jolsen/gh/pytan/lib/pytan/utils.py", line 2509, in check_dictkey
    raise HandlerError(err(key, valid_types, k_type))
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

```python
# Path to lib directory which contains pytan package
PYTAN_LIB_PATH = '../lib'

# connection info for Tanium Server
USERNAME = "Tanium User"
PASSWORD = "T@n!um"
HOST = "172.16.31.128"
PORT = "444"

# Logging conrols
LOGLEVEL = 2
DEBUGFORMAT = False

import sys, tempfile
sys.path.append(PYTAN_LIB_PATH)

import pytan
handler = pytan.Handler(
    username=USERNAME,
    password=PASSWORD,
    host=HOST,
    port=PORT,
    loglevel=LOGLEVEL,
    debugformat=DEBUGFORMAT,
)

print handler

# setup the export_obj kwargs for later
export_kwargs = {}
export_kwargs["export_format"] = u'csv'
export_kwargs["sensors"] = [[]]
export_kwargs["header_add_sensor"] = True

# ask the question that will provide the resultset that we want to use
ask_kwargs = {
    'qtype': 'manual_human',
    'sensors': [
        "Computer Name"
    ],
}
response = handler.ask(**ask_kwargs)
export_kwargs['obj'] = response['question_results']

# export the object to a string
# this should throw an exception: pytan.utils.HandlerError
import traceback

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

Output from Python Code

```
Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
2014-12-08 16:44:03,549 INFO     question_progress: Results 0% (Get Computer Name from all machines)
2014-12-08 16:44:08,565 INFO     question_progress: Results 50% (Get Computer Name from all machines
2014-12-08 16:44:13,582 INFO     question_progress: Results 83% (Get Computer Name from all machines
2014-12-08 16:44:18,605 INFO     question_progress: Results 100% (Get Computer Name from all machine
Traceback (most recent call last):
  File "<string>", line 50, in <module>
  File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 1402, in export_obj
    utils.check_dictkey(**check_args)
  File "/Users/jolsen/gh/pytan/lib/pytan/utils.py", line 2516, in check_dictkey
    raise HandlerError(err(key, valid_list_types, list_types))
HandlerError: 'sensors' must be a list of [<class 'taniumpy.object_types.sensor.Sensor'>], you suppl
```

Invalid export resultset bad format

Export a ResultSet from asking a question using a bad export_format

Example Python Code

```python
# Path to lib directory which contains pytan package
PYTAN_LIB_PATH = '../lib'

# connection info for Tanium Server
```

```
USERNAME = "Tanium User"
PASSWORD = "T@n!um"
HOST = "172.16.31.128"
PORT = "444"

# Logging conrols
LOGLEVEL = 2
DEBUGFORMAT = False

import sys, tempfile
sys.path.append(PYTAN_LIB_PATH)

import pytan
handler = pytan.Handler(
    username=USERNAME,
    password=PASSWORD,
    host=HOST,
    port=PORT,
    loglevel=LOGLEVEL,
    debugformat=DEBUGFORMAT,
)

print handler

# setup the export_obj kwargs for later
export_kwargs = {}
export_kwargs["export_format"] = u'bad'

# ask the question that will provide the resultset that we want to use
ask_kwargs = {
    'qtype': 'manual_human',
    'sensors': [
        "Computer Name"
    ],
}
response = handler.ask(**ask_kwargs)
export_kwargs['obj'] = response['question_results']

# export the object to a string
# this should throw an exception: pytan.utils.HandlerError
import traceback

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

Output from Python Code

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

```
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

```python
# Path to lib directory which contains pytan package
PYTAN_LIB_PATH = '../lib'

# connection info for Tanium Server
USERNAME = "Tanium User"
PASSWORD = "T@n!um"
HOST = "172.16.31.128"
PORT = "444"

# Logging conrols
LOGLEVEL = 2
DEBUGFORMAT = False

import sys, tempfile
sys.path.append(PYTAN_LIB_PATH)

import pytan
handler = pytan.Handler(
    username=USERNAME,
    password=PASSWORD,
    host=HOST,
    port=PORT,
    loglevel=LOGLEVEL,
    debugformat=DEBUGFORMAT,
)

print handler

# setup the export_obj kwargs for later
export_kwargs = {}
export_kwargs["export_format"] = u'csv'

# get the objects that will provide the basetype that we want to use
get_kwargs = {
    'name': [
        "Computer Name", "IP Route Details", "IP Address",
        'Folder Name Search with RegEx Match',
    ],
    'objtype': 'sensor',
}
response = handler.get(**get_kwargs)

# export the object to a string
# (we could just as easily export to a file using export_to_report_file)
export_kwargs['obj'] = response
```

```
export_str = handler.export_obj(**export_kwargs)


print ""
print "print the export_str returned from export_obj():"

out = export_str
if len(out.splitlines()) > 15:
    out = out.splitlines()[0:15]
    out.append('..trimmed for brevity..')
    out = '\n'.join(out)

print out
```

Output from Python Code

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

print the export_str returned from export_obj():
category,creation_time,delimiter,description,exclude_from_parse_flag,hash,hidden_flag,id,ignore_case
Reserved,,,"The assigned name of the client machine.
Example: workstation-1.company.com",0,3409330187,0,3,1,,86400,,,,,Computer Name,,Windows,select CSNa
Network,2014-12-08T19:20:42,|,"Returns IPv4 network routes, filtered to exclude noise. With Flags, M
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
Set objWMIService = GetObject(&quot;winmgmts:&quot; _
    &amp; &quot;{impersonationLevel=impersonate}!\\&quot; &amp; strComputer &amp; &quot;\root\cimv2&

Set collip = objWMIService.ExecQuery(&quot;select * from win32_networkadapterconfiguration where IPE
dim ipaddrs()
ipcount = 0
for each ipItem in collip
    for each ipaddr in ipItem.IPAddress
        ipcount = ipcount + 1
    next
..trimmed for brevity..
```

Export basetype json type false

Export a BaseType from getting objects as JSON with false for include_type

Example Python Code

```
# Path to lib directory which contains pytan package
PYTAN_LIB_PATH = '../lib'

# connection info for Tanium Server
USERNAME = "Tanium User"
PASSWORD = "T@n!um"
HOST = "172.16.31.128"
PORT = "444"

# Logging conrols
LOGLEVEL = 2
DEBUGFORMAT = False
```

```

import sys, tempfile
sys.path.append(PYTAN_LIB_PATH)

import pytan
handler = pytan.Handler(
    username=USERNAME,
    password=PASSWORD,
    host=HOST,
    port=PORT,
    loglevel=LOGLEVEL,
    debugformat=DEBUGFORMAT,
)

print handler

# setup the export_obj kwargs for later
export_kwargs = {}
export_kwargs["export_format"] = u'json'
export_kwargs["include_type"] = False

# get the objects that will provide the basetype that we want to use
get_kwargs = {
    'name': [
        "Computer Name", "IP Route Details", "IP Address",
        'Folder Name Search with RegEx Match',
    ],
    'objtype': 'sensor',
}
response = handler.get(**get_kwargs)

# export the object to a string
# (we could just as easily export to a file using export_to_report_file)
export_kwargs['obj'] = response
export_str = handler.export_obj(**export_kwargs)


print ""
print "print the export_str returned from export_obj():"

out = export_str
if len(out.splitlines()) > 15:
    out = out.splitlines()[0:15]
    out.append('..trimmed for brevity..')
    out = '\n'.join(out)

print out
```

Output from Python Code

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

print the export_str returned from export_obj():
{
  "sensor": [
    {
      "category": "Reserved",
      "description": "The assigned name of the client machine.\nExample: workstation-1.company.com",
      "exclude_from_parse_flag": 0,
      "hash": 3409330187,
      "hidden_flag": 0,
      "id": 3,
      "ignore_case_flag": 1,
      "max_age_seconds": 86400,
      "name": "Computer Name",
      "queries": {
        "query": [
          {
..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

```python
# Path to lib directory which contains pytan package
PYTAN_LIB_PATH = '../lib'

# connection info for Tanium Server
USERNAME = "Tanium User"
PASSWORD = "T@n!um"
HOST = "172.16.31.128"
PORT = "444"

# Logging conrols
LOGLEVEL = 2
DEBUGFORMAT = False

import sys, tempfile
sys.path.append(PYTAN_LIB_PATH)

import pytan
handler = pytan.Handler(
    username=USERNAME,
    password=PASSWORD,
    host=HOST,
    port=PORT,
    loglevel=LOGLEVEL,
    debugformat=DEBUGFORMAT,
)

print handler

# setup the export_obj kwargs for later
export_kwargs = {}
export_kwargs["export_format"] = u'json'
export_kwargs["explode_json_string_values"] = False

# get the objects that will provide the basetype that we want to use
get_kwargs = {
    'name': [
        "Computer Name", "IP Route Details", "IP Address",
        'Folder Name Search with RegEx Match',
    ],
    'objtype': 'sensor',
}
response = handler.get(**get_kwargs)

# export the object to a string
# (we could just as easily export to a file using export_to_report_file)
export_kwargs['obj'] = response
export_str = handler.export_obj(**export_kwargs)


print ""
print "print the export_str returned from export_obj():"

out = export_str
if len(out.splitlines()) > 15:
    out = out.splitlines()[0:15]
    out.append('..trimmed for brevity..')
    out = '\n'.join(out)

print out
```

Output from Python Code

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

print the export_str returned from export_obj():
{
  "_type": "sensors",
  "sensor": [
    {
      "_type": "sensor",
      "category": "Reserved",
      "description": "The assigned name of the client machine.\nExample: workstation-1.company.com",
      "exclude_from_parse_flag": 0,
      "hash": 3409330187,
      "hidden_flag": 0,
      "id": 3,
      "ignore_case_flag": 1,
      "max_age_seconds": 86400,
      "name": "Computer Name",
      "queries": {
..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

```
# Path to lib directory which contains pytan package
PYTAN_LIB_PATH = '../lib'
```

```

# connection info for Tanium Server
USERNAME = "Tanium User"
PASSWORD = "T@n!um"
HOST = "172.16.31.128"
PORT = "444"

# Logging conrols
LOGLEVEL = 2
DEBUGFORMAT = False

import sys, tempfile
sys.path.append(PYTAN_LIB_PATH)

import pytan
handler = pytan.Handler(
    username=USERNAME,
    password=PASSWORD,
    host=HOST,
    port=PORT,
    loglevel=LOGLEVEL,
    debugformat=DEBUGFORMAT,
)

print handler

# setup the export_obj kwargs for later
export_kwargs = {}
export_kwargs["export_format"] = u'json'
export_kwargs["explode_json_string_values"] = True

# get the objects that will provide the basetype that we want to use
get_kwargs = {
    'name': [
        "Computer Name", "IP Route Details", "IP Address",
        'Folder Name Search with RegEx Match',
    ],
    'objtype': 'sensor',
}
response = handler.get(**get_kwargs)

# export the object to a string
# (we could just as easily export to a file using export_to_report_file)
export_kwargs['obj'] = response
export_str = handler.export_obj(**export_kwargs)


print ""
print "print the export_str returned from export_obj():"

out = export_str
if len(out.splitlines()) > 15:
    out = out.splitlines()[0:15]
    out.append('..trimmed for brevity..')
    out = '\n'.join(out)

print out
```

Output from Python Code

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

print the export_str returned from export_obj():
{
  "_type": "sensors",
  "sensor": [
    {
      "_type": "sensor",
      "category": "Reserved",
      "description": "The assigned name of the client machine.\nExample: workstation-1.company.com",
      "exclude_from_parse_flag": 0,
      "hash": 3409330187,
      "hidden_flag": 0,
      "id": 3,
      "ignore_case_flag": 1,
      "max_age_seconds": 86400,
      "name": "Computer Name",
      "queries": {
..trimmed for brevity..
```

Export basetype xml default options

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

Example Python Code

```python
# Path to lib directory which contains pytan package
PYTAN_LIB_PATH = '../lib'

# connection info for Tanium Server
USERNAME = "Tanium User"
PASSWORD = "T@n!um"
HOST = "172.16.31.128"
PORT = "444"

# Logging conrols
LOGLEVEL = 2
DEBUGFORMAT = False

import sys, tempfile
sys.path.append(PYTAN_LIB_PATH)

import pytan
handler = pytan.Handler(
    username=USERNAME,
    password=PASSWORD,
    host=HOST,
    port=PORT,
    loglevel=LOGLEVEL,
    debugformat=DEBUGFORMAT,
)

print handler

```

```
# setup the export_obj kwargs for later
export_kwargs = {}
export_kwargs["export_format"] = u'xml'

# get the objects that will provide the basetype that we want to use
get_kwargs = {
    'name': [
        "Computer Name", "IP Route Details", "IP Address",
        'Folder Name Search with RegEx Match',
    ],
    'objtype': 'sensor',
}
response = handler.get(**get_kwargs)

# export the object to a string
# (we could just as easily export to a file using export_to_report_file)
export_kwargs['obj'] = response
export_str = handler.export_obj(**export_kwargs)


print ""
print "print the export_str returned from export_obj():"

out = export_str
if len(out.splitlines()) > 15:
    out = out.splitlines()[0:15]
    out.append('..trimmed for brevity..')
    out = '\n'.join(out)

print out
```

Output from Python Code

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

print the export_str returned from export_obj():
<sensors><cache_info /><sensor><category>Reserved</category><preview_sensor_flag /><hash>3409330187<
Example: workstation-1.company.com</description><string_hints /><subcolumns /><metadata /><parameter
Example:  172.16.0.0|192.168.1.1|255.255.0.0|UG|100|eth0</description><string_hints /><subcolumns><s
Set objWMIService = GetObject(&amp;quot;winmgmts:&amp;quot; _
    &amp;amp; &amp;quot;{impersonationLevel=impersonate}!\\&amp;quot; &amp;amp; strComputer &amp;amp

Set collip = objWMIService.ExecQuery(&amp;quot;select * from win32_networkadapterconfiguration where
dim ipaddrs()
ipcount = 0
for each ipItem in collip
    for each ipaddr in ipItem.IPAddress
        ipcount = ipcount + 1
    next
next
redim ipaddrs(ipcount)
..trimmed for brevity..
```

Export basetype xml minimal false

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

Example Python Code

```python
# Path to lib directory which contains pytan package
PYTAN_LIB_PATH = '../lib'

# connection info for Tanium Server
USERNAME = "Tanium User"
PASSWORD = "T@n!um"
HOST = "172.16.31.128"
PORT = "444"

# Logging conrols
LOGLEVEL = 2
DEBUGFORMAT = False

import sys, tempfile
sys.path.append(PYTAN_LIB_PATH)

import pytan
handler = pytan.Handler(
    username=USERNAME,
    password=PASSWORD,
    host=HOST,
    port=PORT,
    loglevel=LOGLEVEL,
    debugformat=DEBUGFORMAT,
)

print handler

# setup the export_obj kwargs for later
export_kwargs = {}
export_kwargs["export_format"] = u'xml'
export_kwargs["minimal"] = False

# get the objects that will provide the basetype that we want to use
get_kwargs = {
    'name': [
        "Computer Name", "IP Route Details", "IP Address",
        'Folder Name Search with RegEx Match',
    ],
    'objtype': 'sensor',
}
response = handler.get(**get_kwargs)

# export the object to a string
# (we could just as easily export to a file using export_to_report_file)
export_kwargs['obj'] = response
export_str = handler.export_obj(**export_kwargs)


print ""
print "print the export_str returned from export_obj():"

```

```
out = export_str
if len(out.splitlines()) > 15:
    out = out.splitlines()[0:15]
    out.append('..trimmed for brevity..')
    out = '\n'.join(out)

print out
```

Output from Python Code

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

print the export_str returned from export_obj():
<sensors><cache_info /><sensor><category>Reserved</category><preview_sensor_flag /><hash>3409330187<
Example: workstation-1.company.com</description><string_hints /><subcolumns /><metadata /><parameter
Example:  172.16.0.0|192.168.1.1|255.255.0.0|UG|100|eth0</description><string_hints /><subcolumns><s
Set objWMIService = GetObject(&amp;quot;winmgmts:&amp;quot; _
    &amp;amp; &amp;quot;{impersonationLevel=impersonate}!\\&amp;quot; &amp;amp; strComputer &amp;amp

Set collip = objWMIService.ExecQuery(&amp;quot;select * from win32_networkadapterconfiguration where
dim ipaddrs()
ipcount = 0
for each ipItem in collip
    for each ipaddr in ipItem.IPAddress
        ipcount = ipcount + 1
    next
next
redim ipaddrs(ipcount)
..trimmed for brevity..
```

Export basetype xml minimal true

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

Example Python Code

```
# Path to lib directory which contains pytan package
PYTAN_LIB_PATH = '../lib'

# connection info for Tanium Server
USERNAME = "Tanium User"
PASSWORD = "T@n!um"
HOST = "172.16.31.128"
PORT = "444"

# Logging conrols
LOGLEVEL = 2
DEBUGFORMAT = False

import sys, tempfile
sys.path.append(PYTAN_LIB_PATH)

import pytan
handler = pytan.Handler(
    username=USERNAME,
    password=PASSWORD,
    host=HOST,
    port=PORT,
    loglevel=LOGLEVEL,
    debugformat=DEBUGFORMAT,
)

print handler

# setup the export_obj kwargs for later
export_kwargs = {}
export_kwargs["export_format"] = u'xml'
export_kwargs["minimal"] = True

# get the objects that will provide the basetype that we want to use
get_kwargs = {
    'name': [
        "Computer Name", "IP Route Details", "IP Address",
        'Folder Name Search with RegEx Match',
    ],
    'objtype': 'sensor',
}
response = handler.get(**get_kwargs)

# export the object to a string
# (we could just as easily export to a file using export_to_report_file)
export_kwargs['obj'] = response
export_str = handler.export_obj(**export_kwargs)


print ""
print "print the export_str returned from export_obj():"

out = export_str
if len(out.splitlines()) > 15:
    out = out.splitlines()[0:15]
    out.append('..trimmed for brevity..')
    out = '\n'.join(out)

print out
```

Output from Python Code

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

print the export_str returned from export_obj():
<sensors><sensor><category>Reserved</category><hash>3409330187</hash><name>Computer Name</name><hid
Example: workstation-1.company.com</description><queries><query><platform>Windows</platform><script_
Example:  172.16.0.0|192.168.1.1|255.255.0.0|UG|100|eth0</description><subcolumns><subcolumn><index>
Set objWMIService = GetObject(&amp;quot;winmgmts:&amp;quot; _
    &amp;amp; &amp;quot;{impersonationLevel=impersonate}!\\&amp;quot; &amp;amp; strComputer &amp;amp

Set collip = objWMIService.ExecQuery(&amp;quot;select * from win32_networkadapterconfiguration where
dim ipaddrs()
ipcount = 0
for each ipItem in collip
```

```
    for each ipaddr in ipItem.IPAddress
        ipcount = ipcount + 1
    next
next
redim ipaddrs(ipcount)
..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

```
# Path to lib directory which contains pytan package
PYTAN_LIB_PATH = '../lib'

# connection info for Tanium Server
USERNAME = "Tanium User"
PASSWORD = "T@n!um"
HOST = "172.16.31.128"
PORT = "444"

# Logging conrols
LOGLEVEL = 2
DEBUGFORMAT = False

import sys, tempfile
sys.path.append(PYTAN_LIB_PATH)

import pytan
handler = pytan.Handler(
    username=USERNAME,
    password=PASSWORD,
    host=HOST,
    port=PORT,
    loglevel=LOGLEVEL,
    debugformat=DEBUGFORMAT,
)

print handler

# setup the export_obj kwargs for later
export_kwargs = {}
export_kwargs["export_format"] = u'csv'
export_kwargs["explode_json_string_values"] = False

# get the objects that will provide the basetype that we want to use
get_kwargs = {
    'name': [
        "Computer Name", "IP Route Details", "IP Address",
        'Folder Name Search with RegEx Match',
    ],
    'objtype': 'sensor',
}
response = handler.get(**get_kwargs)
```

```

# export the object to a string
# (we could just as easily export to a file using export_to_report_file)
export_kwargs['obj'] = response
export_str = handler.export_obj(**export_kwargs)


print ""
print "print the export_str returned from export_obj():"

out = export_str
if len(out.splitlines()) > 15:
    out = out.splitlines()[0:15]
    out.append('..trimmed for brevity..')
    out = '\n'.join(out)

print out
```

Output from Python Code

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

print the export_str returned from export_obj():
category,creation_time,delimiter,description,exclude_from_parse_flag,hash,hidden_flag,id,ignore_case
Reserved,,,"The assigned name of the client machine.
Example: workstation-1.company.com",0,3409330187,0,3,1,,86400,,,,,Computer Name,,Windows,select CSNa
Network,2014-12-08T19:20:42,|,"Returns IPv4 network routes, filtered to exclude noise. With Flags, M
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
Set objWMIService = GetObject(&quot;winmgmts:&quot; _
    &amp; &quot;{impersonationLevel=impersonate}!\\&quot; &amp; strComputer &amp; &quot;\root\cimv2&

Set collip = objWMIService.ExecQuery(&quot;select * from win32_networkadapterconfiguration where IPE
dim ipaddrs()
ipcount = 0
for each ipItem in collip
    for each ipaddr in ipItem.IPAddress
        ipcount = ipcount + 1
    next
..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

```
# Path to lib directory which contains pytan package
PYTAN_LIB_PATH = '../lib'

# connection info for Tanium Server
USERNAME = "Tanium User"
PASSWORD = "T@n!um"
HOST = "172.16.31.128"
PORT = "444"
```

```python

# Logging conrols
LOGLEVEL = 2
DEBUGFORMAT = False

import sys, tempfile
sys.path.append(PYTAN_LIB_PATH)

import pytan
handler = pytan.Handler(
    username=USERNAME,
    password=PASSWORD,
    host=HOST,
    port=PORT,
    loglevel=LOGLEVEL,
    debugformat=DEBUGFORMAT,
)

print handler

# setup the export_obj kwargs for later
export_kwargs = {}
export_kwargs["export_format"] = u'csv'
export_kwargs["explode_json_string_values"] = True

# get the objects that will provide the basetype that we want to use
get_kwargs = {
    'name': [
        "Computer Name", "IP Route Details", "IP Address",
        'Folder Name Search with RegEx Match',
    ],
    'objtype': 'sensor',
}
response = handler.get(**get_kwargs)

# export the object to a string
# (we could just as easily export to a file using export_to_report_file)
export_kwargs['obj'] = response
export_str = handler.export_obj(**export_kwargs)


print ""
print "print the export_str returned from export_obj():"

out = export_str
if len(out.splitlines()) > 15:
    out = out.splitlines()[0:15]
    out.append('..trimmed for brevity..')
    out = '\n'.join(out)

print out
```

Output from Python Code

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

print the export_str returned from export_obj():
```

```
category,creation_time,delimiter,description,exclude_from_parse_flag,hash,hidden_flag,id,ignore_case
Reserved,,,"The assigned name of the client machine.
Example: workstation-1.company.com",0,3409330187,0,3,1,,86400,,,,,Computer Name,,,,,,,,,,,,,,,,,,,,
Network,2014-12-08T19:20:42,|,"Returns IPv4 network routes, filtered to exclude noise. With Flags, M
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
Set objWMIService = GetObject(&quot;winmgmts:&quot; _
    &amp; &quot;{impersonationLevel=impersonate}!\\&quot; &amp; strComputer &amp; &quot;\root\cimv2&

Set collip = objWMIService.ExecQuery(&quot;select * from win32_networkadapterconfiguration where IPE
dim ipaddrs()
ipcount = 0
for each ipItem in collip
    for each ipaddr in ipItem.IPAddress
        ipcount = ipcount + 1
    next
..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

```python
# Path to lib directory which contains pytan package
PYTAN_LIB_PATH = '../lib'

# connection info for Tanium Server
USERNAME = "Tanium User"
PASSWORD = "T@n!um"
HOST = "172.16.31.128"
PORT = "444"

# Logging conrols
LOGLEVEL = 2
DEBUGFORMAT = False

import sys, tempfile
sys.path.append(PYTAN_LIB_PATH)

import pytan
handler = pytan.Handler(
    username=USERNAME,
    password=PASSWORD,
    host=HOST,
    port=PORT,
    loglevel=LOGLEVEL,
    debugformat=DEBUGFORMAT,
)

print handler

# setup the export_obj kwargs for later
export_kwargs = {}
export_kwargs["export_format"] = u'csv'
export_kwargs["header_sort"] = []
```

```

# get the objects that will provide the basetype that we want to use
get_kwargs = {
    'name': [
        "Computer Name", "IP Route Details", "IP Address",
        'Folder Name Search with RegEx Match',
    ],
    'objtype': 'sensor',
}
response = handler.get(**get_kwargs)

# export the object to a string
# (we could just as easily export to a file using export_to_report_file)
export_kwargs['obj'] = response
export_str = handler.export_obj(**export_kwargs)


print ""
print "print the export_str returned from export_obj():"

out = export_str
if len(out.splitlines()) > 15:
    out = out.splitlines()[0:15]
    out.append('..trimmed for brevity..')
    out = '\n'.join(out)

print out
```

Output from Python Code

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

print the export_str returned from export_obj():
category,creation_time,delimiter,description,exclude_from_parse_flag,hash,hidden_flag,id,ignore_case
Reserved,,,"The assigned name of the client machine.
Example: workstation-1.company.com",0,3409330187,0,3,1,,86400,,,,,Computer Name,,Windows,select CSNa
Network,2014-12-08T19:20:42,|,"Returns IPv4 network routes, filtered to exclude noise. With Flags, M
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
Set objWMIService = GetObject(&quot;winmgmts:&quot; _
    &amp; &quot;{impersonationLevel=impersonate}!\\&quot; &amp; strComputer &amp; &quot;\root\cimv2&

Set collip = objWMIService.ExecQuery(&quot;select * from win32_networkadapterconfiguration where IPE
dim ipaddrs()
ipcount = 0
for each ipItem in collip
    for each ipaddr in ipItem.IPAddress
        ipcount = ipcount + 1
    next
..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

```python
# Path to lib directory which contains pytan package
PYTAN_LIB_PATH = '../lib'

# connection info for Tanium Server
USERNAME = "Tanium User"
PASSWORD = "T@n!um"
HOST = "172.16.31.128"
PORT = "444"

# Logging conrols
LOGLEVEL = 2
DEBUGFORMAT = False

import sys, tempfile
sys.path.append(PYTAN_LIB_PATH)

import pytan
handler = pytan.Handler(
    username=USERNAME,
    password=PASSWORD,
    host=HOST,
    port=PORT,
    loglevel=LOGLEVEL,
    debugformat=DEBUGFORMAT,
)

print handler

# setup the export_obj kwargs for later
export_kwargs = {}
export_kwargs["export_format"] = u'csv'
export_kwargs["header_sort"] = True

# get the objects that will provide the basetype that we want to use
get_kwargs = {
    'name': [
        "Computer Name", "IP Route Details", "IP Address",
        'Folder Name Search with RegEx Match',
    ],
    'objtype': 'sensor',
}
response = handler.get(**get_kwargs)

# export the object to a string
# (we could just as easily export to a file using export_to_report_file)
export_kwargs['obj'] = response
export_str = handler.export_obj(**export_kwargs)


print ""
print "print the export_str returned from export_obj():"

out = export_str
if len(out.splitlines()) > 15:
    out = out.splitlines()[0:15]
    out.append('..trimmed for brevity..')
    out = '\n'.join(out)
```

```

print out
```

Output from Python Code

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

print the export_str returned from export_obj():
category,creation_time,delimiter,description,exclude_from_parse_flag,hash,hidden_flag,id,ignore_case
Reserved,,,"The assigned name of the client machine.
Example: workstation-1.company.com",0,3409330187,0,3,1,,86400,,,,,Computer Name,,Windows,select CSNa
Network,2014-12-08T19:20:42,|,"Returns IPv4 network routes, filtered to exclude noise. With Flags, M
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
Set objWMIService = GetObject(&quot;winmgmts:&quot; _
    &amp; &quot;{impersonationLevel=impersonate}!\\&quot; &amp; strComputer &amp; &quot;\root\cimv2&

Set collip = objWMIService.ExecQuery(&quot;select * from win32_networkadapterconfiguration where IPE
dim ipaddrs()
ipcount = 0
for each ipItem in collip
    for each ipaddr in ipItem.IPAddress
        ipcount = ipcount + 1
    next
..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

```python
# Path to lib directory which contains pytan package
PYTAN_LIB_PATH = '../lib'

# connection info for Tanium Server
USERNAME = "Tanium User"
PASSWORD = "T@n!um"
HOST = "172.16.31.128"
PORT = "444"

# Logging conrols
LOGLEVEL = 2
DEBUGFORMAT = False

import sys, tempfile
sys.path.append(PYTAN_LIB_PATH)

import pytan
handler = pytan.Handler(
    username=USERNAME,
    password=PASSWORD,
    host=HOST,
    port=PORT,
    loglevel=LOGLEVEL,
    debugformat=DEBUGFORMAT,
)

print handler

# setup the export_obj kwargs for later
export_kwargs = {}
export_kwargs["export_format"] = u'csv'
export_kwargs["header_sort"] = [u'name', u'description']

# get the objects that will provide the basetype that we want to use
get_kwargs = {
    'name': [
        "Computer Name", "IP Route Details", "IP Address",
        'Folder Name Search with RegEx Match',
    ],
    'objtype': 'sensor',
}
response = handler.get(**get_kwargs)

# export the object to a string
# (we could just as easily export to a file using export_to_report_file)
export_kwargs['obj'] = response
export_str = handler.export_obj(**export_kwargs)


print ""
print "print the export_str returned from export_obj():"

out = export_str
if len(out.splitlines()) > 15:
    out = out.splitlines()[0:15]
    out.append('..trimmed for brevity..')
    out = '\n'.join(out)

print out
```

Output from Python Code

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

print the export_str returned from export_obj():
name,description,category,creation_time,delimiter,exclude_from_parse_flag,hash,hidden_flag,id,ignore
Computer Name,"The assigned name of the client machine.
Example: workstation-1.company.com",Reserved,,,0,3409330187,0,3,1,,86400,,,,,,Windows,select CSName
IP Route Details,"Returns IPv4 network routes, filtered to exclude noise. With Flags, Metric, Interf
Example:  172.16.0.0|192.168.1.1|255.255.0.0|UG|100|eth0",Network,2014-12-08T19:20:42,|,1,435227963,
Set objWMIService = GetObject(&quot;winmgmts:&quot; _
    &amp; &quot;{impersonationLevel=impersonate}!\\&quot; &amp; strComputer &amp; &quot;\root\cimv2&

Set collip = objWMIService.ExecQuery(&quot;select * from win32_networkadapterconfiguration where IPE
dim ipaddrs()
ipcount = 0
for each ipItem in collip
    for each ipaddr in ipItem.IPAddress
        ipcount = ipcount + 1
    next
```

```
..trimmed for brevity..
```

Export basetype json default options

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

Example Python Code

```python
# Path to lib directory which contains pytan package
PYTAN_LIB_PATH = '../lib'

# connection info for Tanium Server
USERNAME = "Tanium User"
PASSWORD = "T@n!um"
HOST = "172.16.31.128"
PORT = "444"

# Logging conrols
LOGLEVEL = 2
DEBUGFORMAT = False

import sys, tempfile
sys.path.append(PYTAN_LIB_PATH)

import pytan
handler = pytan.Handler(
    username=USERNAME,
    password=PASSWORD,
    host=HOST,
    port=PORT,
    loglevel=LOGLEVEL,
    debugformat=DEBUGFORMAT,
)

print handler

# setup the export_obj kwargs for later
export_kwargs = {}
export_kwargs["export_format"] = u'json'

# get the objects that will provide the basetype that we want to use
get_kwargs = {
    'name': [
        "Computer Name", "IP Route Details", "IP Address",
        'Folder Name Search with RegEx Match',
    ],
    'objtype': 'sensor',
}
response = handler.get(**get_kwargs)

# export the object to a string
# (we could just as easily export to a file using export_to_report_file)
export_kwargs['obj'] = response
export_str = handler.export_obj(**export_kwargs)

```

```

print ""
print "print the export_str returned from export_obj():"

out = export_str
if len(out.splitlines()) > 15:
    out = out.splitlines()[0:15]
    out.append('..trimmed for brevity..')
    out = '\n'.join(out)

print out
```

Output from Python Code

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

print the export_str returned from export_obj():
{
  "_type": "sensors",
  "sensor": [
    {
      "_type": "sensor",
      "category": "Reserved",
      "description": "The assigned name of the client machine.\nExample: workstation-1.company.com",
      "exclude_from_parse_flag": 0,
      "hash": 3409330187,
      "hidden_flag": 0,
      "id": 3,
      "ignore_case_flag": 1,
      "max_age_seconds": 86400,
      "name": "Computer Name",
      "queries": {
..trimmed for brevity..
```

Export basetype json type true

Export a BaseType from getting objects as JSON with true for include_type

Example Python Code

```
# Path to lib directory which contains pytan package
PYTAN_LIB_PATH = '../lib'

# connection info for Tanium Server
USERNAME = "Tanium User"
PASSWORD = "T@n!um"
HOST = "172.16.31.128"
PORT = "444"

# Logging conrols
LOGLEVEL = 2
DEBUGFORMAT = False

import sys, tempfile
```

```python
sys.path.append(PYTAN_LIB_PATH)

import pytan
handler = pytan.Handler(
    username=USERNAME,
    password=PASSWORD,
    host=HOST,
    port=PORT,
    loglevel=LOGLEVEL,
    debugformat=DEBUGFORMAT,
)

print handler

# setup the export_obj kwargs for later
export_kwargs = {}
export_kwargs["export_format"] = u'json'
export_kwargs["include_type"] = True

# get the objects that will provide the basetype that we want to use
get_kwargs = {
    'name': [
        "Computer Name", "IP Route Details", "IP Address",
        'Folder Name Search with RegEx Match',
    ],
    'objtype': 'sensor',
}
response = handler.get(**get_kwargs)

# export the object to a string
# (we could just as easily export to a file using export_to_report_file)
export_kwargs['obj'] = response
export_str = handler.export_obj(**export_kwargs)


print ""
print "print the export_str returned from export_obj():"

out = export_str
if len(out.splitlines()) > 15:
    out = out.splitlines()[0:15]
    out.append('..trimmed for brevity..')
    out = '\n'.join(out)

print out
```

Output from Python Code

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

print the export_str returned from export_obj():
{
  "_type": "sensors",
  "sensor": [
    {
      "_type": "sensor",
      "category": "Reserved",
      "description": "The assigned name of the client machine.\nExample: workstation-1.company.com",
      "exclude_from_parse_flag": 0,
      "hash": 3409330187,
      "hidden_flag": 0,
      "id": 3,
      "ignore_case_flag": 1,
      "max_age_seconds": 86400,
      "name": "Computer Name",
      "queries": {
..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

```python
# Path to lib directory which contains pytan package
PYTAN_LIB_PATH = '../lib'

# connection info for Tanium Server
USERNAME = "Tanium User"
PASSWORD = "T@n!um"
HOST = "172.16.31.128"
PORT = "444"

# Logging conrols
LOGLEVEL = 2
DEBUGFORMAT = False

import sys, tempfile
sys.path.append(PYTAN_LIB_PATH)

import pytan
handler = pytan.Handler(
    username=USERNAME,
    password=PASSWORD,
    host=HOST,
    port=PORT,
    loglevel=LOGLEVEL,
    debugformat=DEBUGFORMAT,
)

print handler

# setup the export_obj kwargs for later
export_kwargs = {}
export_kwargs["export_format"] = u'csv'
export_kwargs["explode_json_string_values"] = u'bad'

# get the objects that will provide the basetype that we want to use
get_kwargs = {
    'name': [
        "Computer Name", "IP Route Details", "IP Address",
        'Folder Name Search with RegEx Match',
    ],
    'objtype': 'sensor',
}
response = handler.get(**get_kwargs)
export_kwargs['obj'] = response

# export the object to a string
# this should throw an exception: pytan.utils.HandlerError
import traceback

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

Output from Python Code

```
Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
Traceback (most recent call last):
  File "<string>", line 50, in <module>
  File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 1402, in export_obj
    utils.check_dictkey(**check_args)
  File "/Users/jolsen/gh/pytan/lib/pytan/utils.py", line 2509, in check_dictkey
    raise HandlerError(err(key, valid_types, k_type))
HandlerError: 'explode_json_string_values' must be one of [<type 'bool'>], you supplied <type 'unico
```

Invalid export basetype csv bad sort sub type

Export a BaseType from getting objects using a bad header_sort

Example Python Code

```
# Path to lib directory which contains pytan package
PYTAN_LIB_PATH = '../lib'

# connection info for Tanium Server
USERNAME = "Tanium User"
PASSWORD = "T@n!um"
HOST = "172.16.31.128"
PORT = "444"

# Logging conrols
LOGLEVEL = 2
DEBUGFORMAT = False

import sys, tempfile
sys.path.append(PYTAN_LIB_PATH)

import pytan
handler = pytan.Handler(
    username=USERNAME,
    password=PASSWORD,
    host=HOST,
    port=PORT,
    loglevel=LOGLEVEL,
    debugformat=DEBUGFORMAT,
)

print handler

# setup the export_obj kwargs for later
export_kwargs = {}
export_kwargs["export_format"] = u'csv'
export_kwargs["header_sort"] = [[]]

# get the objects that will provide the basetype that we want to use
get_kwargs = {
    'name': [
        "Computer Name", "IP Route Details", "IP Address",
        'Folder Name Search with RegEx Match',
    ],
    'objtype': 'sensor',
}
response = handler.get(**get_kwargs)
export_kwargs['obj'] = response

# export the object to a string
# this should throw an exception: pytan.utils.HandlerError
import traceback

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

Output from Python Code

```
Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
Traceback (most recent call last):
  File "<string>", line 50, in <module>
  File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 1402, in export_obj
    utils.check_dictkey(**check_args)
  File "/Users/jolsen/gh/pytan/lib/pytan/utils.py", line 2516, in check_dictkey
    raise HandlerError(err(key, valid_list_types, list_types))
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

```
# Path to lib directory which contains pytan package
PYTAN_LIB_PATH = '../lib'

# connection info for Tanium Server
```

```
USERNAME = "Tanium User"
PASSWORD = "T@n!um"
HOST = "172.16.31.128"
PORT = "444"

# Logging conrols
LOGLEVEL = 2
DEBUGFORMAT = False

import sys, tempfile
sys.path.append(PYTAN_LIB_PATH)

import pytan
handler = pytan.Handler(
    username=USERNAME,
    password=PASSWORD,
    host=HOST,
    port=PORT,
    loglevel=LOGLEVEL,
    debugformat=DEBUGFORMAT,
)

print handler

# setup the export_obj kwargs for later
export_kwargs = {}
export_kwargs["export_format"] = u'csv'
export_kwargs["header_sort"] = u'bad'

# get the objects that will provide the basetype that we want to use
get_kwargs = {
    'name': [
        "Computer Name", "IP Route Details", "IP Address",
        'Folder Name Search with RegEx Match',
    ],
    'objtype': 'sensor',
}
response = handler.get(**get_kwargs)
export_kwargs['obj'] = response

# export the object to a string
# this should throw an exception: pytan.utils.HandlerError
import traceback

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

Output from Python Code

```
Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
Traceback (most recent call last):
  File "<string>", line 50, in <module>
  File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 1402, in export_obj
    utils.check_dictkey(**check_args)
  File "/Users/jolsen/gh/pytan/lib/pytan/utils.py", line 2509, in check_dictkey
```

```
    raise HandlerError(err(key, valid_types, k_type))
HandlerError: 'header_sort' must be one of [<type 'bool'>, <type 'list'>, <type 'tuple'>], you suppl
```

Invalid export basetype xml bad minimal type

Export a BaseType from getting objects using a bad minimal

Example Python Code

```
# Path to lib directory which contains pytan package
PYTAN_LIB_PATH = '../lib'

# connection info for Tanium Server
USERNAME = "Tanium User"
PASSWORD = "T@n!um"
HOST = "172.16.31.128"
PORT = "444"

# Logging conrols
LOGLEVEL = 2
DEBUGFORMAT = False

import sys, tempfile
sys.path.append(PYTAN_LIB_PATH)

import pytan
handler = pytan.Handler(
    username=USERNAME,
    password=PASSWORD,
    host=HOST,
    port=PORT,
    loglevel=LOGLEVEL,
    debugformat=DEBUGFORMAT,
)

print handler

# setup the export_obj kwargs for later
export_kwargs = {}
export_kwargs["export_format"] = u'xml'
export_kwargs["minimal"] = u'bad'

# get the objects that will provide the basetype that we want to use
get_kwargs = {
    'name': [
        "Computer Name", "IP Route Details", "IP Address",
        'Folder Name Search with RegEx Match',
    ],
    'objtype': 'sensor',
}
response = handler.get(**get_kwargs)
export_kwargs['obj'] = response

# export the object to a string
# this should throw an exception: pytan.utils.HandlerError
```

```python
import traceback

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

Output from Python Code

```
Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
Traceback (most recent call last):
  File "<string>", line 50, in <module>
  File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 1402, in export_obj
    utils.check_dictkey(**check_args)
  File "/Users/jolsen/gh/pytan/lib/pytan/utils.py", line 2509, in check_dictkey
    raise HandlerError(err(key, valid_types, k_type))
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

```python
# Path to lib directory which contains pytan package
PYTAN_LIB_PATH = '../lib'

# connection info for Tanium Server
USERNAME = "Tanium User"
PASSWORD = "T@n!um"
HOST = "172.16.31.128"
PORT = "444"

# Logging conrols
LOGLEVEL = 2
DEBUGFORMAT = False

import sys, tempfile
sys.path.append(PYTAN_LIB_PATH)

import pytan
handler = pytan.Handler(
    username=USERNAME,
    password=PASSWORD,
    host=HOST,
    port=PORT,
    loglevel=LOGLEVEL,
    debugformat=DEBUGFORMAT,
)

print handler

# setup the export_obj kwargs for later
export_kwargs = {}
```

```
export_kwargs["export_format"] = u'json'
export_kwargs["include_type"] = u'bad'

# get the objects that will provide the basetype that we want to use
get_kwargs = {
    'name': [
        "Computer Name", "IP Route Details", "IP Address",
        'Folder Name Search with RegEx Match',
    ],
    'objtype': 'sensor',
}
response = handler.get(**get_kwargs)
export_kwargs['obj'] = response

# export the object to a string
# this should throw an exception: pytan.utils.HandlerError
import traceback

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

Output from Python Code

```
Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
Traceback (most recent call last):
  File "<string>", line 50, in <module>
  File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 1402, in export_obj
    utils.check_dictkey(**check_args)
  File "/Users/jolsen/gh/pytan/lib/pytan/utils.py", line 2509, in check_dictkey
    raise HandlerError(err(key, valid_types, k_type))
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

```
# Path to lib directory which contains pytan package
PYTAN_LIB_PATH = '../lib'

# connection info for Tanium Server
USERNAME = "Tanium User"
PASSWORD = "T@n!um"
HOST = "172.16.31.128"
PORT = "444"

# Logging conrols
LOGLEVEL = 2
DEBUGFORMAT = False

import sys, tempfile
```

```
sys.path.append(PYTAN_LIB_PATH)

import pytan
handler = pytan.Handler(
    username=USERNAME,
    password=PASSWORD,
    host=HOST,
    port=PORT,
    loglevel=LOGLEVEL,
    debugformat=DEBUGFORMAT,
)

print handler

# setup the export_obj kwargs for later
export_kwargs = {}
export_kwargs["export_format"] = u'json'
export_kwargs["explode_json_string_values"] = u'bad'

# get the objects that will provide the basetype that we want to use
get_kwargs = {
    'name': [
        "Computer Name", "IP Route Details", "IP Address",
        'Folder Name Search with RegEx Match',
    ],
    'objtype': 'sensor',
}
response = handler.get(**get_kwargs)
export_kwargs['obj'] = response

# export the object to a string
# this should throw an exception: pytan.utils.HandlerError
import traceback

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

Output from Python Code

```
Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
Traceback (most recent call last):
  File "<string>", line 50, in <module>
  File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 1402, in export_obj
    utils.check_dictkey(**check_args)
  File "/Users/jolsen/gh/pytan/lib/pytan/utils.py", line 2509, in check_dictkey
    raise HandlerError(err(key, valid_types, k_type))
HandlerError: 'explode_json_string_values' must be one of [<type 'bool'>], you supplied <type 'unico
```

Invalid export basetype bad format

Export a BaseType from getting objects using a bad export_format

Example Python Code

```python
# Path to lib directory which contains pytan package
PYTAN_LIB_PATH = '../lib'

# connection info for Tanium Server
USERNAME = "Tanium User"
PASSWORD = "T@n!um"
HOST = "172.16.31.128"
PORT = "444"

# Logging conrols
LOGLEVEL = 2
DEBUGFORMAT = False

import sys, tempfile
sys.path.append(PYTAN_LIB_PATH)

import pytan
handler = pytan.Handler(
    username=USERNAME,
    password=PASSWORD,
    host=HOST,
    port=PORT,
    loglevel=LOGLEVEL,
    debugformat=DEBUGFORMAT,
)

print handler

# setup the export_obj kwargs for later
export_kwargs = {}
export_kwargs["export_format"] = u'bad'

# get the objects that will provide the basetype that we want to use
get_kwargs = {
    'name': [
        "Computer Name", "IP Route Details", "IP Address",
        'Folder Name Search with RegEx Match',
    ],
    'objtype': 'sensor',
}
response = handler.get(**get_kwargs)
export_kwargs['obj'] = response

# export the object to a string
# this should throw an exception: pytan.utils.HandlerError
import traceback

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

Output from Python Code

```
Handler for Session to 172.16.31.128:444, Authenticated: True, Version: 6.2.314.3258
Traceback (most recent call last):
  File "<string>", line 49, in <module>
  File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 1396, in export_obj
    raise HandlerError(err)
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 strs, optional

- each list must be a 2 item list:
- list item 1 property name
- list item 2 property value

Returns **url_obj** : `taniumpy.object_types.white_listed_url.WhiteListedUrl`

TaniumPy object added to Tanium SOAP Server

Handler Methods: Deleting Objects

Delete an Object

`Handler.`**`delete`**(*objtype*, ***kwargs*)

Delete an object type

Parameters **objtype** : string

type of object to delete

id/name/hash : int or string, list of int or string

search attributes of object to delete, must supply at least one valid search attr

Returns **ret** : dict

dict containing deploy action object and results from deploy action

See also:

`pytan.constants.GET_OBJ_MAP` maps objtype to supported 'search' keys

Handler Methods: Getting Objects

Get Single or Multiple Objects of a type

`Handler.`**`get`**(*objtype*, ***kwargs*)

Get an object type

Parameters **objtype** : string

type of object to get

id/name/hash : int or string, list of int or string

search attributes of object to get, must supply at least one valid search attr

See also:

`pytan.constants.GET_OBJ_MAP` maps objtype to supported 'search' keys

Get All Objects of a type

`Handler.`**`get_all`**(*objtype*, ***kwargs*)

Get all objects of a type

Parameters **objtype** : string

type of object to get

See also:

`pytan.constants.GET_OBJ_MAP` maps objtype to supported 'search' keys

Handler Methods: Getting Result Data / Result Info

`Handler.`**`get_result_data`**(*obj*, *aggregate=False*, ***kwargs*)

Get the result data for a python API object

This method issues a GetResultData command to the SOAP api for *obj*. GetResultData returns the columns and rows that are currently available for *obj*.

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

object to get result data for

aggregate : bool, optional

- False: get all the data
- True: get just the aggregate data (row counts of matches)

Returns **rd** : `taniumpy.object_types.result_set.ResultSet`

The return of GetResultData for *obj*

`Handler.`**`get_result_info`**(*obj*, ***kwargs*)

Get the result info for a python API object

This method issues a GetResultInfo command to the SOAP api for *obj*. GetResultInfo returns information about how many servers have passed the *obj*, total number of servers, and so on.

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

object to get result data for

Returns ri : `taniumpy.object_types.result_info.ResultInfo`

The return of GetResultData for *obj*

Handler Methods: Private Methods

`Handler._find`(*api_object*, ***kwargs*)
Wrapper for interfacing with `taniumpy.session.Session.find()`

`Handler._get_multi`(*obj_map*, ***kwargs*)
Find multiple item wrapper using `_find()`

`Handler._get_single`(*obj_map*, ***kwargs*)
Find single item wrapper using `_find()`

`Handler._single_find`(*obj_map*, *k*, *v*, ***kwargs*)
Wrapper for single item searches interfacing with `taniumpy.session.Session.find()`

`Handler._get_sensor_defs`(*defs*)
Uses `get()` to update a definition with a sensor object

`Handler._get_package_def`(*d*)
Uses `get()` to update a definition with a package object

`Handler._export_class_BaseType`(*obj*, *export_format*, ***kwargs*)
Handles exporting `taniumpy.object_types.base.BaseType`

`Handler._export_class_ResultSet`(*obj*, *export_format*, ***kwargs*)
Handles exporting `taniumpy.object_types.result_set.ResultSet`

`Handler._export_format_csv`(*obj*, ***kwargs*)
Handles exporting format: CSV

`Handler._export_format_json`(*obj*, ***kwargs*)
Handles exporting format: JSON

`Handler._export_format_xml`(*obj*, ***kwargs*)
Handles exporting format: XML

1.7.3 pytan.constants module

PyTan Constants

This contains a number of constants that drive PyTan.

`pytan.constants.ACTION_RESULT_STATUS` **= {'Verified.': ['no_verify_done', 'verify_done', 'verify_success'], 'Succeeded**
Maps a deploy action result status to it's respective end states.

`pytan.constants.ASK_KWARGS` **= ['timeout', 'polling_interval', 'pct_complete_threshold']**
A list of arguments that will be passed on to the question asker/poller `taniumpy.question_asker.QuestionAsker`

`pytan.constants.DEBUG_FORMAT` **= '[%(lineno)-5d - %(filename)20s:%(funcName)s()] %(asctime)s\n%(levelname)-8s %**
Logging format for debugformat=True

`pytan.constants.EXPORT_MAPS` **= {'ResultSet': {'json': [], 'csv': [{'valid_list_types': ['str', 'unicode'], 'key': 'header_sor**

Maps a given TaniumPy object to the list of supported export formats for each object type, and the valid optional argume

- 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**
Maps a question type from a human friendly string into the handler method that supports each type

`pytan.constants.`**`REQ_KWARGS`** **= ['hide_errors_flag', 'include_answer_times_flag', 'row_counts_only_flag', 'aggregate_ov**
A list of arguments that will be pulled from any respective kwargs for most calls to `taniumpy.session.Session`

`pytan.constants.`**`SELECTORS`** **= ['id', 'name', 'hash']**
The search selectors that can be extracted from a string. Ex: `name`:*Sensor1*, or `id`:*1*, or `hash`:*1111111*

`pytan.constants.`**`SENSOR_TYPE_MAP`** **= {0: 'Hash', 1: 'String', 2: 'Version', 3: 'NumericDecimal', 4: 'BESDate', 5: 'IPAd**
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 strs from if list contains any item from *i*

i : list of str

list of strs to check if for existence in any lists in *d*

Returns list : list of str

list of strings from *d* that have *i* in their values

`pytan.utils.`**`get_dict_list_len`**(*d, keys=[], negate=False*)
Gets the sum of each list in dict *d*

Parameters d : dict of str

dict to sums of

keys : list of str

list of keys to get sums of, if empty gets a sum of all keys

negate : bool

- only used if keys supplied
- False : get the sums of *d* that do match keys
- True : get the sums of *d* that do not match keys

Returns **list_len** : int

sum of lists in *d* that match keys

`pytan.utils.`**`get_now`**`()`

Get current time in human friendly format

Returns str :

str of current time return from `human_time()`

`pytan.utils.`**`human_time`**`(`*t*, *tformat='%Y_%m_%d-%H_%M_%S-%Z'*`)`

Get time in human friendly format

Parameters **t** : int, float, time

either a unix epoch or struct_time object to convert to string

tformat : str, optional

format of string to convert time to

Returns str :

t converted to str

`pytan.utils.`**`jsonify`**`(`*v*, *indent=2*, *sort_keys=True*`)`

Turns python object *v* into a pretty printed JSON string

Parameters **v** : object

python object to convert to JSON

indent : int, 2

number of spaces to indent JSON string when pretty printing

sort_keys : bool, True

sort keys of JSON string when pretty printing

Returns str :

JSON pretty printed string

`pytan.utils.`**`port_check`**`(`*address*, *port*, *timeout=5*`)`

Check if *address*:*port* can be reached within *timeout*

Parameters **address** : str

hostname/ip address to check *port* on

port : int

port to check on *address*

timeout : int, optional

timeout after N seconds of not being able to connect

Returns `socket` or False :

if connection succeeds, the socket object is returned, else False is returned

`pytan.utils.`**`seconds_from_now`**`(`*secs=0*, *tz='utc'*`)`

Get time in Tanium SOAP API format *secs* from now

Parameters **secs** : int

seconds from now to get time str

tz : str, optional

time zone to return string in, default is 'utc' - supplying anything else will supply local time

Returns str :

time *secs* from now in Tanium SOAP API format

`pytan.utils.`**`test_app_port`**`(`*host*, *port*`)`

Validates that *host*:*port* can be reached using `port_check()`

Parameters **host** : str

hostname/ip address to check *port* on

port : int

port to check on *host*

Raises **HandlerError** : `pytan.utils.HandlerError`

if *host*:*port* can not be reached

`pytan.utils.`**`version_check`**`(`*reqver*`)`

Allows scripts using `pytan` to validate the version of the script aginst the version of `pytan`

Parameters **reqver** : str

string containing version number to check against `Exception`

Raises **Exception** : `Exception`

if `pytan.__version__` is not greater or equal to *reqver*

`pytan.utils.`**`xml_pretty`**`(`*x*`)`

Uses `xmltodict` to pretty print an XML str *x*

Parameters **x** : str

XML string to pretty print

Returns str :

The pretty printed string of *x*

`pytan.utils.`**`xml_pretty_resultobj`**`(`*x*`)`

Uses `xmltodict` 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 `xmltodict` 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.ArgParse`

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.ArgParse`

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.ArgParse`

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.ArgParse`

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 strs

list of lists, each list having two strs - 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 of 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.`**`SoapError`**
Bases: `taniumpy.object_types.base.BaseType`

taniumpy.object_types.system_setting module

class `taniumpy.object_types.system_setting.`**`SystemSetting`**
Bases: `taniumpy.object_types.base.BaseType`

taniumpy.object_types.system_settings_list module

class `taniumpy.object_types.system_settings_list.`**`SystemSettingsList`**
Bases: `taniumpy.object_types.base.BaseType`

taniumpy.object_types.system_status_aggregate module

class `taniumpy.object_types.system_status_aggregate.`**`SystemStatusAggregate`**
Bases: `taniumpy.object_types.base.BaseType`

taniumpy.object_types.system_status_list module

class `taniumpy.object_types.system_status_list.`**`SystemStatusList`**
Bases: `taniumpy.object_types.base.BaseType`

taniumpy.object_types.upload_file module

class `taniumpy.object_types.upload_file.`**`UploadFile`**
Bases: `taniumpy.object_types.base.BaseType`

taniumpy.object_types.upload_file_list module

class `taniumpy.object_types.upload_file_list.`**`UploadFileList`**
Bases: `taniumpy.object_types.base.BaseType`

taniumpy.object_types.upload_file_status module

class `taniumpy.object_types.upload_file_status.`**`UploadFileStatus`**
Bases: `taniumpy.object_types.base.BaseType`

taniumpy.object_types.user module

class `taniumpy.object_types.user.`**`User`**
Bases: `taniumpy.object_types.base.BaseType`

taniumpy.object_types.user_list module

class `taniumpy.object_types.user_list.`**`UserList`**
Bases: `taniumpy.object_types.base.BaseType`

taniumpy.object_types.user_permissions module

class `taniumpy.object_types.user_permissions.`**`UserPermissions`**
Bases: `taniumpy.object_types.base.BaseType`

taniumpy.object_types.user_role module

class `taniumpy.object_types.user_role.`**`UserRole`**
Bases: `taniumpy.object_types.base.BaseType`

taniumpy.object_types.user_role_list module

class `taniumpy.object_types.user_role_list.`**`UserRoleList`**
Bases: `taniumpy.object_types.base.BaseType`

taniumpy.object_types.version_aggregate module

class `taniumpy.object_types.version_aggregate.`**`VersionAggregate`**
Bases: `taniumpy.object_types.base.BaseType`

taniumpy.object_types.version_aggregate_list module

class `taniumpy.object_types.version_aggregate_list.`**`VersionAggregateList`**
Bases: `taniumpy.object_types.base.BaseType`

taniumpy.object_types.white_listed_url module

class `taniumpy.object_types.white_listed_url.`**`WhiteListedUrl`**
Bases: `taniumpy.object_types.base.BaseType`

taniumpy.object_types.white_listed_url_list module

class `taniumpy.object_types.white_listed_url_list.`**`WhiteListedUrlList`**
Bases: `taniumpy.object_types.base.BaseType`

taniumpy.object_types.xml_error module

class `taniumpy.object_types.xml_error.`**`XmlError`**
Bases: `taniumpy.object_types.base.BaseType`

1.9 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-attribs 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
>>> xmltodict.parse('<a>hello</a>', expat=defusedexpat.pyexpat)
OrderedDict([(u'a', u'hello')])
```

xmltodict.**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

CHAPTER TWO

INDICES AND TABLES

- *genindex*
- *modindex*
- *search*

Symbols