PyTan Documentation

Release 2.1.0

Jim Olsen

August 28, 2015

CONTENTS

TABLE OF CONTENTS

1.1 PyTan Introduction

1.1.1 Description

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

- **Tanium Server SOAP API: The SOAP server embedded into the Tanium server itself.**

    - For Tanium version 6.2: The SOAP servers listens on port 444 but is also available via port 443

    - For Tanium version 6.5: The SOAP servers listens on port 443, and is not available on port 444

- **TaniumPy Python Package (:mod:'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: (:mod:'pytan')**: A python package that provides a set of methods to make interfacing with TaniumPy more human friendly. Located in `lib/pytan`

- **PyTan Command Line Scripts**: A set of command line scripts that utilize the PyTan Package (`pytan`) to make it easy for non-programmers to create/get/delete/ask/deploy objects via the Tanium Server SOAP API.

1.1.2 Why it was created

This was created to solve for the following needs:

- Create a python package (`pytan`) to provide a set of methods for making it easier to programmatically interface with Tanium via the SOAP API.

- Create a set of command line scripts utilizing the `pytan` package that handle the argument parsing, thereby providing non-programmers with command line access to the functionality therein.

- Provide a way to ask questions and get results via Python and/or the command line.

- Provide a way to deploy actions and get results via Python and/or the command line.

- Provide a way to export/import objects in JSON via Python and/or the command line.

1.1.3 Requirements

- Python 2.7: To date PyTan has only been qualified against 2.7.6 and 2.7.9 on Mac/Linux/Windows.

- A working install of the Tanium Platform.

1.1.4 Tanium Versions Validated Against

PyTan has been fully tested against the following versions of the Tanium Platform:

- 6.2.314.3315

- 6.2.314.3321

- 6.5.314.4254

- 6.5.314.4268

- 6.5.314.4275

1.1.5 Installation

Windows Installation

- Download Python 2.7.9 from https://www.python.org/downloads/windows/

- Install Python 2.7.9 – 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 2.7.6 out of the box

- Copy PyTan from github to your local machine somewhere

Linux Installation

- Ensure Python 2.7.9 is installed

- Ensure the first python binary in your path points to your Python 2.7 installation

- Copy PyTan from github to your local machine somewhere

1.1.6 Usage

- For command line usage, refer to Command Line Help Index

- For API Examples, refer to the *PyTan API examples*

- For in depth API Documentation, refer to the *pytan package*, especially the *pytan.handler module*

1.1.7 Directory Layout

- **EXAMPLES/ directory**: contains a set of example python files that show how to use the various methods exposed by (`pytan`)

- **BUILD/ directory**: contains the scripts that build the HTML and PDF documentation in doc/, generate the (`taniumpy`), generate the python examples in EXAMPLES/, generate some of the command line scripts in bin/, and generate all of the documentation for the command line scripts in doc/_static/bin_doc

- **bin/ directory**: contains all of the command line scripts that utilize the (`pytan`)

- **doc/ directory**: contains the HTML and PDF documentation

- **lib/ directory**: contains the python libraries (`pytan`) and (`taniumpy`), as well as other python libraries

- **test/ directory**: contains the unit and functional tests for (`pytan`)

- **winbin/ directory**: contains the Windows batch scripts which wrap around the python command line scripts in bin/

- **ZIP_DIST/ directory**: contains standalone windows executables for certain tools, created by batch files in BUILD/STATICWINBUILD/

1.1.8 Other References

- Tanium Platform Website

- Tanium Knowledge Base

- Tanium SOAP Knowledge Base Article

- The console.wsdl used to build the taniumpy library for this version, also useful as a reference tool.

1.2 pytan package

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

pytan.**__version__** = '2.1.0'
>   Version of PyTan

pytan.**__copyright__** = 'Copyright 2015 Tanium'
>   Copyright for PyTan

pytan.**__license__** = 'MIT'
>   License for PyTan

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

1.2.1 pytan.handler module

The main `pytan` module that provides first level entities for programmatic use.

Handler Class

class pytan.handler.**Handler**(*username=None*, *password=None*, *host=None*, *port=443*, *loglevel=0*, *debugformat=False*, *gmt_log=True*, *session_id=None*, ***kwargs*)

>   Bases: `object`

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

>   >   **Parameters username** : str

>   >   >   - default: None

>   >   >   - *username* to connect to *host* with

>   >   **password** : str

- default: None

- *password* to connect to *host* with

host : str

- default: None

- hostname or ip of Tanium SOAP Server

port : int, optional

- default: 443

- port of Tanium SOAP Server on *host*

loglevel : int, optional

- default: 0

- 0 do not print anything except warnings/errors

- 1 and higher will print more

debugformat : bool, optional

- default: False

- False: use one line logformat

- True: use two lines

gmt_log : bool, optional

- default: True

- True: use GMT timezone for log output

- False: use local time for log output

session_id : str, optional

- default: None

- session_id to use while authenticating instead of username/password

Other Parameters **http_debug** : bool, optional

- default: False

- False: do not print requests package debug

- True: do print requests package debug

- This is passed through to `pytan.sessions.Session`

http_auth_retry: bool, optional

- default: True

- True: retry HTTP GET/POST's

- False: do not retry HTTP GET/POST's

- This is passed through to `pytan.sessions.Session`

http_retry_count: int, optional

- default: 5

- number of times to retry HTTP GET/POST's if the connection times out/fails

- This is passed through to `pytan.sessions.Session`

soap_request_headers : dict, optional

- default: {'Content-Type': 'text/xml; charset=utf-8', 'Accept-Encoding': 'gzip'}

- dictionary of headers to add to every HTTP GET/POST

- This is passed through to `pytan.sessions.Session`

auth_connect_timeout_sec : int, optional

- default: 5

- number of seconds before timing out for a connection while authenticating

- This is passed through to `pytan.sessions.Session`

auth_response_timeout_sec : int, optional

- default: 15

- number of seconds before timing out for a response while authenticating

- This is passed through to `pytan.sessions.Session`

info_connect_timeout_sec : int, optional

- default: 5

- number of seconds before timing out for a connection while getting /info.json

- This is passed through to `pytan.sessions.Session`

info_response_timeout_sec : int, optional

- default: 15

- number of seconds before timing out for a response while getting /info.json

- This is passed through to `pytan.sessions.Session`

soap_connect_timeout_sec : int, optional

- default: 15

- number of seconds before timing out for a connection for a SOAP request

- This is passed through to `pytan.sessions.Session`

soap_response_timeout_sec : int, optional

- default: 540

- number of seconds before timing out for a response for a SOAP request

- This is passed through to `pytan.sessions.Session`

stats_loop_enabled : bool, optional

- default: False

- False: do not enable the statistics loop thread

- True: enable the statistics loop thread

- This is passed through to `pytan.sessions.Session`

stats_loop_sleep_sec : int, optional

- default: 5

- number of seconds to sleep in between printing the statistics when stats_loop_enabled is True

- This is passed through to `pytan.sessions.Session`

record_all_requests: bool, optional

- default: False

- False: do not add each requests response object to session.ALL_REQUESTS_RESPONSES

- True: add each requests response object to session.ALL_REQUESTS_RESPONSES

- This is passed through to `pytan.sessions.Session`

stats_loop_targets : list of dict, optional

- default: [{'Version': 'Settings/Version'}, {'Active Questions': 'Active Question Cache/Active Question Estimate'}, {'Clients': 'Active Question Cache/Active Client Estimate'}, {'Strings': 'String Cache/Total String Count'}, {'Handles': 'System Performance Info/HandleCount'}, {'Processes': 'System Performance Info/ProcessCount'}, {'Memory Available': 'percentage(System Performance Info/PhysicalAvailable,System Performance Info/PhysicalTotal)'}]

- list of dictionaries with the key being the section of info.json to print info from, and the value being the item with in that section to print the value

- This is passed through to `pytan.sessions.Session`

persistent: bool, optional

- default: False

- False: do not request a persistent session

- True: do request a persistent

- This is passed through to `pytan.sessions.Session.authenticate()`

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

`taniumpy.session.Session` Session object used by Handler

Notes

- for 6.2: 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. However, port 444 is the only port that exposes the /info page in 6.2

- for 6.5: port 443 is the default SOAP port, there is no port 444

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, optional
>
> > • default: 'manual'
> >
> > • type of question to ask: {'saved', 'manual', '_manual'}
>
> **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()
>
> **pytan.handler.Handler.ask_saved()** method used when qtype == 'saved'
>
> **pytan.handler.Handler.ask_manual()** method used when qtype == 'manual'
>
> **pytan.handler.Handler._ask_manual()** method used when qtype == '_manual'

Ask a Saved Question

Handler.**ask_saved**(*refresh_data=False*, ***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
>
> **refresh_data: bool, optional**
>
> > • default False
> >
> > • False: do not perform a getResultInfo before issuing a getResultData
> >
> > • True: perform a getResultInfo before issuing a getResultData
>
> **Returns ret** : dict, containing
>
> > • *question_object* : taniumpy.object_types.saved_question.SavedQuestion, the saved question object

- *question_object* : `taniumpy.object_types.question.Question`, the question asked by *saved_question_object*

- *question_results* : `taniumpy.object_types.result_set.ResultSet`, the results for *question_object*

- *poller_object* : None if *refresh_data* == False, elsewise `pytan.pollers.QuestionPoller`, poller object used to wait until all results are in before getting *question_results*,

- *poller_success* : None if *refresh_data* == False, elsewise True or False

Notes

id or name must be supplied

Asking a Manual Question

`Handler.`**`ask_manual`**`(**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
> >
> > > - default: []
> > >
> > > - sensors (columns) to include in question
> > >
> > > **question_filters** : str, list of str, optional
> > >
> > > - default: []
> > >
> > > - filters that apply to the whole question
> > >
> > > **question_options** : str, list of str, optional
> > >
> > > - default: []
> > >
> > > - options that apply to the whole question
> > >
> > > **get_results** : bool, optional
> > >
> > > - default: True
> > >
> > > - True: wait for result completion after asking question
> > >
> > > - False: just ask the question and return it in result
> > >
> > > **sensors_help** : bool, optional
> > >
> > > - default: False
> > >
> > > - False: do not print the help string for sensors
> > >
> > > - True: print the help string for sensors and exit
> > >
> > > **filters_help** : bool, optional
> > >
> > > - default: False
> > >
> > > - False: do not print the help string for filters
> > >
> > > - True: print the help string for filters and exit

> **options_help** : bool, optional
>
> - default: False
>
> - 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`, the actual question created and added by PyTan
>
> - *question_results* : `taniumpy.object_types.result_set.ResultSet`, the Result Set for *question_object* if *get_results* == True
>
> - *poller_object* : `pytan.pollers.QuestionPoller`, poller object used to wait until all results are in before getting *question_results*
>
> - *poller_success* : None if *get_results* == True, elsewise True or False

See also:

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

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

`pytan.handler.Handler._ask_manual()` private method with the actual workflow used to create and add the question object

Notes

When asking a question from the Tanium console, you construct a question like:

> Get Computer Name and IP Route Details from all machines with Is Windows containing "True"

Asking the same question in PyTan has some similarities:

```
>>> r = handler.ask_manual(sensors=['Computer Name', 'IP Route Details'], question_filters=['Is
```

There are two sensors in this question, after the "Get" and before the "from all machines": "Computer Name" and "IP Route Details". The sensors after the "Get" and before the "from all machines" can be referred to as any number of things:

> •sensors
>
> •left hand side
>
> •column selects

The sensors that are defined after the "Get" and before the "from all machines" are best described as a column selection, and control what columns you want to show up in your results. These sensor names are the same ones that would need to be passed into ask_question() for the sensors arguments.

You can filter your column selections by using a filter in the console like so:

> Get Computer Name starting with "finance" and IP Route Details from all machines with Is Windows containing "True"

And in PyTan:

```
>>> r = handler.ask_manual(sensors=['Computer Name, that starts with:finance', 'IP Route Detail
```

This will cause the results to have the same number of columns, but for any machine that returns results that do not match the filter specified for a given sensor, the row for that column will contain "[no results]".

There is also a sensor specified after the "from all machines with": "Is Windows". This sensor can be referred to as any number of things:

- question filters

- sensors (also)

- right hand side

- row selects

Any system that does not match the conditions in the question filters will return no results at all. These question filters are really just sensors all over again, but instead of controlling what columns are output in the results, they control what rows are output in the results.

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**(***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

- package to deploy with this action

action_filters : str, list of str, optional

- default: []

- 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

- default: []

- options to apply to *action_filters*

start_seconds_from_now : int, optional

- default: 0

- start action N seconds from now

expire_seconds : int, optional

- default: package.expire_seconds

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

run : bool, optional

- default: False

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

- True: actually deploy the action

get_results : bool, optional

- default: True

- True: wait for result completion after deploying action

- False: just deploy the action and return the object in *ret*

package_help : bool, optional

- default: False

- False: do not print the help string for package

- True: print the help string for package and exit

filters_help : bool, optional

- default: False

- False: do not print the help string for filters

- True: print the help string for filters and exit

options_help : bool, optional

- default: False

- False: do not print the help string for options

- True: print the help string for options and exit

Returns **ret** : dict, containing:

- *saved_action_object* : `taniumpy.object_types.saved_action.SavedAction`, the saved_action added for this action (None if 6.2)

- *action_object* : `taniumpy.object_types.action.Action`, the action object that tanium created for *saved_action*

- *package_object* : `taniumpy.object_types.package_spec.PackageSPec`, the package object used in *saved_action*

- *action_info* : `taniumpy.object_types.result_info.ResultInfo`, the initial GetResultInfo call done before getting results

- *poller_object* : `pytan.pollers.ActionPoller`, poller object used to wait until all results are in before getting *action_results*

- *poller_success* : None if *get_results* == False, elsewise True or False

- *action_results* : None if *get_results* == False, elsewise `taniumpy.object_types.result_set.ResultSet`, the results for *action_object*

- *action_result_map* : None if *get_results* == False, elsewise progress map for *action_object* in dictionary form

See also:

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

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

`pytan.handler.Handler._deploy_action()` private method with the actual workflow used to create and add the action object

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']
```

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

Export Object

Handler.**export_obj**(*obj*, *export_format='csv'*, ***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, optional

>>> • default: 'csv'

>>> • the format to export *obj* to, one of: {'csv', 'xml', 'json'}

>> **header_sort** : list of str, bool, optional

>>> • default: True

>>> • 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

>>> • default: False

>>> • 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

>>> • default: False

>>> • 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

- default: False
- 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

- default: False
- 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

- default: False
- 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

Notes

When performing a CSV export and importing that CSV into excel, keep in mind that Excel has a per cell character limit of 32,000. Any cell larger than that will be broken up into a whole new row, which can wreak havoc with data in Excel.

Export Object to Report File
Handler.**export_to_report_file**(*obj*, *export_format='csv'*, ***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, optional

- default: 'csv'
- the format to export *obj* to, one of: {'csv', 'xml', 'json'}

header_sort : list of str, bool, optional

- default: True
- 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

- default: False
- 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

- default: False
- 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

- default: False
- 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

- default: False
- 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

- default: False
- 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

- default: None
- filename to save report as, will be automatically generated if not supplied

report_dir: str, optional

- default: None

- directory to save report in, will use current working directory if not supplied

prefix: str, optional

- default: ''

- prefix to add to *report_file*

postfix: str, optional

- default: ''

- postfix to add to *report_file*

Returns report_path, result : tuple

- report_path : str, the full path to the file created with contents of *result*

- result : str, the contents written to report_path

See also:

`pytan.handler.Handler.export_obj()` method that performs the actual work to do the exporting

Notes

When performing a CSV export and importing that CSV into excel, keep in mind that Excel has a per cell character limit of 32,000. Any cell larger than that will be broken up into a whole new row, which can wreak havoc with data in Excel.

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

- default: []

- each string must describe a filter

filter_options : str or list of str, optional

- default: []

- each string must describe an option for *filters*

filters_help : bool, optional

- default: False

- False: do not print the help string for filters

- True: print the help string for filters and exit

options_help : bool, optional

- default: False

- 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

- default: []

- 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

- default: 600

- timeout for command execution in seconds

parameters_json_file : str, optional

- default: ''

- path to json file describing parameters for package

expire_seconds : int, optional

- default: 600

- timeout for action expiry in seconds

verify_filters : str or list of str, optional

- default: []

- each string must describe a filter to be used to verify the package

verify_filter_options : str or list of str, optional

- default: []

- each string must describe an option for *verify_filters*

verify_expire_seconds : int, optional

- default: 600

- timeout for verify action expiry in seconds

filters_help : bool, optional

- default: False

- False: do not print the help string for filters

- True: print the help string for filters and exit

options_help : bool, optional

- default: False

- False: do not print the help string for options

- True: print the help string for options and exit

metadata: list of list of strs, optional

- default: []

- each list must be a 2 item list:

- list item 1 property name

- list item 2 property value

Returns **package_obj** : `taniumpy.object_types.package_spec.PackageSpec`

- TaniumPy object added to Tanium SOAP Server

See also:

`pytan.constants.FILTER_MAPS` valid filters for verify_filters

`pytan.constants.OPTION_MAPS` valid options for verify_filter_options

Create a Sensor

`Handler.`**`create_sensor`**`()`
    Create a sensor object

Raises **pytan.exceptions.HandlerError** : pytan.utils.pytan.exceptions.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

  - default: []

  - name(s) of roles to add to user

roleid : int or list of int, optional

  - default: []

  - id(s) of roles to add to user

properties: list of list of strs, optional

  - default: []

  - 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

  - default: False

  - False: *url* is not a regex pattern

  - True: *url* is a regex pattern

  download_seconds : int, optional

  - default: 86400

  - how often to re-download *url*

  properties: list of list of strs, optional

  - default: []

  - 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

Create a Dashboard

Handler.**create_dashboard**(*name*, *text=''*, *group=''*, *public_flag=True*)

> Calls `pytan.handler.Handler.run_plugin()` to run the CreateDashboard plugin and parse the response

> > **Parameters name** : str
> >
> > > • name of dashboard to create
> >
> > **text** : str, optional
> >
> > > • default: ''
> > >
> > > • text for this dashboard
> >
> > **group** : str, optional
> >
> > > • default: ''
> > >
> > > • group name for this dashboard
> >
> > **public_flag** : bool, optional
> >
> > > • default: True
> > >
> > > • True: make this dashboard public
> > >
> > > • False: do not make this dashboard public
> >
> > **Returns plugin_result, sql_zipped** : tuple
> >
> > > • plugin_result will be the taniumpy object representation of the SOAP response from Tanium server
> > >
> > > • sql_zipped will be a dict with the SQL results embedded in the SOAP response

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

Delete a Dashboard

Handler.**delete_dashboard**(*name*)

> Calls `pytan.handler.Handler.run_plugin()` to run the DeleteDashboards plugin and parse the response

>> **Parameters name** : str

>>> • name of dashboard to delete

>> **Returns plugin_result, sql_zipped** : tuple

>>> • plugin_result will be the taniumpy object representation of the SOAP response from Tanium server

>>> • sql_zipped will be a dict with the SQL results embedded in the SOAP response

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

>> **Returns obj_list** : `taniumpy.object_types.base.BaseType`

>>> • The object list of items found for *objtype*

> See also:

> `pytan.constants.GET_OBJ_MAP` maps objtype to supported 'search' keys

> `pytan.handler.Handler._get_multi()` private method used to get multiple items

> `pytan.handler.Handler._get_single()` private method used to get singular items

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

>> **Returns obj_list** : `taniumpy.object_types.base.BaseType`

>>> • The object list of items found for *objtype*

> See also:

> `pytan.constants.GET_OBJ_MAP` maps objtype to supported 'search' keys

> `pytan.handler.Handler._find()` private method used to find items

Get Dashboards

Handler.**get_dashboards**(*name=''*)

> Calls `pytan.handler.Handler.run_plugin()` to run the GetDashboards plugin and parse the response

> > **Parameters name** : str, optional
> >
> > > • default: ''
> > >
> > > • name of dashboard to get, if empty will return all dashboards
> >
> > **Returns plugin_result, sql_zipped** : tuple
> >
> > > • plugin_result will be the taniumpy object representation of the SOAP response from Tanium server
> > >
> > > • sql_zipped will be a dict with the SQL results embedded in the SOAP response

Get Server Version

Handler.**get_server_version**()

> Uses `taniumpy.session.Session.get_server_version()` to get the version of the Tanium Server

> Updates self.server_version with the return, and updates self.server_version_dict with a parsed version of self.server_version into major, minor, revision, and build.

> > **Returns** self.server_version: str
> >
> > > • Version of Tanium Server in string format

Handler Methods: Getting Result Data / Result Info

Handler.**get_result_data**(*obj*, *aggregate=False*, *shrink=True*, ***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
> >
> > > • default: False
> > >
> > > • False: get all the data
> > >
> > > • True: get just the aggregate data (row counts of matches)
> >
> > **shrink** : bool, optional
> >
> > > • default: True
> > >
> > > • True: Shrink the object down to just id/name/hash attributes (for smaller request)
> > >
> > > • False: Use the full object as is
> >
> > **Returns rd** : `taniumpy.object_types.result_set.ResultSet`
> >
> > > The return of GetResultData for *obj*

Handler.**get_result_data_sse**(*obj*, *export_format='csv'*, *leading=''*, *trailing=''*, ***kwargs*)
Get the result data for a python API object using a server side export (sse)

This method issues a GetResultData command to the SOAP api for *obj* with the option *export_flag* set to 1. This will cause the server to process all of the data for a given result set and save it as *export_format*. Then the user can use an authenticated GET request to get the status of the file via "/export/${export_id}.status". Once the status returns "Completed.", the actual report file can be retrieved by an authenticated GET request to "/export/${export_id}.gz". This workflow saves a lot of processing time and removes the need to paginate large result sets necessary in normal GetResultData calls.

Version support

> - 6.5.314.4231: initial sse support (csv only)
>
> - 6.5.314.4300: export_format support (adds xml and cef)
>
> - 6.5.314.4300: fix core dump if multiple sse done on empty resultset
>
> - 6.5.314.4300: fix no status file if sse done on empty resultset
>
> - 6.5.314.4300: fix response if more than two sse done in same second

> **Parameters obj** : `taniumpy.object_types.base.BaseType`
>
> > - object to get result data for
>
> **export_format** : str, optional
>
> > - default: 'csv'
> >
> > - format to have server create report in, one of: {'csv', 'xml', 'xml_obj', 'cef', 0, 1, 2}
>
> **leading** : str, optional
>
> > - default: ''
> >
> > - used for export_format 'cef' only, the string to prepend to each row
>
> **trailing** : str, optional
>
> > - default: ''
> >
> > - used for export_format 'cef' only, the string to append to each row
>
> **Returns export_data** : either *str* or `taniumpy.object_types.result_set.ResultSet`
>
> > - If export_format is one of csv, xml, or cef, export_data will be a *str* containing the contents of the ResultSet in said format
> >
> > - If export_format is xml_obj, export_data will be a `taniumpy.object_types.result_set.ResultSet`

See also:

`pytan.constants.SSE_FORMAT_MAP` maps *export_format* to an integer for use by the SOAP API

`pytan.constants.SSE_RESTRICT_MAP` maps export_format integers to supported platform versions

`pytan.constants.SSE_CRASH_MAP` maps platform versions that can cause issues in various scenarios

Handler.**get_result_info**(*obj*, *shrink=True*, ***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

shrink : bool, optional

- default: True

- True: Shrink the object down to just id/name/hash attributes (for smaller request)

- False: Use the full object as is

Returns **ri** : `taniumpy.object_types.result_info.ResultInfo`

- The return of GetResultData for *obj*

Handler Methods: Plugins

`Handler.`**`run_plugin`**(*plugin*)

Wrapper around `pytan.session.Session.run_plugin()` to run the plugin and zip up the SQL results into a python dictionary

Returns **plugin_result, sql_zipped** : tuple

- plugin_result will be the taniumpy object representation of the SOAP response from Tanium server

- sql_zipped will be a dict with the SQL results embedded in the SOAP response

Handler Methods: Private Methods

`Handler.`**`_add`**(*api_object*, ***kwargs*)

Wrapper for interfacing with `taniumpy.session.Session.add()`

`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()` instead.

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

- default: []

- sensor definitions

question_filter_defs : dict, list of dict, optional

- default: []

- question filter definitions

question_option_defs : dict, list of dict, optional

- default: []

- question option definitions

get_results : bool, optional

- default: True

- 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`, the actual question created and added by PyTan

- *question_results* : `taniumpy.object_types.result_set.ResultSet`, the Result Set for *question_object* if *get_results* == True

- *poller_object* : `pytan.pollers.QuestionPoller`, poller object used to wait until all results are in before getting *question_results*

- *poller_success* : None if *get_results* == True, elsewise True or False

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 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.**_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()` instead.

Parameters **package_def** : dict

- definition that describes a package

action_filter_defs : str, dict, list of str or dict, optional

- default: []

- action filter definitions

action_option_defs : dict, list of dict, optional

- default: []

- action filter option definitions

start_seconds_from_now : int, optional

- default: 0

- start action N seconds from now

expire_seconds : int, optional

- default: package.expire_seconds

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

run : bool, optional

- default: False

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

- True: actually deploy the action

get_results : bool, optional

- default: True

- True: wait for result completion after deploying action

- False: just deploy the action and return the object in *ret*

action_name : str, optional

- default: prepend package name with "API Deploy "

- custom name for action

action_comment : str, optional

- default:

- custom comment for action

Returns **ret** : dict, containing:

- *saved_action_object* : `taniumpy.object_types.saved_action.SavedAction`, the saved_action added for this action (None if 6.2)

- *action_object* : `taniumpy.object_types.action.Action`, the action object that tanium created for *saved_action*

- *package_object* : `taniumpy.object_types.package_spec.PackageSPec`, the package object used in *saved_action*

- *action_info* : `taniumpy.object_types.result_info.ResultInfo`, the initial GetResultInfo call done before getting results

- *poller_object* : `pytan.pollers.ActionPoller`, poller object used to wait until all results are in before getting *action_results*

- *poller_success* : None if *get_results* == False, elsewise True or False

- *action_results* : None if *get_results* == False, elsewise `taniumpy.object_types.result_set.ResultSet`, the results for *action_object*

- *action_result_map* : None if *get_results* == False, elsewise progress map for *action_object* in dictionary form

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.**_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

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_package_def**(*d*)
    Uses `get()` to update a definition with a package object

Handler.**_get_sensor_defs**(*defs*)
>   Uses `get()` to update a definition with a sensor object

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.**_parse_versioning**()
>   Parses self.server_version into a dictionary

>   >   **Returns** dict, containing major, minor, revision, and build of tanium server version

>   **Notes**

>   If pytan has not yet fetched info.json, then server_version will == "Not yet determined!" force a call to self.session.get_server_version() to attempt to get info.json and parse the version from that and update self.server_version with that value

>   If pytan is unable to fetch info.json properly for some reason, then server_version will == "Unable to determine"

Handler.**_platform_is_6_2**()
>   Check to see if self.server_version_dict matches 6.2.xxx.xxx

>   >   **Returns** bool

>   >   >   • True if self.server_version_dict major == 6 and minor == 2

>   >   >   • False otherwise

Handler.**_version_support_check**(*v_maps*)
>   Checks that each of the version maps in v_maps is greater than or equal to the current servers version

>   >   **Parameters v_maps** : list of dict

>   >   >   • each dict can have major, minor, build, revision as keys, the corresponding values will be checked against self.server_version_dict to see if they are greater or equal to those values

>   >   **Returns** bool

>   >   >   • True if all values in all v_maps are greater than or equal to all values in self.server_version_dict

>   >   >   • False otherwise

Handler.**_check_export_format_support**(*export_format*, *export_format_int*)
>   Determines if the export format integer is supported in the server version

Handler.**_resolve_sse_format**(*export_format*)
>   Resolves the export format the user supplied to an integer for the API

Handler.**_check_sse_version**()
>   Validates that the server version supports server side export

Handler.**_check_sse_crash_prevention**(*obj*)
>   Runs a number of methods used to prevent crashing the platform server when performing server side exports

Handler.**_check_sse_timing**()
>   Checks that the last server side export was at least 1 second ago if server version is less than any versions in pytan.constants.SSE_CRASH_MAP

Handler.**_check_sse_empty_rs**(*obj*)
> Checks if the server version is less than any versions in pytan.constants.SSE_CRASH_MAP, if so verifies that the result set is not empty

1.2.2 pytan.exceptions module

Provides exceptions for the `pytan` module.

Exception Classes

Exceptions used throughout `pytan`:

exception pytan.exceptions.**HandlerError**
> Bases: `exceptions.Exception`
>
> Exception thrown for errors in `pytan.handler`

exception pytan.exceptions.**HumanParserError**
> Bases: `exceptions.Exception`
>
> Exception thrown for errors while parsing human strings from `pytan.handler`

exception pytan.exceptions.**DefinitionParserError**
> Bases: `exceptions.Exception`
>
> Exception thrown for errors while parsing definitions from `pytan.handler`

exception pytan.exceptions.**RunFalse**
> Bases: `exceptions.Exception`
>
> Exception thrown when run=False from `pytan.handler.Handler.deploy_action()`

exception pytan.exceptions.**PytanHelp**
> Bases: `exceptions.Exception`
>
> Exception thrown when printing out help

exception pytan.exceptions.**PollingError**
> Bases: `exceptions.Exception`
>
> Exception thrown for errors in `pytan.polling`

exception pytan.exceptions.**TimeoutException**
> Bases: `exceptions.Exception`
>
> Exception thrown for timeout errors in `pytan.polling`

exception pytan.exceptions.**HttpError**
> Bases: `exceptions.Exception`
>
> Exception thrown for HTTP errors in `pytan.sessions`

exception pytan.exceptions.**AuthorizationError**
> Bases: `exceptions.Exception`
>
> Exception thrown for authorization errors in `pytan.sessions`

exception pytan.exceptions.**BadResponseError**
> Bases: `exceptions.Exception`
>
> Exception thrown for BadResponse messages from Tanium in `pytan.sessions`

exception pytan.exceptions.**NotFoundError**
>   Bases: exceptions.Exception
>
>   Exception thrown for Not Found messages from Tanium in pytan.handler

exception pytan.exceptions.**VersionMismatchError**
>   Bases: exceptions.Exception
>
>   Exception thrown for version_check in pytan.utils

1.2.3 pytan.sessions module

Session classes for the pytan module.

Session Class

class pytan.sessions.**Session**(*server*, *port=443*, ***kwargs*)
>   Bases: object
>
>   This session object uses the requests package instead of the built in httplib library.
>
>   This provides support for keep alive, gzip, cookies, forwarding, and a host of other features automatically.

Example: Create a Session object

Setup a Session() object:

```
>>> import sys
>>> sys.path.append('/path/to/pytan/')
>>> import pytan
>>> session = pytan.sessions.Session('host')
```

Authenticate with the Session() object:

```
>>> session.authenticate('username', 'password')
```

Session Attributes

Session.**REQ_SESSION** = <requests.sessions.Session object at 0x102eb26d0>
>   The Requests session allows you to persist certain parameters across requests. It also persists cookies across all
>   requests made from the Session instance. Any requests that you make within a session will automatically reuse
>   the appropriate connection

Session.**XMLNS** = {'xsi': 'xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"', 'typens': 'xmlns:typens="urn:Taniu
>   The namespace mappings for use in XML Request bodies

Session.**REQUEST_BODY_BASE** = '<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
>   The XML template used for all SOAP Requests in string form

Session.**REQUEST_BODY_TEMPLATE** = <string.Template object at 0x10312e9d0>
>   The XML template used for all SOAP Requests in string.template form

Session.**GET_OBJECT_CMD = 'GetObject'**
> The text used in the command element for XML requests to get objects

Session.**UPDATE_OBJECT_CMD = 'UpdateObject'**
> The text used in the command element for XML requests to update objects

Session.**ADD_OBJECT_CMD = 'AddObject'**
> The text used in the command element for XML requests to add objects

Session.**DELETE_OBJECT_CMD = 'DeleteObject'**
> The text used in the command element for XML requests to delete objects

Session.**GET_RESULT_INFO_CMD = 'GetResultInfo'**
> The text used in the command element for XML requests to get result info for an object

Session.**GET_RESULT_DATA_CMD = 'GetResultData'**
> The text used in the command element for XML requests to get result data for an object

Session.**RUN_PLUGIN_CMD = 'RunPlugin'**
> The text used in the command element for XML requests to run a plugin for an object

Session.**AUTH_RES = 'auth'**
> The URL to use for authentication requests

Session.**SOAP_RES = 'soap'**
> The URL to use for SOAP requests

Session.**INFO_RES = 'info.json'**
> The URL to use for server info requests

Session.**AUTH_CONNECT_TIMEOUT_SEC = 5**
> number of seconds before timing out for a connection while authenticating

Session.**AUTH_RESPONSE_TIMEOUT_SEC = 15**
> number of seconds before timing out for a response while authenticating

Session.**INFO_CONNECT_TIMEOUT_SEC = 5**
> number of seconds before timing out for a connection while getting server info

Session.**INFO_RESPONSE_TIMEOUT_SEC = 15**
> number of seconds before timing out for a response while getting server info

Session.**SOAP_CONNECT_TIMEOUT_SEC = 15**
> number of seconds before timing out for a connection while sending a SOAP Request

Session.**SOAP_RESPONSE_TIMEOUT_SEC = 540**
> number of seconds before timing out for a response while sending a SOAP request

Session.**SOAP_REQUEST_HEADERS = {'Content-Type': 'text/xml; charset=utf-8', 'Accept-Encoding': 'gzip'}**
> dictionary of headers to add to every HTTP GET/POST

Session.**COMMAND_RE = <_sre.SRE_Pattern object at 0x102ec2ae0>**
> regex object to search for command element in XML bodies

Session.**SESSION_RE = <_sre.SRE_Pattern object at 0x102ec2c10>**
> regex object to search for session element in XML bodies

Session.**VERSION_RE = <_sre.SRE_Pattern object at 0x102e4f370>**
> regex object to search for server_version element in XML bodies

Session.**HTTP_DEBUG = False**
> print requests package debug or not

Session.**HTTP_RETRY_COUNT = 5**
> number of times to retry HTTP GET/POST's if the connection times out/fails

Session.**HTTP_AUTH_RETRY = True**
> retry HTTP GET/POST's with username/password if session_id fails or not

Session.**STATS_LOOP_ENABLED = False**
> enable the statistics loop thread or not

Session.**STATS_LOOP_SLEEP_SEC = 5**
> number of seconds to sleep in between printing the statistics when stats_loop_enabled is True

Session.**STATS_LOOP_TARGETS = [{'Version': 'Settings/Version'}, {'Active Questions': 'Active Question Cache/Active Que**
> list of dictionaries with the key being the section of info.json to print info from, and the value being the item
> with in that section to print the value

Session Methods

Authentication
Session.**authenticate**(*username=None*, *password=None*, *session_id=None*, ***kwargs*)
> Authenticate against a Tanium Server using a username/password or a session ID

>> **Parameters username** : str, optional
>>
>>> • default: None
>>>
>>> • username to authenticate as
>>
>> **password** : str, optional
>>
>>> • default: None
>>>
>>> • password for *username*
>>
>> **session_id** : str, optional
>>
>>> • default: None
>>>
>>> • session_id to authenticate with, this will be used in favor of username/password if all 3 are
>>>   supplied.
>>
>> **persistent: bool, optional**
>>
>>> • default: False
>>>
>>> • False: do not request a persistent session (returns a session_id that expires 5 minutes after
>>>   last use)
>>>
>>> • True: do request a persistent (returns a session_id that expires 1 week after last use)

Notes

Can request a persistent session that will last up to 1 week when authenticating with username and password.

New persistent sessions may be handed out by the Tanium server when the session handed by this auth call is used to login with that week. The new session must be used to login, as no matter what persistent sessions will expire 1 week after issuance (or when logout is called with that session, or when logout with all_sessions=True is called for any session for this user)

the way sessions get issued:

> •a GET request to /auth is issued

- username/password supplied in headers as base64 encoded, or session is supplied in headers as string

- session is returned upon successful auth

- if there is a header "persistent=1" in the headers, a session that expires after 1 week will be issued if username/password was used to auth. persistent is ignored if session is used to auth.

- if there is not a header "persistent=1" in the headers, a session that expires after 5 minutes will be issued

- if session is used before it expires, it's expiry will be extended by 5 minutes or 1 week, depending on the type of persistence

- while using the SOAP api, new session ID's may be returned as part of the response. these new session ID's should be used in lieu of the old session ID

/auth URL This url is used for validating a server user's credentials. It supports a few different ways to authenticate and returns a SOAP session ID on success. These sessions expire after 5 minutes by default if they aren't used in SOAP requests. This expiration is configured with the server setting 'session_expiration_seconds'.

Supported Authentication Methods:

- HTTP Basic Auth (Clear Text/BASE64)

- Username/Password/Domain Headers (Clear Text)

- Negotiate (NTLM Only)

NTLM is enabled by default in 6.3 or greater and requires a persistent connection until a session is generated.

Session.**logout**(*all_session_ids=False*)

Logout a given session_id from Tanium. If not session_id currently set, it will authenticate to get one.

> **Parameters all_session_ids** : bool, optional
>
> > - default: False
> >
> > - False: only log out the current session id for the current user
> >
> > - True: log out ALL session id's associated for the current user

Session.**is_auth**

Property to determine if there is a valid session_id or username and password stored in this object

> **Returns** bool
>
> > - True: if self._session_id or self._username and _self.password are set
> >
> > - False: if not

Session.**session_id**

Property to fetch the session_id for this object

> **Returns self._session_id** : str

SOAP API Wrappers

Session.**find**(*object_type*, ***kwargs*)

Creates and sends a GetObject XML Request body from *object_type* and parses the response into an appropriate `taniumpy` object

> **Parameters obj** : `taniumpy.object_types.base.BaseType`
>
> > - object to find
>
> **Returns obj** : `taniumpy.object_types.base.BaseType`
>
> > - found objects

Session.**save**(*obj*, ***kwargs*)

    Creates and sends a UpdateObject XML Request body from *obj* and parses the response into an appropriate `taniumpy` object

        Parameters obj : `taniumpy.object_types.base.BaseType`

            • object to save

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

            • saved object

Session.**add**(*obj*, ***kwargs*)

    Creates and sends a AddObject XML Request body from *obj* and parses the response into an appropriate `taniumpy` object

        Parameters obj : `taniumpy.object_types.base.BaseType`

            • object to add

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

            • added object

Session.**delete**(*obj*, ***kwargs*)

    Creates and sends a DeleteObject XML Request body from *obj* and parses the response into an appropriate `taniumpy` object

        Parameters obj : `taniumpy.object_types.base.BaseType`

            • object to delete

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

            • deleted object

Session.**run_plugin**(*obj*, ***kwargs*)

    Creates and sends a RunPlugin XML Request body from *obj* and parses the response into an appropriate `taniumpy` object

        Parameters obj : `taniumpy.object_types.base.BaseType`

            • object to run

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

            • results from running object

Session.**get_result_info**(*obj*, ***kwargs*)

    Creates and sends a GetResultInfo XML Request body from *obj* and parses the response into an appropriate `taniumpy` object

        Parameters obj : `taniumpy.object_types.base.BaseType`

            • object to get result info for

        Returns obj : `taniumpy.object_types.result_info.ResultInfo`

            • ResultInfo for *obj*

Session.**get_result_data**(*obj*, ***kwargs*)

    Creates and sends a GetResultData XML Request body from *obj* and parses the response into an appropriate `taniumpy` object

        Parameters obj : `taniumpy.object_types.base.BaseType`

            • object to get result set for

Returns obj : [`taniumpy.object_types.result_set.ResultSet`](#) or str

- if export_id element found in *obj*, element text value is extracted and returned as str

- otherwise, *obj* will be the ResultSet for *obj*

Server Info / Version / Stats

Session.**get_server_info**(*port=None*, *fallback_port=444*, ***kwargs*)

Gets the /info.json

Parameters port : int, optional

- default: None

- port to attempt getting /info.json from, if not specified will use self.port

fallback_port : int, optional

- default: 444

- fallback port to attempt getting /info.json from if *port* fails

Returns info_dict : dict

- raw json response converted into python dict

- 'diags_flat': info.json flattened out into an easier to use structure for python handling

- 'server_info_pass_msgs': messages about successfully retrieving info.json

- 'server_info_fail_msgs': messages about failing to retrieve info.json

See also:

[`pytan.sessions.Session._flatten_server_info()`](#) method to flatten the dictionary received from info.json into a python friendly format

Notes

- 6.2 /info.json is only available on soap port (default port: 444)

- 6.5 /info.json is only available on server port (default port: 443)

Session.**get_server_version**()

Tries to parse the server version from /info.json

Returns str

- str containing server version from /info.json

Session.**get_server_stats**()

Creates a str containing a number of stats gathered from /info.json

Returns str

- str containing stats from /info.json

See also:

[`pytan.sessions.Session.STATS_LOOP_TARGETS`](#) list of dict containing stat keys to pull from /info.json

Session.**enable_stats_loop**(*sleep=None*)

Enables the stats loop thread, which will print out the results of pytan.sessions.Session.get_server_stats() every pytan.sessions.Session.STATS_LOOP_SLEEP_S

> **Parameters sleep** : int, optional
>
> > - when enabling the stats loop, update pytan.sessions.Session.STATS_LOOP_SLEEP_SEC with *sleep*

> **See also:**

> **pytan.sessions.Session._stats_loop()** method started as a thread which checks self.STATS_LOOP_ENABLED before running pytan.sessions.Session.get_server_stats()

Session.**disable_stats_loop**(*sleep=None*)

Disables the stats loop thread, which will print out the results of pytan.sessions.Session.get_server_stats() every pytan.sessions.Session.STATS_LOOP_SLEEP_S

> **Parameters sleep** : int, optional
>
> > - when disabling the stats loop, update pytan.sessions.Session.STATS_LOOP_SLEEP_SEC with *sleep*

> **See also:**

> **pytan.sessions.Session._stats_loop()** method started as a thread which checks self.STATS_LOOP_ENABLED before running pytan.sessions.Session.get_server_stats()

HTTP GET / POST

Session.**http_get**(*url*, ***kwargs*)

This is an authenticated HTTP GET method. It will always forcibly use the authentication credentials that are stored in the current object when performing an HTTP GET.

> **Parameters url** : str
>
> > - url to fetch on the server
>
> **host: str, optional**
>
> > - default: self.server
> >
> > - host to connect to
>
> **port: int, optional**
>
> > - default: self.port
> >
> > - port to connect to
>
> **headers: dict, optional**
>
> > - default: {}
> >
> > - headers to supply as part of GET request
>
> **connect_timeout: int, optional**
>
> > - default: self.SOAP_CONNECT_TIMEOUT_SEC
> >
> > - timeout in seconds for connection to host
>
> **response_timeout: int, optional**

- default: self.SOAP_RESPONSE_TIMEOUT_SEC
- timeout in seconds for response from host

debug: bool, optional

- default: self.HTTP_DEBUG
- False: do not print requests debug messages
- True: print requests debug messages

auth_retry: bool, optional

- default: self.HTTP_AUTH_RETRY
- True: retry authentication with username/password if session_id fails
- False: throw exception if session_id fails

retry_count: int, optional

- default: self.HTTP_RETRY_COUNT
- number of times to retry the GET request if the server fails to respond properly or in time

Returns body : str

- str containing body of response from server

See also:

pytan.sessions.Session._http_get() private method used to perform the actual HTTP GET

Session.**_http_get**(*host*, *port*, *url*, *headers=None*, *connect_timeout=15*, *response_timeout=180*, *de-*
                     bug=False)
    This is an HTTP GET method that utilizes the requests package.

Parameters host: str

- host to connect to

port: int

- port to connect to

url : str

- url to fetch on the server

headers: dict, optional

- default: None
- headers to supply as part of POST request

connect_timeout: int, optional

- default: 15
- timeout in seconds for connection to host

response_timeout: int, optional

- default: 180
- timeout in seconds for response from host

debug: bool, optional

- default: False

- False: do not print requests debug messages

- True: print requests debug messages

Returns body : str

- str containing body of response from server

Session.**http_post**(***kwargs*)

This is an authenticated HTTP POST method. It will always forcibly use the authentication credentials that are stored in the current object when performing an HTTP POST.

Parameters url : str, optional

- default: self.SOAP_RES

- url to fetch on the server

host: str, optional

- default: self.server

- host to connect to

port: int, optional

- default: self.port

- port to connect to

headers: dict, optional

- default: {}

- headers to supply as part of POST request

body: str, optional

- default: ''

- body to send as part of the POST request

connect_timeout: int, optional

- default: self.SOAP_CONNECT_TIMEOUT_SEC

- timeout in seconds for connection to host

response_timeout: int, optional

- default: self.SOAP_RESPONSE_TIMEOUT_SEC

- timeout in seconds for response from host

debug: bool, optional

- default: self.HTTP_DEBUG

- False: do not print requests debug messages

- True: print requests debug messages

auth_retry: bool, optional

- default: self.HTTP_AUTH_RETRY

- True: retry authentication with username/password if session_id fails

- False: throw exception if session_id fails

retry_count: int, optional

> - default: self.HTTP_RETRY_COUNT
> - number of times to retry the POST request if the server fails to respond properly or in time

> **Returns body** : str

> > - str containing body of response from server

See also:

pytan.sessions.Session._http_post() private method used to perform the actual HTTP POST

Session._**http_post**(*host*, *port*, *url*, *body=None*, *headers=None*, *connect_timeout=15*, *response_timeout=180*, *debug=False*)
This is an HTTP POST method that utilizes the requests package.

> **Parameters host: str**

> > - host to connect to

> **port: int**

> > - port to connect to

> **url** : str

> > - url to fetch on the server

> **body: str, optional**

> > - default: None
> > - body to send as part of the POST request

> **headers: dict, optional**

> > - default: None
> > - headers to supply as part of POST request

> **connect_timeout: int, optional**

> > - default: 15
> > - timeout in seconds for connection to host

> **response_timeout: int, optional**

> > - default: 180
> > - timeout in seconds for response from host

> **debug: bool, optional**

> > - default: False
> > - False: do not print requests debug messages
> > - True: print requests debug messages

> **Returns body** : str

> > - str containing body of response from server

See also:

pytan.xml_clean.xml_cleaner() function to remove invalid/bad characters from XML responses

Utility Methods

`Session.`**`_replace_auth`**(*headers*)

> Utility method for removing username, password, and/or session from supplied headers and replacing them with the current objects session or username and password

>> **Parameters headers** : dict

>>> • dict of key/value pairs for a set of headers for a given request

>> **Returns headers** : dict

>>> • dict of key/value pairs for a set of headers for a given request

`Session.`**`_full_url`**(*url*, ***kwargs*)

> Utility method for constructing a full url

>> **Parameters url** : str

>>> • url to use in string

>> **host** : str, optional

>>> • default: self.server

>>> • hostname/IP address to use in string

>> **port** : str, optional

>>> • default: self.port

>>> • port to use in string

>> **Returns full_url** : str

>>> • full url in the form of https://$host:$port/$url

`Session.`**`_clean_headers`**(*headers=None*)

> Utility method for getting the headers for the current request, combining them with the session headers used for every request, and obfuscating the value of any 'password' header.

>> **Parameters headers** : dict

>>> • dict of key/value pairs for a set of headers for a given request

>> **Returns headers** : dict

>>> • dict of key/value pairs for a set of cleaned headers for a given request

`Session.`**`_start_stats_thread`**()

> Utility method starting the `pytan.sessions.Session._stats_loop()` method in a threaded daemon

`Session.`**`_stats_loop`**()

> Utility method for logging server stats via `pytan.sessions.Session.get_server_stats()` every self.STATS_LOOP_SLEEP_SEC

`Session.`**`_flatten_server_info`**(*structure*)

> Utility method for flattening the JSON structure for info.json into a more python usable format

>> **Parameters structure**

>>> • dict/tuple/list to flatten

>> **Returns** flattened

>>> • the dict/tuple/list flattened out

`Session.`**`_get_percentage`**(*part*, *whole*)

> Utility method for getting percentage of part out of whole

> **Parameters part: int, float**
>
> > **whole: int, float**
>
> **Returns str** : the percentage of part out of whole in 2 decimal places

Session.**_find_stat_target**(*target*, *diags*)

> Utility method for finding a target in info.json and returning the value, optionally performing a percentage calculation on two values if the target[0] starts with percentage(
>
> > **Parameters target** : list
> >
> > > • index0 : label : human friendly name to refer to search_path
> > >
> > > • index1 : search_path : / seperated search path to find a given value from info.json
> > >
> > > **diags** : dict
> > >
> > > > • flattened dictionary of info.json diagnostics
> > >
> > > **Returns** dict
> > >
> > > > • label : same as provided in *target* index0 (label)
> > > >
> > > > • result : value resolved from `pytan.sessions.Session._resolve_stat_target()` for *target* index1 (search_path)

Session.**_resolve_stat_target**(*search_path*, *diags*)

> Utility method for resolving the value of search_path in info.json and returning the value
>
> > **Parameters search_path** : str
> >
> > > • / seperated search path to find a given value from info.json
> > >
> > > **diags** : dict
> > >
> > > > • flattened dictionary of info.json diagnostics
> > >
> > > **Returns** str
> > >
> > > > • value resolved from *diags* for *search_path*

Session.**_build_body**(*command*, *object_list*, ***kwargs*)

> Utility method for building an XML Request Body
>
> > **Parameters command** : str
> >
> > > • text to use in command node when building template
> > >
> > > **object_list** : str
> > >
> > > > • XML string to use in object list node when building template
> > >
> > > **kwargs** : dict, optional
> > >
> > > > • any number of attributes that can be set via `taniumpy.object_types.options.Options` that control the servers response.
> > >
> > > **Returns body** : str
> > >
> > > > • The XML request body created from the string.template self.REQUEST_BODY_TEMPLATE

Session.**_create_run_plugin_object_body**(*obj*, ***kwargs*)

> Utility method for building an XML Request Body to run a plugin
>
> > **Parameters obj** : `taniumpy.object_types.base.BaseType`
> >
> > > • object to convert into XML

    kwargs : dict, optional

- any number of attributes that can be set via `taniumpy.object_types.options.Options` that control the servers response.

    Returns obj_body : str

- The XML request body created from `pytan.sessions.Session._build_body()`

`Session.`**`_create_add_object_body`**(*obj*, ***kwargs*)

    Utility method for building an XML Request Body to add an object

    Parameters obj : `taniumpy.object_types.base.BaseType`

- object to convert into XML

    kwargs : dict, optional

- any number of attributes that can be set via `taniumpy.object_types.options.Options` that control the servers response.

    Returns obj_body : str

- The XML request body created from `pytan.sessions.Session._build_body()`

`Session.`**`_create_delete_object_body`**(*obj*, ***kwargs*)

    Utility method for building an XML Request Body to delete an object

    Parameters obj : `taniumpy.object_types.base.BaseType`

- object to convert into XML

    kwargs : dict, optional

- any number of attributes that can be set via `taniumpy.object_types.options.Options` that control the servers response.

    Returns obj_body : str

- The XML request body created from `pytan.sessions.Session._build_body()`

`Session.`**`_create_get_result_info_body`**(*obj*, ***kwargs*)

    Utility method for building an XML Request Body to get result info for an object

    Parameters obj : `taniumpy.object_types.base.BaseType`

- object to convert into XML

    kwargs : dict, optional

- any number of attributes that can be set via `taniumpy.object_types.options.Options` that control the servers response.

    Returns obj_body : str

- The XML request body created from `pytan.sessions.Session._build_body()`

`Session.`**`_create_get_result_data_body`**(*obj*, ***kwargs*)

    Utility method for building an XML Request Body to get result data for an object

    Parameters obj : `taniumpy.object_types.base.BaseType`

- object to convert into XML

    kwargs : dict, optional

- any number of attributes that can be set via `taniumpy.object_types.options.Options` that control the servers response.

Returns obj_body : str

> • The XML request body created from `pytan.sessions.Session._build_body()`

Session.**_create_get_object_body**(*object_or_type*, ***kwargs*)

> Utility method for building an XML Request Body to get an object

> **Parameters obj** : `taniumpy.object_types.base.BaseType`
>
>> • object to convert into XML
>
> **kwargs** : dict, optional
>
>> • any number of attributes that can be set via `taniumpy.object_types.options.Options` that control the servers response.
>
> **Returns obj_body** : str
>
>> • The XML request body created from `pytan.sessions.Session._build_body()`

Session.**_create_update_object_body**(*obj*, ***kwargs*)

> Utility method for building an XML Request Body to update an object

> **Parameters obj** : `taniumpy.object_types.base.BaseType`
>
>> • object to convert into XML
>
> **kwargs** : dict, optional
>
>> • any number of attributes that can be set via `taniumpy.object_types.options.Options` that control the servers response.
>
> **Returns obj_body** : str
>
>> • The XML request body created from `pytan.sessions.Session._build_body()`

Session.**_check_auth**()

> Utility method to check if authentication has been done yet, and throw an exception if not

Session.**_parse_response_for_regex**(*body*, *regex*, *fail=True*)

> Utility method to use a regex to get an element from an XML body

> **Parameters body** : str
>
>> • XML to search
>
> **regex** : re object
>
>> • regex object to search for in body
>
> **fail** : bool, optional
>
>> • default: True
>>
>> • True: throw exception if unable to find any matches for *regex* in *body*
>>
>> • False do not throw exception if unable to find any matches for *regex* in *body*
>
> **Returns ret** : str
>
>> • The first value that matches *regex*

Notes

> • Using regex is WAY faster than ElementTree chewing the body in and out, this matters a LOT on LARGE return bodies

Session.**_extract_export_id**(*el*)
> Utility method to get the 'export_id' element from an ElementTree object

>> **Parameters el** : ElementTree object

>>> • ElementTree object to search for 'export_id' in

>> **Returns ret** : str

>>> • The text value contained in the 'export_id' element, if found

Session.**_extract_cdata_el**(*response_body*)
> Utility method to get the 'ResultXML' element from an XML body

>> **Parameters response_body** : str

>>> • XML body to search for the 'ResultXML' element in

>> **Returns ret** : str or ElementTree object

>>> • str if 'export_id' element found in XML

>>> • ElementTree object of ResultXML element otherwise

Session.**_get_response**(*request_body*, ***kwargs*)
> This is a wrapper around `pytan.sessions.Session.http_post()` for SOAP XML requests and responses.

> This method will update self.session_id if the response contains a different session_id than what is currently in this object.

>> **Parameters request_body** : str

>>> • the XML request body to send to the server

>> **connect_timeout: int, optional**

>>> • default: self.SOAP_CONNECT_TIMEOUT_SEC

>>> • timeout in seconds for connection to host

>> **response_timeout: int, optional**

>>> • default: self.SOAP_RESPONSE_TIMEOUT_SEC

>>> • timeout in seconds for response from host

>> **retry_auth: bool, optional**

>>> • default: True

>>> • True: retry authentication with username/password if session_id fails

>>> • False: throw exception if session_id fails

>> **retry_count: int, optional**

>>> • number of times to retry the request if the server fails to respond properly or in time

>> **Returns body** : str

>>> • str containing body of response from server

> **See also:**

> `pytan.sessions.Session.http_post()` wrapper method used to perform the HTTP POST

1.2.4 pytan.pollers module

Collection of classes and methods for polling of actions/questions in `pytan`

class `pytan.pollers.`**`ActionPoller`**(*handler*, *obj*, *override_timeout_secs=0*, ***kwargs*)
> Bases: `pytan.pollers.QuestionPoller`

> A class to poll the progress of an Action.

> The primary function of this class is to poll for result info for actions, and fire off events:

> ProgressChanged AnswersChanged AnswersComplete

> > **Parameters** **handler** : `pytan.handler.Handler`

> > > PyTan handler to use for GetResultInfo calls

> > **obj** : `taniumpy.object_types.action.Action`

> > > object to poll for progress

> > **polling_secs** : int, optional

> > > Number of seconds to wait in between GetResultInfo loops

> > **complete_pct** : int/float, optional

> > > Percentage of mr_tested out of estimated_total to consider the action "done"

> > **override_timeout_secs** : int, optional

> > > If supplied and not 0, timeout in seconds instead of when object expires

> `ACTION_DONE_KEY` = 'success'

> `COMPLETE_PCT` = 100

> `OBJECT_TYPE`
> > alias of `Action`

> `RUNNING_STATUSES` = ['active', 'open']

> `__module__` = 'pytan.pollers'

> `_derive_expiration`()
> > Derive the expiration datetime string from a object

> > Will generate a datetime string from self.EXPIRY_FALLBACK_SECS if unable to get the expiration from the object (self.obj) itself.

> `_derive_object_info`()
> > Derive self.object_info from self.obj

> `_derive_package_spec`()

> `_derive_result_map`()
> > A package object has to have a verify_group defined on it in order for deploy action verification to trigger. That can be only done at package creation/update

> > If verify_enable is True, then the various result states for an action change

> `_derive_status`()

> `_derive_stopped_flag`()

> `_derive_target_group`()

> `_derive_verify_enabled`()

_fix_group(*g*)
> Sets ID to null on a group object and all of it's sub_groups, needed for 6.5

_post_init()
> Post init class setup

finished_eq_passed_loop(*callbacks={}, **kwargs*)

pollerlog = <logging.Logger object at 0x103066050>

progresslog = <logging.Logger object at 0x103066110>

run(*callbacks={}, **kwargs*)
> Poll for action data and issue callbacks.
>
>> **Parameters callbacks** : dict
>>
>>> Callbacks should be a dict with any of these members: 'ProgressChanged' 'AnswersChanged' 'AnswersComplete'
>>>
>>> Each callback should be a function that accepts a poller instance and a percent complete.
>>>
>>> Any callback can choose to get data from the session by calling poller.get_result_data() or new info by calling poller.get_result_info()
>>>
>>> Any callback can choose to stop the poller by calling poller.stop()
>>>
>>> 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

seen_eq_passed_loop(*callbacks={}, **kwargs*)

class pytan.pollers.**QuestionPoller**(*handler, obj, override_timeout_secs=0, **kwargs*)
> Bases: `object`

A class to poll the progress of a Question.

The primary function of this class is to poll for result info for questions, and fire off events:

ProgressChanged AnswersChanged AnswersComplete

>> **Parameters handler** : `pytan.handler.Handler`
>>
>>> PyTan handler to use for GetResultInfo calls
>>
>> **obj** : `taniumpy.object_types.question.Question`
>>
>>> object to poll for progress
>>
>> **polling_secs** : int, optional
>>
>>> Number of seconds to wait in between GetResultInfo loops
>>
>> **complete_pct** : int/float, optional
>>
>>> Percentage of mr_tested out of estimated_total to consider the question "done"
>>
>> **override_timeout_secs** : int, optional
>>
>>> If supplied and not 0, timeout in seconds instead of when object expires

COMPLETE_PCT = 99

EXPIRY_FALLBACK_SECS = 600

OBJECT_TYPE
> alias of `Question`

POLLING_SECS = 5

STR_ATTRS = ['object_info', 'polling_secs', 'override_timeout_secs', 'complete_pct', 'expiration']

__dict__ = dict_proxy({'passed_eq_est_total_loop': <function passed_eq_est_total_loop at 0x103064320>, '__module_

__init__(*handler*, *obj*, *override_timeout_secs=0*, ***kwargs*)

__module__ = 'pytan.pollers'

__str__()

__weakref__
    list of weak references to the object (if defined)

_derive_attribute(*attr*, *fallback=''*)
    Derive an attributes value from self.obj

    Will re-fetch self.obj if the attribute is not set

        Parameters attr : string

            string of attribute name to fetch from self.obj

        fallback : string

            value to fallback to if it still can't be accessed after re-fetching the obj if fallback is
            None, an exception will be raised

        Returns val : perspective

            The value of the attr from self.obj

_derive_expiration()
    Derive the expiration datetime string from a object

    Will generate a datetime string from self.EXPIRY_FALLBACK_SECS if unable to get the expiration from
    the object (self.obj) itself.

_derive_object_info()
    Derive self.object_info from self.obj

_post_init()
    Post init class setup

_refetch_obj()
    Utility method to re-fetch a object

    This is used in the case that the obj supplied does not have all the metadata available

get_result_data(***kwargs*)
    Simple utility wrapper around handler.get_result_data()

get_result_info(***kwargs*)
    Simple utility wrapper around handler.get_result_info()

passed_eq_est_total_loop(*callbacks={}*, ***kwargs*)

pollerlog = <logging.Logger object at 0x103058e50>

progresslog = <logging.Logger object at 0x103058f10>

run(*callbacks={}*, ***kwargs*)
    Poll for question data and issue callbacks.

        Parameters callbacks : dict

Callbacks should be a dict with any of these members: 'ProgressChanged' 'AnswersChanged' 'AnswersComplete'

Each callback should be a function that accepts a poller instance and a percent complete.

Any callback can choose to get data from the session by calling poller.get_result_data() or new info by calling poller.get_result_info()

Any callback can choose to stop the poller by calling poller.stop()

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

set_complect_pct(*val*)
Set the complete_pct to a new value

> **Parameters val** : int/float
>
> > float value representing the new percentage to consider self.obj complete

stop()

class pytan.pollers.**SSEPoller**(*handler*, *export_id*, *timeout_secs=600*, ***kwargs*)
Bases: `pytan.pollers.QuestionPoller`

A class to poll the progress of a Server Side Export.

The primary function of this class is to poll for status of server side exports.

> **Parameters handler** : `pytan.handler.Handler`
>
> > PyTan handler to use for GetResultInfo calls
>
> **export_id** : str
>
> > ID of server side export
>
> **polling_secs** : int, optional
>
> > Number of seconds to wait in between status check loops (default: 2)
>
> **timeout_secs** : int, optional
>
> > timeout in seconds for waiting for status completion, 0 does not time out (default: 600)

POLLING_SECS = 2

STR_ATTRS = ['export_id', 'polling_secs', 'timeout_secs', 'sse_status']

__init__(*handler*, *export_id*, *timeout_secs=600*, ***kwargs*)

__module__ = 'pytan.pollers'

_post_init()
Post init class setup

get_sse_data()

get_sse_status()

pollerlog = <logging.Logger object at 0x1030661d0>

progresslog = <logging.Logger object at 0x103066290>

run(***kwargs*)
Poll for server side export status

sse_status_completed(***kwargs*)

class pytan.pollers.**QuestionPoller**(*handler*, *obj*, *override_timeout_secs=0*, ***kwargs*)

    Bases: `object`

A class to poll the progress of a Question.

The primary function of this class is to poll for result info for questions, and fire off events:

ProgressChanged AnswersChanged AnswersComplete

> **Parameters** **handler** : `pytan.handler.Handler`
>
> > PyTan handler to use for GetResultInfo calls
>
> **obj** : `taniumpy.object_types.question.Question`
>
> > object to poll for progress
>
> **polling_secs** : int, optional
>
> > Number of seconds to wait in between GetResultInfo loops
>
> **complete_pct** : int/float, optional
>
> > Percentage of mr_tested out of estimated_total to consider the question "done"
>
> **override_timeout_secs** : int, optional
>
> > If supplied and not 0, timeout in seconds instead of when object expires

COMPLETE_PCT = 99

EXPIRY_FALLBACK_SECS = 600

OBJECT_TYPE

    alias of `Question`

POLLING_SECS = 5

STR_ATTRS = ['object_info', 'polling_secs', 'override_timeout_secs', 'complete_pct', 'expiration']

_derive_attribute(*attr*, *fallback=''*)

    Derive an attributes value from self.obj

Will re-fetch self.obj if the attribute is not set

> **Parameters** **attr** : string
>
> > string of attribute name to fetch from self.obj
>
> **fallback** : string
>
> > value to fallback to if it still can't be accessed after re-fetching the obj if fallback is None, an exception will be raised
>
> **Returns** **val** : perspective
>
> > The value of the attr from self.obj

_derive_expiration()

    Derive the expiration datetime string from a object

Will generate a datetime string from self.EXPIRY_FALLBACK_SECS if unable to get the expiration from the object (self.obj) itself.

_derive_object_info()

    Derive self.object_info from self.obj

_post_init()

    Post init class setup

_refetch_obj()
> Utility method to re-fetch a object

> This is used in the case that the obj supplied does not have all the metadata available

get_result_data(***kwargs*)
> Simple utility wrapper around handler.get_result_data()

get_result_info(***kwargs*)
> Simple utility wrapper around handler.get_result_info()

passed_eq_est_total_loop(*callbacks={}, **kwargs*)

pollerlog = <logging.Logger object at 0x103058e50>

progresslog = <logging.Logger object at 0x103058f10>

run(*callbacks={}, **kwargs*)
> Poll for question data and issue callbacks.

>> **Parameters callbacks** : dict

>>> Callbacks should be a dict with any of these members: 'ProgressChanged' 'AnswersChanged' 'AnswersComplete'

>>> Each callback should be a function that accepts a poller instance and a percent complete.

>>> Any callback can choose to get data from the session by calling poller.get_result_data() or new info by calling poller.get_result_info()

>>> Any callback can choose to stop the poller by calling poller.stop()

>>> 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

set_complect_pct(*val*)
> Set the complete_pct to a new value

>> **Parameters val** : int/float

>>> float value representing the new percentage to consider self.obj complete

stop()

class pytan.pollers.**ActionPoller**(*handler, obj, override_timeout_secs=0, **kwargs*)
> Bases: `pytan.pollers.QuestionPoller`

> A class to poll the progress of an Action.

> The primary function of this class is to poll for result info for actions, and fire off events:

> ProgressChanged AnswersChanged AnswersComplete

>> **Parameters handler** : `pytan.handler.Handler`

>>> PyTan handler to use for GetResultInfo calls

>> **obj** : `taniumpy.object_types.action.Action`

>>> object to poll for progress

>> **polling_secs** : int, optional

>>> Number of seconds to wait in between GetResultInfo loops

>> **complete_pct** : int/float, optional

Percentage of mr_tested out of estimated_total to consider the action "done"

override_timeout_secs : int, optional

If supplied and not 0, timeout in seconds instead of when object expires

ACTION_DONE_KEY = 'success'

COMPLETE_PCT = 100

OBJECT_TYPE
  alias of `Action`

RUNNING_STATUSES = ['active', 'open']

_derive_expiration()
  Derive the expiration datetime string from a object

  Will generate a datetime string from self.EXPIRY_FALLBACK_SECS if unable to get the expiration from the object (self.obj) itself.

_derive_object_info()
  Derive self.object_info from self.obj

_derive_package_spec()

_derive_result_map()
  A package object has to have a verify_group defined on it in order for deploy action verification to trigger. That can be only done at package creation/update

  If verify_enable is True, then the various result states for an action change

_derive_status()

_derive_stopped_flag()

_derive_target_group()

_derive_verify_enabled()

_fix_group(*g*)
  Sets ID to null on a group object and all of it's sub_groups, needed for 6.5

_post_init()
  Post init class setup

finished_eq_passed_loop(*callbacks={}, **kwargs*)

pollerlog = <logging.Logger object at 0x103066050>

progresslog = <logging.Logger object at 0x103066110>

run(*callbacks={}, **kwargs*)
  Poll for action data and issue callbacks.

  Parameters callbacks : dict

  Callbacks should be a dict with any of these members: 'ProgressChanged' 'AnswersChanged' 'AnswersComplete'

  Each callback should be a function that accepts a poller instance and a percent complete.

  Any callback can choose to get data from the session by calling poller.get_result_data() or new info by calling poller.get_result_info()

  Any callback can choose to stop the poller by calling poller.stop()

> > 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

> **seen_eq_passed_loop**(*callbacks={}, **kwargs*)

1.2.5 pytan.constants module

PyTan Constants

This contains a number of constants that drive PyTan.

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 argumen**

> - 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, {'stats': 'DEBUG'}, 'Sets all loggers to only output at WARNING or above exc

> **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
> - str, description of 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': {'ha
> 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

---

`pytan.constants.`**`SSE_CRASH_MAP`** `= [{'major': 6, 'build': 4300, 'minor': 5, 'revision': 314}]`
    Mapping of versions to watch out for crashes/handle bugs for server side export

`pytan.constants.`**`SSE_FORMAT_MAP`** `= [('csv', '0', 0), ('xml', '1', 1), ('xml_obj', '1', 1), ('cef', '2', 2)]`
    Mapping of human friendly strings to API integers for server side export

`pytan.constants.`**`SSE_RESTRICT_MAP`** `= {1: [{'major': 6, 'build': 4300, 'minor': 5, 'revision': 314}], 2: [{'major': 6, 'bui`
    Mapping of API integers for server side export format to version support

`pytan.constants.`**`TIME_FORMAT`** `= '%Y-%m-%dT%H:%M:%S'`
    Tanium's format for date time strings

1.2.6 pytan.utils module

Collection of exceptions, classes, and methods used throughout `pytan`

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='all'*)
    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**(*gmt_tz=True*)
    Creates a console logging handler using `SplitStreamHandler`

pytan.utils.**spew**(*t*)
    Prints a string based on DEBUG_OUTPUT bool

> **Parameters t** : str

> - string to debug print

pytan.utils.**print_log_levels**()
    Prints info about each loglevel from `pytan.constants.LOG_LEVEL_MAPS`

pytan.utils.**get_all_loggers**()
    Gets all loggers currently known to pythons logging system that exist in `pytan.constants.LOG_LEVEL_MAPS`

pytan.utils.**log_session_communication**(*h*)
    Uses `xml_pretty()` to pretty print the last request and response bodies from the session object in h to the logging system

> **Parameters h** : Handler object

> - Handler object with session object containing last request and response body

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_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.**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.**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 pytan.exceptions.HandlerError** : `pytan.exceptions.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 VersionMismatchError** : `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*

pytan.utils.**get_percentage**(*part*, *whole*)

> Utility method for getting percentage of part out of whole
>
> > **Parameters part: int, float**
> >
> > > **whole: int, float**
> >
> > **Returns int** : the percentage of part out of whole

pytan.utils.**calc_percent**(*percent*, *whole*)

> Utility method for getting percentage of whole
>
> > **Parameters percent: int, float**
> >
> > > **whole: int, float**
> >
> > **Returns int** : the percentage of whole

Utility Functions: Time and Date handling

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.**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.**timestr_to_datetime**(*timestr*)
    Get a datetime.datetime object for *timestr*

>   **Parameters** **timestr** : str

>>   • date & time in taniums format

>   **Returns** datetime.datetime

>>   • the datetime object for the timestr

pytan.utils.**datetime_to_timestr**(*dt*)
    Get a timestr for *dt*

>   **Parameters** **dt** : datetime.datetime

>>   • datetime object

>   **Returns** timestr: str

>>   • the timestr for *dt* in taniums format

pytan.utils.**func_timing**(*f*)
    Decorator to add timing information around a function

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=True*)
>   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 `pytan.exceptions.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_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

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.**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

pytan.utils.**load_param_json_file**(*parameters_json_file*)
> Opens a json file and sanity checks it for use as a parameters element for a taniumpy object

> > **Parameters parameters_json_file** : str

> > > • path to JSON file that describes an API object

> > **Returns** obj

> > > • contents of parameters_json_file de-serialized

pytan.utils.**shrink_obj**(*obj*, *attrs=None*)
> Returns a new class of obj with only id/name/hash defined

> > **Parameters obj** : `taniumpy.object_types.base.BaseType`

> > > • Object to shrink

> > **attrs** : list of str

> > > • default: None

> > > • list of attribute str's to copy over to new object, will default to ['name', 'id', 'hash'] if None

> > **Returns new_obj** : `taniumpy.object_types.base.BaseType`

> > > • Shrunken object

pytan.utils.**get_taniumpy_obj**(*obj_map*)
> Gets a taniumpy object from *obj_map*

> **Parameters obj_map** : str
>
> > • str of taniumpy object to fetch
>
> **Returns obj** : `taniumpy.object_types.base.BaseType`
>
> > • matching taniumpy object for *obj_map*

pytan.utils.**plugin_zip**(*p*)

> Maps columns to values for each row in a plugins sql_response and returns a list of dicts
>
> > **Parameters p** : `taniumpy.object_types.plugin.Plugin`
> >
> > > • plugin object
> >
> > **Returns** dict
> >
> > > • the columns and result_rows of the sql_response in Plugin object zipped up into a dictionary

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 `pytan.exceptions.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 `pytan.exceptions.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.**copy_obj**(*obj*, *skip_attrs=None*)
> Returns a new class of obj with with out any attributes in skip_attrs specified

> > **Parameters obj** : `taniumpy.object_types.base.BaseType`

> > > • Object to copy

> > **skip_attrs** : list of str

- default: None

- list of attribute str's to skip copying over to new object, will default to [] if None

Returns new_obj : `taniumpy.object_types.base.BaseType`

- Copied object with attributes in skip_attrs skipped

`pytan.utils.`**`copy_package_obj_for_action`**(*obj*, *skip_attrs=None*)
Returns a new class of package obj with with out any attributes in skip_attrs specified

Parameters obj : `taniumpy.object_types.base.BaseType`

- Object to copy

skip_attrs : list of str

- default: None

- list of attribute str's to skip copying over to new object, default if None: ['id', 'deleted_flag', 'available_time', 'creation_time', 'modification_time', 'source_id']

Returns new_obj : `taniumpy.object_types.base.BaseType`

- Copied object with attributes in skip_attrs skipped

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 `pytan.exceptions.DefinitionParserError` if not

`pytan.utils.`**`parse_defs`**(*defname*, *deftypes*, *strconv=None*, *empty_ok=True*, *defs=None*, ***kwargs*)
Parses and validates defs into new_defs

    Parameters defname : str

- Name of definition

    deftypes : list of str

- list of valid types that defs can be

    strconv : str

- if supplied, and defs is a str, turn defs into a dict with key = strconv, value = defs

    empty_ok : bool

- True: defs is allowed to be empty

- False: defs is not allowed to be empty

    Returns new_defs : list of dict

- parsed and validated defs

`pytan.utils.`**`val_package_def`**(*package_def*)
Validates package definitions

Ensures package definition has a selector, and if a package definition has a params key, that key is valid

    Parameters package_def : dict

- package definition

`pytan.utils.`**`val_q_filter_defs`**(*q_filter_defs*)
Validates question filter definitions

Ensures each question filter definition has a selector, and if a question filter definition has a filter key, that key is valid

    Parameters q_filter_defs : list of dict

- list of question filter definitions

`pytan.utils.`**`val_sensor_defs`**(*sensor_defs*)
Validates sensor definitions

Ensures each sensor definition has a selector, and if a sensor definition has a params, options, or filter key, that each key is valid

    Parameters sensor_defs : list of dict

- list of sensor definitions

1.2.7 pytan.xml_clean module

This is a regex based XML cleaner that will replace unsupported characters

`pytan.xml_clean.`**`DEFAULT_REPLACEMENT = u'\ufffd'`**
The default character to use when replacing characters

`pytan.xml_clean.`**`INVALID_UNICODE_RAW_RE`**` = u'[^\t\n\r -\ud7ff\ue000-\ufffd]'`
   The raw regex string to use when replacing invalid characters

`pytan.xml_clean.`**`INVALID_UNICODE_RE`**` = <_sre.SRE_Pattern object at 0x7fbd986ccb00>`
   The regex object to use when replacing invalid characters

`pytan.xml_clean.`**`RESTRICTED_UNICODE_RAW_RE`**` = u'[\x7f-\x84\x86-\x9f\ufdd0-\ufdef]'`
   The raw regex string to use when replacing restricted characters

`pytan.xml_clean.`**`RESTRICTED_UNICODE_RE`**` = <_sre.SRE_Pattern object at 0x7fbd98456960>`
   The regex object to use when replacing restricted characters

`pytan.xml_clean.`**`XML_1_0_RESTRICTED_HEX`**` = [[127, 132], [134, 159], [64976, 65007]]`

   Restricted/discouraged Unicode characters for XML documents: [#x7F-#x84], [#x86-#x9F], [#xFDD0-#xFDEF], [#x1FFFE-#x1FFFF], [#x2FFFE-#x2FFFF], [#x3FFFE-#x3FFFF], [#x4FFFE-#x4FFFF], [#x5FFFE-#x5FFFF], [#x6FFFE-#x6FFFF], [#x7FFFE-#x7FFFF], [#x8FFFE-#x8FFFF], [#x9FFFE-#x9FFFF], [#xAFFFE-#xAFFFF], [#xBFFFE-#xBFFFF], [#xCFFFE-#xCFFFF], [#xDFFFE-#xDFFFF], [#xEFFFE-#xEFFFF], [#xFFFFE-#xFFFFF], [#x10FFFE-#x10FFFF]

   Source: http://www.w3.org/TR/REC-xml/#NT-Char

`pytan.xml_clean.`**`XML_1_0_VALID_HEX`**` = [[9], [10], [13], [32, 55295], [57344, 65533]]`

   Valid Unicode characters for XML documents: (any Unicode character, excluding the surrogate blocks, FFFE, and FFFF) #x9, #xA, #xD, [#x20-#xD7FF], [#xE000-#xFFFD], [#x10000-#x10FFFF]

   Source: http://www.w3.org/TR/REC-xml/#NT-Char

`pytan.xml_clean.`**`replace_invalid_unicode`**(*text*, *replacement=None*)
   Replaces invalid unicode characters with *replacement*

   Parameters text : str

      • str to clean

   replacement : str, optional

      • default: None

      • if invalid characters found, they will be replaced with this

      • if not supplied, will default to DEFAULT_REPLACEMENT

   Returns str, cnt, RE : tuple

      • str : the cleaned version of *text*

      • cnt : the number of replacements that took place

      • RE : the regex object that was used to do the replacements

`pytan.xml_clean.`**`replace_restricted_unicode`**(*text*, *replacement=None*)
   Replaces restricted unicode characters with *replacement*

   Parameters text : str

      • str to clean

   replacement : str, optional

      • default: None

      • if restricted characters found, they will be replaced with this

      • if not supplied, will default to DEFAULT_REPLACEMENT

   Returns str, cnt, RE : tuple

- str : the cleaned version of *text*

- cnt : the number of replacements that took place

- RE : the regex object that was used to do the replacements

pytan.xml_clean.**xml_cleaner**(*s*, *encoding='utf-8'*, *clean_restricted=True*, *log_messages=True*, *show_bad_characters=False*, ***kwargs*)

Removes invalid /restricted characters per XML 1.0 spec

> **Parameters s** : str
>
> > - str to clean
> >
> > **encoding** : str, optional
> >
> > > - default: 'utf-8'
> > >
> > > - encoding of *s*
> > >
> > > **clean_restricted** : bool, optional
> > >
> > > > - default: True
> > > >
> > > > - remove restricted characters from *s* or not
> > > >
> > > > **log_messages** : bool, optional
> > > >
> > > > > - default: True
> > > > >
> > > > > - log messages using python logging or not
> > > > >
> > > > > **show_bad_characters** : bool, optional
> > > > >
> > > > > > - default: False
> > > > > >
> > > > > > - log bad character matches or not
>
> **Returns** str
>
> > - the cleaned version of *s*

1.2.8 pytan Unit Tests

This contains unit tests for pytan.

These unit tests do not require a connection to a Tanium server in order to run.

class test_pytan_unit.**TestDehumanizeExtractionUtils**(*methodName='runTest'*)

Bases: unittest.case.TestCase

__module__ = 'test_pytan_unit'

test_extract_filter_invalid()

test_extract_filter_nofilter()

test_extract_filter_valid()

test_extract_filter_valid_all()

test_extract_options_invalid_option()

test_extract_options_many()

test_extract_options_missing_value_max_data_age()

test_extract_options_missing_value_value_type()

test_extract_options_nooptions()

test_extract_options_single()

test_extract_params()

test_extract_params_missing_seperator()

test_extract_params_multiparams()

test_extract_params_noparams()

test_extract_selector()

test_extract_selector_use_name_if_noselector()

class test_pytan_unit.**TestDehumanizeQuestionFilterUtils**(*methodName='runTest'*)
Bases: unittest.case.TestCase

__module__ = 'test_pytan_unit'

test_empty_filterlist()

test_empty_filterstr()

test_invalid_filter1()

test_invalid_filter2()

test_invalid_filter3()

test_multi_filter_list()

test_single_filter_list()

test_single_filter_str()

class test_pytan_unit.**TestDehumanizeQuestionOptionUtils**(*methodName='runTest'*)
Bases: unittest.case.TestCase

__module__ = 'test_pytan_unit'

test_empty_optionlist()

test_empty_optionstr()

test_invalid_option1()

test_invalid_option2()

test_option_list_many()

test_option_list_multi()

test_option_list_single()

test_option_str()

class test_pytan_unit.**TestDehumanizeSensorUtils**(*methodName='runTest'*)
Bases: unittest.case.TestCase

__module__ = 'test_pytan_unit'

test_empty_args_dict()

test_empty_args_list()

test_empty_args_str()

test_multi_list_complex()

test_single_str()

test_single_str_complex1()

test_single_str_complex2()

test_single_str_with_filter()

test_valid_simple_list()

test_valid_simple_str_hash_selector()

test_valid_simple_str_id_selector()

test_valid_simple_str_name_selector()

class `test_pytan_unit`.**TestDeserializeBadXML**(*methodName='runTest'*)

    Bases: `unittest.case.TestCase`

    __module__ = 'test_pytan_unit'

    test_bad_chars_basetype_control()

        This XML file has a number of control characters that are not valid in XML.

        This test validates that pytan.xml_clean.xml_cleaner() will remove all the invalid and restricted characters, which should allow the body to be parsed properly.

    test_bad_chars_resultset_latin1()

        This XML file has some characters that are actually encoded as latin1 (as well as some restricted characters).

        This test validates that pytan.xml_clean.xml_cleaner() will properly fall back to latin1 for decoding the docuemnt, as well as remove all the invalid and restricted characters, which should allow the body to be parsed properly.

    test_bad_chars_resultset_surrogate()

        This XML file has some characters that are unpaired surrogates in unicode. Surrogates (unpaired or otherwise) are not legal XML characters.

        This test validates that pytan.xml_clean.xml_cleaner() will properly remove all the invalid and restricted characters, which should allow the body to be parsed properly.

class `test_pytan_unit`.**TestGenericUtils**(*methodName='runTest'*)

    Bases: `unittest.case.TestCase`

    __module__ = 'test_pytan_unit'

    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_load_param_file_invalid_file()

test_load_param_file_invalid_json()

test_load_param_file_valid()

test_load_taniumpy_file_invalid_file()

test_load_taniumpy_file_invalid_json()

test_version_higher()

test_version_lower()

class test_pytan_unit.**TestManualBuildObjectUtils**(*methodName='runTest'*)
    Bases: unittest.case.TestCase

    __module__ = 'test_pytan_unit'

    classmethod **setUpClass**()

    test_build_group_obj()

    test_build_manual_q()

    test_build_selectlist_obj_invalid_filter()

    test_build_selectlist_obj_missing_value()

    test_build_selectlist_obj_noparamssensorobj_noparams()
        builds a selectlist object using a sensor obj with no params

    test_build_selectlist_obj_noparamssensorobj_withparams()
        builds a selectlist object using a sensor obj with no params, but passing in params (which should be added as of 1.0.4)

    test_build_selectlist_obj_withparamssensorobj_noparams()
        builds a selectlist object using a sensor obj with 4 params but not supplying any values for any of the params

    test_build_selectlist_obj_withparamssensorobj_withparams()
        builds a selectlist object using a sensor obj with 4 params but supplying a value for only one param

class test_pytan_unit.**TestManualPackageDefValidateUtils**(*methodName='runTest'*)
    Bases: unittest.case.TestCase

    __module__ = 'test_pytan_unit'

    test_invalid1()

    test_invalid2()

    test_valid1()

    test_valid2()

class test_pytan_unit.**TestManualQuestionFilterDefParseUtils**(*methodName='runTest'*)
    Bases: unittest.case.TestCase

    __module__ = 'test_pytan_unit'

    test_parse_emptydict()

---

test_parse_emptylist()

test_parse_emptystr()

test_parse_multi_filter()

test_parse_noargs()

test_parse_none()

test_parse_single_filter()

test_parse_str()

class test_pytan_unit.**TestManualQuestionFilterDefValidateUtils**(*methodName='runTest'*)
  Bases: unittest.case.TestCase

  __module__ = 'test_pytan_unit'

  test_invalid1()

  test_valid1()

  test_valid2()

class test_pytan_unit.**TestManualQuestionOptionDefParseUtils**(*methodName='runTest'*)
  Bases: unittest.case.TestCase

  __module__ = 'test_pytan_unit'

  test_parse_emptydict()

  test_parse_emptylist()

  test_parse_emptystr()

  test_parse_list()

  test_parse_noargs()

  test_parse_none()

  test_parse_options_dict()

  test_parse_str()

class test_pytan_unit.**TestManualSensorDefParseUtils**(*methodName='runTest'*)
  Bases: unittest.case.TestCase

  __module__ = 'test_pytan_unit'

  test_parse_complex()
    list with many items is parsed into same list

  test_parse_dict_hash()
    dict with hash is parsed into list of same dict

  test_parse_dict_id()
    dict with id is parsed into list of same dict

  test_parse_dict_name()
    dict with name is parsed into list of same dict

  test_parse_emptydict()
    args=={} throws exception

  test_parse_emptylist()
    args==[] throws exception

test_parse_emptystr()
: args=='' throws exception

test_parse_noargs()
: no args throws exception

test_parse_none()
: args==None throws exception

test_parse_str1()
: simple str is parsed into list of same str

class test_pytan_unit.**TestManualSensorDefValidateUtils**(*methodName='runTest'*)
: Bases: unittest.case.TestCase

    __module__ = 'test_pytan_unit'

    test_invalid1()

    test_invalid2()

    test_invalid3()

    test_invalid4()

    test_valid1()

    test_valid2()

    test_valid3()

    test_valid4()

1.2.9 pytan Functional Tests

This contains valid 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_valid_server_tests.**ValidServerTests**(*methodName='runTest'*)
: Bases: unittest.case.TestCase

    __module__ = 'test_pytan_valid_server_tests'

    classmethod setUpClass()

    setup_test()

    classmethod tearDownClass()

    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_question_sensor_help()

test_invalid_question_2_invalid_ask_manual_question_parameter_split()

test_invalid_question_3_invalid_ask_manual_question_filter_help()

test_invalid_question_4_invalid_ask_manual_question_option()

test_invalid_question_5_invalid_ask_manual_question_sensor()

test_invalid_question_6_invalid_ask_manual_question_option_help()

test_invalid_question_7_invalid_ask_manual_question_paramater_too_many()

test_invalid_question_8_invalid_ask_manual_question_filter()

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_question_sensor_with_filter()

test_valid_question_11_ask_manual_question_multiple_sensors_identified_by_name()

test_valid_question_12_ask_manual_question_sensor_with_parameters_and_filter_and_option

test_valid_question_13_ask_manual_question_sensor_with_filter_and_3_options()

test_valid_question_14_ask_manual_question_complex_query2()

test_valid_question_15_ask_manual_question_complex_query1()

test_valid_question_1_ask_manual_question_sensor_with_parameters_and_some_supplied_par

test_valid_question_2_ask_manual_question_multiple_sensors_with_parameters_and_some_su

> > **test_valid_question_3_ask_manual_question_simple_multiple_sensors**()
>
> > **test_valid_question_4_ask_manual_question_sensor_without_parameters_and_supplied_param**
>
> > **test_valid_question_5_ask_manual_question_sensor_with_filter_and_2_options**()
>
> > **test_valid_question_6_ask_manual_question_sensor_with_parameters_and_filter**()
>
> > **test_valid_question_7__ask_manual_question_sensor_complex**()
>
> > **test_valid_question_8_ask_manual_question_sensor_with_parameters_and_no_supplied_param**
>
> > **test_valid_question_9_ask_manual_question_simple_single_sensor**()
>
> > **test_valid_saved_question_1_ask_saved_question_refresh_data**()
>
> > **test_valid_saved_question_2_ask_saved_question_by_name**()
>
> > **test_valid_saved_question_3_ask_saved_question_by_name_in_list**()

test_pytan_valid_server_tests.**chew_csv**(*c*)

test_pytan_valid_server_tests.**spew**(*m*, *l=3*)

This contains invalid 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_invalid_server_tests.**InvalidServerTests**(*methodName='runTest'*)

> Bases: unittest.case.TestCase
>
> **__module__** = 'test_pytan_invalid_server_tests'
>
> 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**()

test_pytan_invalid_server_tests.**spew**(*m*, *l=3*)

1.2.10 PyTan API examples

Pytan api basic handler example

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

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

Example Python Code

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

```
6   my_file = os.path.abspath(sys.argv[0])
7   my_dir = os.path.dirname(my_file)
8
9   # determine the pytan lib dir and add it to the path
10  parent_dir = os.path.dirname(my_dir)
11  pytan_root_dir = os.path.dirname(parent_dir)
12  lib_dir = os.path.join(pytan_root_dir, 'lib')
13  path_adds = [lib_dir]
14
15  for aa in path_adds:
16      if aa not in sys.path:
17          sys.path.append(aa)
18
19
20  # connection info for Tanium Server
21  USERNAME = "Tanium User"
22  PASSWORD = "T@n!um"
23  HOST = "172.16.31.128"
24  PORT = "443"
25
26  # Logging conrols
27  LOGLEVEL = 2
28  DEBUGFORMAT = False
29
30  import tempfile
31
32  import pytan
33  handler = pytan.Handler(
34      username=USERNAME,
35      password=PASSWORD,
36      host=HOST,
37      port=PORT,
38      loglevel=LOGLEVEL,
39      debugformat=DEBUGFORMAT,
40  )
41
42  print handler
```

Output from Python Code

```
1   Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
```

PyTan API Valid Saved Question Examples

Ask saved question refresh data

Ask a saved question and refresh the data for the saved question (asks a new question)

Example Python Code

```
1   import os
2   import sys
```

```
3   sys.dont_write_bytecode = True
4
5   # Determine our script name, script dir
6   my_file = os.path.abspath(sys.argv[0])
7   my_dir = os.path.dirname(my_file)
8
9   # determine the pytan lib dir and add it to the path
10  parent_dir = os.path.dirname(my_dir)
11  pytan_root_dir = os.path.dirname(parent_dir)
12  lib_dir = os.path.join(pytan_root_dir, 'lib')
13  path_adds = [lib_dir]
14
15  for aa in path_adds:
16      if aa not in sys.path:
17          sys.path.append(aa)
18
19
20  # connection info for Tanium Server
21  USERNAME = "Tanium User"
22  PASSWORD = "T@n!um"
23  HOST = "172.16.31.128"
24  PORT = "443"
25
26  # Logging conrols
27  LOGLEVEL = 2
28  DEBUGFORMAT = False
29
30  import tempfile
31
32  import pytan
33  handler = pytan.Handler(
34      username=USERNAME,
35      password=PASSWORD,
36      host=HOST,
37      port=PORT,
38      loglevel=LOGLEVEL,
39      debugformat=DEBUGFORMAT,
40  )
41
42  print handler
43
44  # setup the arguments for the handler method
45  kwargs = {}
46  kwargs["refresh_data"] = True
47  kwargs["qtype"] = u'saved'
48  kwargs["name"] = u'Installed Applications'
49
50  # call the handler with the ask method, passing in kwargs for arguments
51  response = handler.ask(**kwargs)
52  import pprint, io
53
54  print ""
55  print "Type of response: ", type(response)
56
57  print ""
58  print "Pretty print of response:"
59  print pprint.pformat(response)
60
```

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

Output from Python Code

```
1   Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2   2015-08-07 19:37:49,991 DEBUG     pytan.handler.QuestionPoller: ID 1279: id resolved to 1279
3   2015-08-07 19:37:49,991 DEBUG     pytan.handler.QuestionPoller: ID 1279: expiration resolved to 2015-
4   2015-08-07 19:37:49,991 DEBUG     pytan.handler.QuestionPoller: ID 1279: query_text resolved to Get n
5   2015-08-07 19:37:49,991 DEBUG     pytan.handler.QuestionPoller: ID 1279: id resolved to 1279
6   2015-08-07 19:37:49,991 DEBUG     pytan.handler.QuestionPoller: ID 1279: Object Info resolved to Ques
7   2015-08-07 19:37:49,997 DEBUG     pytan.handler.QuestionPoller: ID 1279: Progress: Tested: 0, Passed:
8   2015-08-07 19:37:49,997 DEBUG     pytan.handler.QuestionPoller: ID 1279: Timing: Started: 2015-08-07
9   2015-08-07 19:37:49,997 INFO      pytan.handler.QuestionPoller: ID 1279: Progress Changed 0% (0 of 2)
10  2015-08-07 19:37:55,007 DEBUG     pytan.handler.QuestionPoller: ID 1279: Progress: Tested: 2, Passed:
11  2015-08-07 19:37:55,008 DEBUG     pytan.handler.QuestionPoller: ID 1279: Timing: Started: 2015-08-07
12  2015-08-07 19:37:55,008 INFO      pytan.handler.QuestionPoller: ID 1279: Progress Changed 100% (2 of
13  2015-08-07 19:37:55,008 INFO      pytan.handler.QuestionPoller: ID 1279: Reached Threshold of 99% (2
14
15  Type of response:  <type 'dict'>
16
17  Pretty print of response:
18  {'poller_object': <pytan.pollers.QuestionPoller object at 0x10a6c0410>,
19   'poller_success': True,
20   'question_object': <taniumpy.object_types.question.Question object at 0x10a7e7f50>,
21   'question_results': <taniumpy.object_types.result_set.ResultSet object at 0x10a808190>,
22   'saved_question_object': <taniumpy.object_types.saved_question.SavedQuestion object at 0x10a7ecb90>
23
24  Equivalent Question if it were to be asked in the Tanium Console:
25  Get number of machines
26
27  CSV Results of response:
28  Name,Silent Uninstall String,Uninstallable,Version
29  Image Capture Extension,nothing,Not Uninstallable,10.2
30  Dictation,nothing,Not Uninstallable,1.6.1
31  Wish,nothing,Not Uninstallable,8.5.9
32  Uninstall AnyConnect,nothing,Not Uninstallable,3.1.08009
33  Time Machine,nothing,Not Uninstallable,1.3
34  AppleGraphicsWarning,nothing,Not Uninstallable,2.3.0
35  soagent,nothing,Not Uninstallable,7.0
36  Feedback Assistant,nothing,Not Uninstallable,4.1.3
```

```
37  AinuIM,nothing,Not Uninstallable,1.0
38  vpndownloader,nothing,Not Uninstallable,3.1.08009
39  Pass Viewer,nothing,Not Uninstallable,1.0
40  ARDAgent,nothing,Not Uninstallable,3.8.4
41  OBEXAgent,nothing,Not Uninstallable,4.3.5
42  PressAndHold,nothing,Not Uninstallable,1.2
43  ..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

```python
1   import os
2   import sys
3   sys.dont_write_bytecode = True
4
5   # Determine our script name, script dir
6   my_file = os.path.abspath(sys.argv[0])
7   my_dir = os.path.dirname(my_file)
8
9   # determine the pytan lib dir and add it to the path
10  parent_dir = os.path.dirname(my_dir)
11  pytan_root_dir = os.path.dirname(parent_dir)
12  lib_dir = os.path.join(pytan_root_dir, 'lib')
13  path_adds = [lib_dir]
14
15  for aa in path_adds:
16      if aa not in sys.path:
17          sys.path.append(aa)
18
19
20  # connection info for Tanium Server
21  USERNAME = "Tanium User"
22  PASSWORD = "T@n!um"
23  HOST = "172.16.31.128"
24  PORT = "443"
25
26  # Logging conrols
27  LOGLEVEL = 2
28  DEBUGFORMAT = False
29
30  import tempfile
31
32  import pytan
33  handler = pytan.Handler(
34      username=USERNAME,
35      password=PASSWORD,
36      host=HOST,
37      port=PORT,
38      loglevel=LOGLEVEL,
39      debugformat=DEBUGFORMAT,
40  )
41
```

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

Output from Python Code

```
1   Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2
3   Type of response:  <type 'dict'>
4
5   Pretty print of response:
6   {'poller_object': None,
7    'poller_success': None,
8    'question_object': <taniumpy.object_types.question.Question object at 0x10a80a150>,
9    'question_results': <taniumpy.object_types.result_set.ResultSet object at 0x10a810c10>,
10   'saved_question_object': <taniumpy.object_types.saved_question.SavedQuestion object at 0x10a810a50>
11
12  Equivalent Question if it were to be asked in the Tanium Console:
13  Get Installed Applications from all machines
14
15  CSV Results of response:
16  Name,Silent Uninstall String,Uninstallable,Version
17  Image Capture Extension,nothing,Not Uninstallable,10.2
18  Dictation,nothing,Not Uninstallable,1.6.1
```

```
19  Wish,nothing,Not Uninstallable,8.5.9
20  Uninstall AnyConnect,nothing,Not Uninstallable,3.1.08009
21  Time Machine,nothing,Not Uninstallable,1.3
22  AppleGraphicsWarning,nothing,Not Uninstallable,2.3.0
23  soagent,nothing,Not Uninstallable,7.0
24  Feedback Assistant,nothing,Not Uninstallable,4.1.3
25  AinuIM,nothing,Not Uninstallable,1.0
26  vpndownloader,nothing,Not Uninstallable,3.1.08009
27  Pass Viewer,nothing,Not Uninstallable,1.0
28  ARDAgent,nothing,Not Uninstallable,3.8.4
29  OBEXAgent,nothing,Not Uninstallable,4.3.5
30  PressAndHold,nothing,Not Uninstallable,1.2
31  ..trimmed for brevity..
```

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
1   import os
2   import sys
3   sys.dont_write_bytecode = True
4
5   # Determine our script name, script dir
6   my_file = os.path.abspath(sys.argv[0])
7   my_dir = os.path.dirname(my_file)
8
9   # determine the pytan lib dir and add it to the path
10  parent_dir = os.path.dirname(my_dir)
11  pytan_root_dir = os.path.dirname(parent_dir)
12  lib_dir = os.path.join(pytan_root_dir, 'lib')
13  path_adds = [lib_dir]
14
15  for aa in path_adds:
16      if aa not in sys.path:
17          sys.path.append(aa)
18
19
20  # connection info for Tanium Server
21  USERNAME = "Tanium User"
22  PASSWORD = "T@n!um"
23  HOST = "172.16.31.128"
24  PORT = "443"
25
26  # Logging conrols
27  LOGLEVEL = 2
28  DEBUGFORMAT = False
29
30  import tempfile
31
32  import pytan
33  handler = pytan.Handler(
34      username=USERNAME,
35      password=PASSWORD,
```

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

Output from Python Code

```
1     Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2
3     Type of response:  <type 'dict'>
4
5     Pretty print of response:
6     {'poller_object': None,
7      'poller_success': None,
8      'question_object': <taniumpy.object_types.question.Question object at 0x10a613d90>,
9      'question_results': <taniumpy.object_types.result_set.ResultSet object at 0x10a6c0410>,
10     'saved_question_object': <taniumpy.object_types.saved_question.SavedQuestion object at 0x10a808b10>
11
12    Equivalent Question if it were to be asked in the Tanium Console:
```

```
13   Get Installed Applications from all machines
14
15   CSV Results of response:
16   Name,Silent Uninstall String,Uninstallable,Version
17   Image Capture Extension,nothing,Not Uninstallable,10.2
18   Dictation,nothing,Not Uninstallable,1.6.1
19   Wish,nothing,Not Uninstallable,8.5.9
20   Uninstall AnyConnect,nothing,Not Uninstallable,3.1.08009
21   Time Machine,nothing,Not Uninstallable,1.3
22   AppleGraphicsWarning,nothing,Not Uninstallable,2.3.0
23   soagent,nothing,Not Uninstallable,7.0
24   Feedback Assistant,nothing,Not Uninstallable,4.1.3
25   AinuIM,nothing,Not Uninstallable,1.0
26   vpndownloader,nothing,Not Uninstallable,3.1.08009
27   Pass Viewer,nothing,Not Uninstallable,1.0
28   ARDAgent,nothing,Not Uninstallable,3.8.4
29   OBEXAgent,nothing,Not Uninstallable,4.3.5
30   PressAndHold,nothing,Not Uninstallable,1.2
31   ..trimmed for brevity..
```

PyTan API Valid Question Examples

Ask manual 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

```python
1    import os
2    import sys
3    sys.dont_write_bytecode = True
4
5    # Determine our script name, script dir
6    my_file = os.path.abspath(sys.argv[0])
7    my_dir = os.path.dirname(my_file)
8
9    # determine the pytan lib dir and add it to the path
10   parent_dir = os.path.dirname(my_dir)
11   pytan_root_dir = os.path.dirname(parent_dir)
12   lib_dir = os.path.join(pytan_root_dir, 'lib')
13   path_adds = [lib_dir]
14
15   for aa in path_adds:
16       if aa not in sys.path:
17           sys.path.append(aa)
18
19
20   # connection info for Tanium Server
21   USERNAME = "Tanium User"
22   PASSWORD = "T@n!um"
23   HOST = "172.16.31.128"
24   PORT = "443"
25
```

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

Output from Python Code

```
1   Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2   2015-08-07 19:37:55,255 DEBUG    pytan.handler.QuestionPoller: ID 1280: id resolved to 1280
```

```
3   2015-08-07 19:37:55,255 DEBUG     pytan.handler.QuestionPoller: ID 1280: expiration resolved to 2015-
4   2015-08-07 19:37:55,255 DEBUG     pytan.handler.QuestionPoller: ID 1280: query_text resolved to Get C
5   2015-08-07 19:37:55,255 DEBUG     pytan.handler.QuestionPoller: ID 1280: id resolved to 1280
6   2015-08-07 19:37:55,255 DEBUG     pytan.handler.QuestionPoller: ID 1280: Object Info resolved to Ques
7   2015-08-07 19:37:55,260 DEBUG     pytan.handler.QuestionPoller: ID 1280: Progress: Tested: 0, Passed:
8   2015-08-07 19:37:55,260 DEBUG     pytan.handler.QuestionPoller: ID 1280: Timing: Started: 2015-08-07
9   2015-08-07 19:37:55,260 INFO      pytan.handler.QuestionPoller: ID 1280: Progress Changed 0% (0 of 2)
10  2015-08-07 19:38:00,263 DEBUG     pytan.handler.QuestionPoller: ID 1280: Progress: Tested: 1, Passed:
11  2015-08-07 19:38:00,263 DEBUG     pytan.handler.QuestionPoller: ID 1280: Timing: Started: 2015-08-07
12  2015-08-07 19:38:00,264 INFO      pytan.handler.QuestionPoller: ID 1280: Progress Changed 50% (1 of 2
13  2015-08-07 19:38:05,271 DEBUG     pytan.handler.QuestionPoller: ID 1280: Progress: Tested: 1, Passed:
14  2015-08-07 19:38:05,271 DEBUG     pytan.handler.QuestionPoller: ID 1280: Timing: Started: 2015-08-07
15  2015-08-07 19:38:10,277 DEBUG     pytan.handler.QuestionPoller: ID 1280: Progress: Tested: 2, Passed:
16  2015-08-07 19:38:10,277 DEBUG     pytan.handler.QuestionPoller: ID 1280: Timing: Started: 2015-08-07
17  2015-08-07 19:38:10,277 INFO      pytan.handler.QuestionPoller: ID 1280: Progress Changed 100% (2 of
18  2015-08-07 19:38:10,277 INFO      pytan.handler.QuestionPoller: ID 1280: Reached Threshold of 99% (2
19
20  Type of response:  <type 'dict'>
21
22  Pretty print of response:
23  {'poller_object': <pytan.pollers.QuestionPoller object at 0x10a808810>,
24   'poller_success': True,
25   'question_object': <taniumpy.object_types.question.Question object at 0x10a810650>,
26   'question_results': <taniumpy.object_types.result_set.ResultSet object at 0x10a6c0410>}
27
28  Equivalent Question if it were to be asked in the Tanium Console:
29  Get Computer Name and Installed Applications from all machines
30
31  CSV Results of response:
32  Computer Name,Name,Silent Uninstall String,Uninstallable,Version
33  Casus-Belli.local,"Image Capture Extension
34  Dictation
35  Wish
36  Uninstall AnyConnect
37  Time Machine
38  AppleGraphicsWarning
39  soagent
40  Feedback Assistant
41  AinuIM
42  vpndownloader
43  Pass Viewer
44  ARDAgent
45  OBEXAgent
46  PressAndHold
47  ..trimmed for brevity..
```

Ask manual question simple single sensor

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

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

Example Python Code

```
1   import os
2   import sys
```

```
3   sys.dont_write_bytecode = True
4
5   # Determine our script name, script dir
6   my_file = os.path.abspath(sys.argv[0])
7   my_dir = os.path.dirname(my_file)
8
9   # determine the pytan lib dir and add it to the path
10  parent_dir = os.path.dirname(my_dir)
11  pytan_root_dir = os.path.dirname(parent_dir)
12  lib_dir = os.path.join(pytan_root_dir, 'lib')
13  path_adds = [lib_dir]
14
15  for aa in path_adds:
16      if aa not in sys.path:
17          sys.path.append(aa)
18
19
20  # connection info for Tanium Server
21  USERNAME = "Tanium User"
22  PASSWORD = "T@n!um"
23  HOST = "172.16.31.128"
24  PORT = "443"
25
26  # Logging conrols
27  LOGLEVEL = 2
28  DEBUGFORMAT = False
29
30  import tempfile
31
32  import pytan
33  handler = pytan.Handler(
34      username=USERNAME,
35      password=PASSWORD,
36      host=HOST,
37      port=PORT,
38      loglevel=LOGLEVEL,
39      debugformat=DEBUGFORMAT,
40  )
41
42  print handler
43
44  # setup the arguments for the handler method
45  kwargs = {}
46  kwargs["sensors"] = u'Computer Name'
47  kwargs["qtype"] = u'manual'
48
49  # call the handler with the ask method, passing in kwargs for arguments
50  response = handler.ask(**kwargs)
51  import pprint, io
52
53  print ""
54  print "Type of response: ", type(response)
55
56  print ""
57  print "Pretty print of response:"
58  print pprint.pformat(response)
59
60  print ""
```

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

Output from Python Code

```
1   Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2   2015-08-07 19:38:10,340 DEBUG     pytan.handler.QuestionPoller: ID 1281: id resolved to 1281
3   2015-08-07 19:38:10,340 DEBUG     pytan.handler.QuestionPoller: ID 1281: expiration resolved to 2015-
4   2015-08-07 19:38:10,340 DEBUG     pytan.handler.QuestionPoller: ID 1281: query_text resolved to Get C
5   2015-08-07 19:38:10,340 DEBUG     pytan.handler.QuestionPoller: ID 1281: id resolved to 1281
6   2015-08-07 19:38:10,340 DEBUG     pytan.handler.QuestionPoller: ID 1281: Object Info resolved to Ques
7   2015-08-07 19:38:10,343 DEBUG     pytan.handler.QuestionPoller: ID 1281: Progress: Tested: 0, Passed:
8   2015-08-07 19:38:10,343 DEBUG     pytan.handler.QuestionPoller: ID 1281: Timing: Started: 2015-08-07
9   2015-08-07 19:38:10,343 INFO      pytan.handler.QuestionPoller: ID 1281: Progress Changed 0% (0 of 2)
10  2015-08-07 19:38:15,351 DEBUG     pytan.handler.QuestionPoller: ID 1281: Progress: Tested: 1, Passed:
11  2015-08-07 19:38:15,351 DEBUG     pytan.handler.QuestionPoller: ID 1281: Timing: Started: 2015-08-07
12  2015-08-07 19:38:15,351 INFO      pytan.handler.QuestionPoller: ID 1281: Progress Changed 50% (1 of 2
13  2015-08-07 19:38:20,357 DEBUG     pytan.handler.QuestionPoller: ID 1281: Progress: Tested: 2, Passed:
14  2015-08-07 19:38:20,357 DEBUG     pytan.handler.QuestionPoller: ID 1281: Timing: Started: 2015-08-07
15  2015-08-07 19:38:20,357 INFO      pytan.handler.QuestionPoller: ID 1281: Progress Changed 100% (2 of
16  2015-08-07 19:38:20,357 INFO      pytan.handler.QuestionPoller: ID 1281: Reached Threshold of 99% (2
17
18  Type of response:  <type 'dict'>
19
20  Pretty print of response:
21  {'poller_object': <pytan.pollers.QuestionPoller object at 0x10a7ecb90>,
22   'poller_success': True,
23   'question_object': <taniumpy.object_types.question.Question object at 0x10a808290>,
24   'question_results': <taniumpy.object_types.result_set.ResultSet object at 0x10a6133d0>}
25
26  Equivalent Question if it were to be asked in the Tanium Console:
27  Get Computer Name from all machines
28
29  CSV Results of response:
30  Computer Name
31  Casus-Belli.local
32  JTANIUM1.localdomain
```

Ask manual 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
import os
import sys
sys.dont_write_bytecode = True

# Determine our script name, script dir
my_file = os.path.abspath(sys.argv[0])
my_dir = os.path.dirname(my_file)

# determine the pytan lib dir and add it to the path
parent_dir = os.path.dirname(my_dir)
pytan_root_dir = os.path.dirname(parent_dir)
lib_dir = os.path.join(pytan_root_dir, 'lib')
path_adds = [lib_dir]

for aa in path_adds:
    if aa not in sys.path:
        sys.path.append(aa)


# connection info for Tanium Server
USERNAME = "Tanium User"
PASSWORD = "T@n!um"
HOST = "172.16.31.128"
PORT = "443"

# Logging conrols
LOGLEVEL = 2
DEBUGFORMAT = False

import tempfile

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'name:Computer Name', u'name:Installed Applications']
kwargs["qtype"] = u'manual'

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

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

Output from Python Code

```
1    Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2    2015-08-07 19:38:20,411 DEBUG     pytan.handler.QuestionPoller: ID 1282: id resolved to 1282
3    2015-08-07 19:38:20,411 DEBUG     pytan.handler.QuestionPoller: ID 1282: expiration resolved to 2015-
4    2015-08-07 19:38:20,411 DEBUG     pytan.handler.QuestionPoller: ID 1282: query_text resolved to Get C
5    2015-08-07 19:38:20,411 DEBUG     pytan.handler.QuestionPoller: ID 1282: id resolved to 1282
6    2015-08-07 19:38:20,411 DEBUG     pytan.handler.QuestionPoller: ID 1282: Object Info resolved to Ques
7    2015-08-07 19:38:20,414 DEBUG     pytan.handler.QuestionPoller: ID 1282: Progress: Tested: 0, Passed:
8    2015-08-07 19:38:20,414 DEBUG     pytan.handler.QuestionPoller: ID 1282: Timing: Started: 2015-08-07
9    2015-08-07 19:38:20,414 INFO      pytan.handler.QuestionPoller: ID 1282: Progress Changed 0% (0 of 2)
10   2015-08-07 19:38:25,422 DEBUG     pytan.handler.QuestionPoller: ID 1282: Progress: Tested: 2, Passed:
11   2015-08-07 19:38:25,422 DEBUG     pytan.handler.QuestionPoller: ID 1282: Timing: Started: 2015-08-07
12   2015-08-07 19:38:25,423 INFO      pytan.handler.QuestionPoller: ID 1282: Progress Changed 100% (2 of
13   2015-08-07 19:38:25,423 INFO      pytan.handler.QuestionPoller: ID 1282: Reached Threshold of 99% (2
14
15   Type of response:  <type 'dict'>
16
17   Pretty print of response:
18   {'poller_object': <pytan.pollers.QuestionPoller object at 0x10a7ecc10>,
19    'poller_success': True,
20    'question_object': <taniumpy.object_types.question.Question object at 0x10a7ec090>,
21    'question_results': <taniumpy.object_types.result_set.ResultSet object at 0x10a7ecc90>}
22
23   Equivalent Question if it were to be asked in the Tanium Console:
24   Get Computer Name and Installed Applications from all machines
25
26   CSV Results of response:
```

```
27  Computer Name,Name,Silent Uninstall String,Uninstallable,Version
28  Casus-Belli.local,"Image Capture Extension
29  Dictation
30  Wish
31  Uninstall AnyConnect
32  Time Machine
33  AppleGraphicsWarning
34  soagent
35  Feedback Assistant
36  AinuIM
37  vpndownloader
38  Pass Viewer
39  ARDAgent
40  OBEXAgent
41  PressAndHold
42  ..trimmed for brevity..
```

Ask manual 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
1   import os
2   import sys
3   sys.dont_write_bytecode = True
4
5   # Determine our script name, script dir
6   my_file = os.path.abspath(sys.argv[0])
7   my_dir = os.path.dirname(my_file)
8
9   # determine the pytan lib dir and add it to the path
10  parent_dir = os.path.dirname(my_dir)
11  pytan_root_dir = os.path.dirname(parent_dir)
12  lib_dir = os.path.join(pytan_root_dir, 'lib')
13  path_adds = [lib_dir]
14
15  for aa in path_adds:
16      if aa not in sys.path:
17          sys.path.append(aa)
18
19
20  # connection info for Tanium Server
21  USERNAME = "Tanium User"
22  PASSWORD = "T@n!um"
23  HOST = "172.16.31.128"
24  PORT = "443"
25
26  # Logging conrols
27  LOGLEVEL = 2
28  DEBUGFORMAT = False
```

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

Output from Python Code

```
1  Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2  2015-08-07 19:38:25,510 DEBUG     pytan.handler.QuestionPoller: ID 1283: id resolved to 1283
3  2015-08-07 19:38:25,510 DEBUG     pytan.handler.QuestionPoller: ID 1283: expiration resolved to 2015-
4  2015-08-07 19:38:25,510 DEBUG     pytan.handler.QuestionPoller: ID 1283: query_text resolved to Get F
5  2015-08-07 19:38:25,510 DEBUG     pytan.handler.QuestionPoller: ID 1283: id resolved to 1283
```

```
6   2015-08-07 19:38:25,510 DEBUG     pytan.handler.QuestionPoller: ID 1283: Object Info resolved to Ques
7   2015-08-07 19:38:25,513 DEBUG     pytan.handler.QuestionPoller: ID 1283: Progress: Tested: 0, Passed:
8   2015-08-07 19:38:25,513 DEBUG     pytan.handler.QuestionPoller: ID 1283: Timing: Started: 2015-08-07
9   2015-08-07 19:38:25,513 INFO      pytan.handler.QuestionPoller: ID 1283: Progress Changed 0% (0 of 2)
10  2015-08-07 19:38:30,521 DEBUG     pytan.handler.QuestionPoller: ID 1283: Progress: Tested: 0, Passed:
11  2015-08-07 19:38:30,521 DEBUG     pytan.handler.QuestionPoller: ID 1283: Timing: Started: 2015-08-07
12  2015-08-07 19:38:35,526 DEBUG     pytan.handler.QuestionPoller: ID 1283: Progress: Tested: 0, Passed:
13  2015-08-07 19:38:35,526 DEBUG     pytan.handler.QuestionPoller: ID 1283: Timing: Started: 2015-08-07
14  2015-08-07 19:38:40,532 DEBUG     pytan.handler.QuestionPoller: ID 1283: Progress: Tested: 0, Passed:
15  2015-08-07 19:38:40,532 DEBUG     pytan.handler.QuestionPoller: ID 1283: Timing: Started: 2015-08-07
16  2015-08-07 19:38:45,536 DEBUG     pytan.handler.QuestionPoller: ID 1283: Progress: Tested: 0, Passed:
17  2015-08-07 19:38:45,536 DEBUG     pytan.handler.QuestionPoller: ID 1283: Timing: Started: 2015-08-07
18  2015-08-07 19:38:50,539 DEBUG     pytan.handler.QuestionPoller: ID 1283: Progress: Tested: 0, Passed:
19  2015-08-07 19:38:50,539 DEBUG     pytan.handler.QuestionPoller: ID 1283: Timing: Started: 2015-08-07
20  2015-08-07 19:38:55,543 DEBUG     pytan.handler.QuestionPoller: ID 1283: Progress: Tested: 0, Passed:
21  2015-08-07 19:38:55,543 DEBUG     pytan.handler.QuestionPoller: ID 1283: Timing: Started: 2015-08-07
22  2015-08-07 19:39:00,547 DEBUG     pytan.handler.QuestionPoller: ID 1283: Progress: Tested: 0, Passed:
23  2015-08-07 19:39:00,547 DEBUG     pytan.handler.QuestionPoller: ID 1283: Timing: Started: 2015-08-07
24  2015-08-07 19:39:05,554 DEBUG     pytan.handler.QuestionPoller: ID 1283: Progress: Tested: 0, Passed:
25  2015-08-07 19:39:05,554 DEBUG     pytan.handler.QuestionPoller: ID 1283: Timing: Started: 2015-08-07
26  2015-08-07 19:39:10,558 DEBUG     pytan.handler.QuestionPoller: ID 1283: Progress: Tested: 0, Passed:
27  2015-08-07 19:39:10,558 DEBUG     pytan.handler.QuestionPoller: ID 1283: Timing: Started: 2015-08-07
28  2015-08-07 19:39:15,561 DEBUG     pytan.handler.QuestionPoller: ID 1283: Progress: Tested: 0, Passed:
29  2015-08-07 19:39:15,561 DEBUG     pytan.handler.QuestionPoller: ID 1283: Timing: Started: 2015-08-07
30  2015-08-07 19:39:20,566 DEBUG     pytan.handler.QuestionPoller: ID 1283: Progress: Tested: 0, Passed:
31  2015-08-07 19:39:20,566 DEBUG     pytan.handler.QuestionPoller: ID 1283: Timing: Started: 2015-08-07
32  2015-08-07 19:39:25,571 DEBUG     pytan.handler.QuestionPoller: ID 1283: Progress: Tested: 0, Passed:
33  2015-08-07 19:39:25,571 DEBUG     pytan.handler.QuestionPoller: ID 1283: Timing: Started: 2015-08-07
34  2015-08-07 19:39:30,577 DEBUG     pytan.handler.QuestionPoller: ID 1283: Progress: Tested: 0, Passed:
35  2015-08-07 19:39:30,577 DEBUG     pytan.handler.QuestionPoller: ID 1283: Timing: Started: 2015-08-07
36  2015-08-07 19:39:35,581 DEBUG     pytan.handler.QuestionPoller: ID 1283: Progress: Tested: 0, Passed:
37  2015-08-07 19:39:35,581 DEBUG     pytan.handler.QuestionPoller: ID 1283: Timing: Started: 2015-08-07
38  2015-08-07 19:39:40,585 DEBUG     pytan.handler.QuestionPoller: ID 1283: Progress: Tested: 0, Passed:
39  2015-08-07 19:39:40,585 DEBUG     pytan.handler.QuestionPoller: ID 1283: Timing: Started: 2015-08-07
40  2015-08-07 19:39:45,588 DEBUG     pytan.handler.QuestionPoller: ID 1283: Progress: Tested: 0, Passed:
41  2015-08-07 19:39:45,588 DEBUG     pytan.handler.QuestionPoller: ID 1283: Timing: Started: 2015-08-07
42  2015-08-07 19:39:50,592 DEBUG     pytan.handler.QuestionPoller: ID 1283: Progress: Tested: 0, Passed:
43  2015-08-07 19:39:50,592 DEBUG     pytan.handler.QuestionPoller: ID 1283: Timing: Started: 2015-08-07
44  2015-08-07 19:39:55,597 DEBUG     pytan.handler.QuestionPoller: ID 1283: Progress: Tested: 0, Passed:
45  2015-08-07 19:39:55,597 DEBUG     pytan.handler.QuestionPoller: ID 1283: Timing: Started: 2015-08-07
46  2015-08-07 19:40:00,603 DEBUG     pytan.handler.QuestionPoller: ID 1283: Progress: Tested: 0, Passed:
47  2015-08-07 19:40:00,603 DEBUG     pytan.handler.QuestionPoller: ID 1283: Timing: Started: 2015-08-07
48  2015-08-07 19:40:05,612 DEBUG     pytan.handler.QuestionPoller: ID 1283: Progress: Tested: 0, Passed:
49  2015-08-07 19:40:05,613 DEBUG     pytan.handler.QuestionPoller: ID 1283: Timing: Started: 2015-08-07
50  2015-08-07 19:40:10,618 DEBUG     pytan.handler.QuestionPoller: ID 1283: Progress: Tested: 1, Passed:
51  2015-08-07 19:40:10,618 DEBUG     pytan.handler.QuestionPoller: ID 1283: Timing: Started: 2015-08-07
52  2015-08-07 19:40:10,618 INFO      pytan.handler.QuestionPoller: ID 1283: Progress Changed 50% (1 of 2
53  2015-08-07 19:40:15,626 DEBUG     pytan.handler.QuestionPoller: ID 1283: Progress: Tested: 1, Passed:
54  2015-08-07 19:40:15,626 DEBUG     pytan.handler.QuestionPoller: ID 1283: Timing: Started: 2015-08-07
55  2015-08-07 19:40:20,631 DEBUG     pytan.handler.QuestionPoller: ID 1283: Progress: Tested: 1, Passed:
56  2015-08-07 19:40:20,631 DEBUG     pytan.handler.QuestionPoller: ID 1283: Timing: Started: 2015-08-07
57  2015-08-07 19:40:25,635 DEBUG     pytan.handler.QuestionPoller: ID 1283: Progress: Tested: 1, Passed:
58  2015-08-07 19:40:25,635 DEBUG     pytan.handler.QuestionPoller: ID 1283: Timing: Started: 2015-08-07
59  2015-08-07 19:40:30,639 DEBUG     pytan.handler.QuestionPoller: ID 1283: Progress: Tested: 2, Passed:
60  2015-08-07 19:40:30,639 DEBUG     pytan.handler.QuestionPoller: ID 1283: Timing: Started: 2015-08-07
61  2015-08-07 19:40:30,639 INFO      pytan.handler.QuestionPoller: ID 1283: Progress Changed 100% (2 of
62  2015-08-07 19:40:30,639 INFO      pytan.handler.QuestionPoller: ID 1283: Reached Threshold of 99% (2
63
```

```
64  Type of response:  <type 'dict'>
65
66  Pretty print of response:
67  {'poller_object': <pytan.pollers.QuestionPoller object at 0x10a613090>,
68   'poller_success': True,
69   'question_object': <taniumpy.object_types.question.Question object at 0x10a613cd0>,
70   'question_results': <taniumpy.object_types.result_set.ResultSet object at 0x10a810650>}
71
72  Equivalent Question if it were to be asked in the Tanium Console:
73  Get Folder Name Search with RegEx Match[Program Files, , No, No, Microsoft.*] from all machines
74
75  CSV Results of response:
76  "Folder Name Search with RegEx Match[Program Files, , No, No, Microsoft.*]"
77  C:\Program Files\VMware\VMware Tools\plugins\vmsvc
78  C:\Program Files\Common Files\Microsoft Shared\VS7Debug
79  C:\Program Files\Tanium\Tanium Server\http\taniumjs\sensor-query\src
80  C:\Program Files\Microsoft SQL Server\110\LocalDB\Binn\Resources\1033
81  C:\Program Files\Tanium\Tanium Server\http\tux\spin\src
82  C:\Program Files\Tanium\Tanium Server\http\taniumjs\archived-question\src
83  C:\Program Files\Tanium\Tanium Module Server\plugins\content
84  C:\Program Files\Tanium\Tanium Server\http\libraries\kendoui\styles\Moonlight
85  C:\Program Files\Common Files\VMware\Drivers\vmci\sockets\include
86  C:\Program Files\Tanium\Tanium Server\http\taniumjs\plugin
87  C:\Program Files\Common Files\Microsoft Shared\ink\ar-SA
88  C:\Program Files\Tanium\Tanium Server\plugins\console\WorkbenchesManager
89  C:\Program Files\Tanium\Tanium Module Server\logs
90  C:\Program Files\Common Files\SpeechEngines\Microsoft
91  ..trimmed for brevity..
```

Ask manual 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

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

```
16          if aa not in sys.path:
17              sys.path.append(aa)
18
19
20      # connection info for Tanium Server
21      USERNAME = "Tanium User"
22      PASSWORD = "T@n!um"
23      HOST = "172.16.31.128"
24      PORT = "443"
25
26      # Logging conrols
27      LOGLEVEL = 2
28      DEBUGFORMAT = False
29
30      import tempfile
31
32      import pytan
33      handler = pytan.Handler(
34          username=USERNAME,
35          password=PASSWORD,
36          host=HOST,
37          port=PORT,
38          loglevel=LOGLEVEL,
39          debugformat=DEBUGFORMAT,
40      )
41
42      print handler
43
44      # setup the arguments for the handler method
45      kwargs = {}
46      kwargs["sensors"] = [u'Folder Name Search with RegEx Match{dirname=Program Files,regex=Microsoft.*}'
47       u'Computer Name']
48      kwargs["qtype"] = u'manual'
49
50      # call the handler with the ask method, passing in kwargs for arguments
51      response = handler.ask(**kwargs)
52      import pprint, io
53
54      print ""
55      print "Type of response: ", type(response)
56
57      print ""
58      print "Pretty print of response:"
59      print pprint.pformat(response)
60
61      print ""
62      print "Equivalent Question if it were to be asked in the Tanium Console: "
63      print response['question_object'].query_text
64
65      # create an IO stream to store CSV results to
66      out = io.BytesIO()
67
68      # call the write_csv() method to convert response to CSV and store it in out
69      response['question_results'].write_csv(out, response['question_results'])
70
71      print ""
72      print "CSV Results of response: "
73      out = out.getvalue()
```

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

Output from Python Code

```
1   Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2   2015-08-07 19:40:30,754 DEBUG    pytan.handler.QuestionPoller: ID 1284: id resolved to 1284
3   2015-08-07 19:40:30,754 DEBUG    pytan.handler.QuestionPoller: ID 1284: expiration resolved to 2015-
4   2015-08-07 19:40:30,754 DEBUG    pytan.handler.QuestionPoller: ID 1284: query_text resolved to Get F
5   2015-08-07 19:40:30,754 DEBUG    pytan.handler.QuestionPoller: ID 1284: id resolved to 1284
6   2015-08-07 19:40:30,754 DEBUG    pytan.handler.QuestionPoller: ID 1284: Object Info resolved to Ques
7   2015-08-07 19:40:30,757 DEBUG    pytan.handler.QuestionPoller: ID 1284: Progress: Tested: 0, Passed:
8   2015-08-07 19:40:30,757 DEBUG    pytan.handler.QuestionPoller: ID 1284: Timing: Started: 2015-08-07
9   2015-08-07 19:40:30,757 INFO     pytan.handler.QuestionPoller: ID 1284: Progress Changed 0% (0 of 2)
10  2015-08-07 19:40:35,761 DEBUG    pytan.handler.QuestionPoller: ID 1284: Progress: Tested: 0, Passed:
11  2015-08-07 19:40:35,761 DEBUG    pytan.handler.QuestionPoller: ID 1284: Timing: Started: 2015-08-07
12  2015-08-07 19:40:40,766 DEBUG    pytan.handler.QuestionPoller: ID 1284: Progress: Tested: 0, Passed:
13  2015-08-07 19:40:40,766 DEBUG    pytan.handler.QuestionPoller: ID 1284: Timing: Started: 2015-08-07
14  2015-08-07 19:40:45,773 DEBUG    pytan.handler.QuestionPoller: ID 1284: Progress: Tested: 0, Passed:
15  2015-08-07 19:40:45,773 DEBUG    pytan.handler.QuestionPoller: ID 1284: Timing: Started: 2015-08-07
16  2015-08-07 19:40:50,777 DEBUG    pytan.handler.QuestionPoller: ID 1284: Progress: Tested: 0, Passed:
17  2015-08-07 19:40:50,777 DEBUG    pytan.handler.QuestionPoller: ID 1284: Timing: Started: 2015-08-07
18  2015-08-07 19:40:55,782 DEBUG    pytan.handler.QuestionPoller: ID 1284: Progress: Tested: 0, Passed:
19  2015-08-07 19:40:55,782 DEBUG    pytan.handler.QuestionPoller: ID 1284: Timing: Started: 2015-08-07
20  2015-08-07 19:41:00,790 DEBUG    pytan.handler.QuestionPoller: ID 1284: Progress: Tested: 0, Passed:
21  2015-08-07 19:41:00,790 DEBUG    pytan.handler.QuestionPoller: ID 1284: Timing: Started: 2015-08-07
22  2015-08-07 19:41:05,798 DEBUG    pytan.handler.QuestionPoller: ID 1284: Progress: Tested: 0, Passed:
23  2015-08-07 19:41:05,798 DEBUG    pytan.handler.QuestionPoller: ID 1284: Timing: Started: 2015-08-07
24  2015-08-07 19:41:10,805 DEBUG    pytan.handler.QuestionPoller: ID 1284: Progress: Tested: 0, Passed:
25  2015-08-07 19:41:10,805 DEBUG    pytan.handler.QuestionPoller: ID 1284: Timing: Started: 2015-08-07
26  2015-08-07 19:41:15,809 DEBUG    pytan.handler.QuestionPoller: ID 1284: Progress: Tested: 0, Passed:
27  2015-08-07 19:41:15,809 DEBUG    pytan.handler.QuestionPoller: ID 1284: Timing: Started: 2015-08-07
28  2015-08-07 19:41:20,813 DEBUG    pytan.handler.QuestionPoller: ID 1284: Progress: Tested: 0, Passed:
29  2015-08-07 19:41:20,813 DEBUG    pytan.handler.QuestionPoller: ID 1284: Timing: Started: 2015-08-07
30  2015-08-07 19:41:25,817 DEBUG    pytan.handler.QuestionPoller: ID 1284: Progress: Tested: 2, Passed:
31  2015-08-07 19:41:25,817 DEBUG    pytan.handler.QuestionPoller: ID 1284: Timing: Started: 2015-08-07
32  2015-08-07 19:41:25,817 INFO     pytan.handler.QuestionPoller: ID 1284: Progress Changed 100% (2 of
33  2015-08-07 19:41:25,817 INFO     pytan.handler.QuestionPoller: ID 1284: Reached Threshold of 99% (2
34
35  Type of response:  <type 'dict'>
36
37  Pretty print of response:
38  {'poller_object': <pytan.pollers.QuestionPoller object at 0x10a614b50>,
39   'poller_success': True,
40   'question_object': <taniumpy.object_types.question.Question object at 0x10a5f51d0>,
41   'question_results': <taniumpy.object_types.result_set.ResultSet object at 0x10a614f10>}
42
43  Equivalent Question if it were to be asked in the Tanium Console:
44  Get Folder Name Search with RegEx Match[Program Files, , No, No, Microsoft.*] and Computer Name from
45
46  CSV Results of response:
47  Computer Name,"Folder Name Search with RegEx Match[Program Files, , No, No, Microsoft.*]"
48  Casus-Belli.local,Windows Only
49  JTANIUM1.localdomain,"C:\Program Files\VMware\VMware Tools\plugins\vmsvc
```

```
50  C:\Program Files\Common Files\Microsoft Shared\VS7Debug
51  C:\Program Files\Tanium\Tanium Server\http\taniumjs\sensor-query\src
52  C:\Program Files\Microsoft SQL Server\110\LocalDB\Binn\Resources\1033
53  C:\Program Files\Tanium\Tanium Server\http\tux\spin\src
54  C:\Program Files\Tanium\Tanium Server\http\taniumjs\archived-question\src
55  C:\Program Files\Tanium\Tanium Module Server\plugins\content
56  C:\Program Files\Tanium\Tanium Server\http\libraries\kendoui\styles\Moonlight
57  C:\Program Files\Common Files\VMware\Drivers\vmci\sockets\include
58  C:\Program Files\Tanium\Tanium Server\http\taniumjs\plugin
59  C:\Program Files\Common Files\Microsoft Shared\ink\ar-SA
60  C:\Program Files\Tanium\Tanium Server\plugins\console\WorkbenchesManager
61  C:\Program Files\Tanium\Tanium Module Server\logs
62  ..trimmed for brevity..
```

Ask manual 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
1   import os
2   import sys
3   sys.dont_write_bytecode = True
4
5   # Determine our script name, script dir
6   my_file = os.path.abspath(sys.argv[0])
7   my_dir = os.path.dirname(my_file)
8
9   # determine the pytan lib dir and add it to the path
10  parent_dir = os.path.dirname(my_dir)
11  pytan_root_dir = os.path.dirname(parent_dir)
12  lib_dir = os.path.join(pytan_root_dir, 'lib')
13  path_adds = [lib_dir]
14
15  for aa in path_adds:
16      if aa not in sys.path:
17          sys.path.append(aa)
18
19
20  # connection info for Tanium Server
21  USERNAME = "Tanium User"
22  PASSWORD = "T@n!um"
23  HOST = "172.16.31.128"
24  PORT = "443"
25
26  # Logging conrols
27  LOGLEVEL = 2
28  DEBUGFORMAT = False
29
30  import tempfile
31
32  import pytan
33  handler = pytan.Handler(
```

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

Output from Python Code

```
1    Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2    2015-08-07 19:41:25,887 DEBUG     pytan.handler.QuestionPoller: ID 1286: id resolved to 1286
3    2015-08-07 19:41:25,887 DEBUG     pytan.handler.QuestionPoller: ID 1286: expiration resolved to 2015-
4    2015-08-07 19:41:25,887 DEBUG     pytan.handler.QuestionPoller: ID 1286: query_text resolved to Get C
5    2015-08-07 19:41:25,887 DEBUG     pytan.handler.QuestionPoller: ID 1286: id resolved to 1286
6    2015-08-07 19:41:25,887 DEBUG     pytan.handler.QuestionPoller: ID 1286: Object Info resolved to Ques
7    2015-08-07 19:41:25,892 DEBUG     pytan.handler.QuestionPoller: ID 1286: Progress: Tested: 0, Passed:
8    2015-08-07 19:41:25,892 DEBUG     pytan.handler.QuestionPoller: ID 1286: Timing: Started: 2015-08-07
9    2015-08-07 19:41:25,892 INFO      pytan.handler.QuestionPoller: ID 1286: Progress Changed 0% (0 of 2)
10   2015-08-07 19:41:30,900 DEBUG     pytan.handler.QuestionPoller: ID 1286: Progress: Tested: 0, Passed:
```

```
11   2015-08-07 19:41:30,900 DEBUG     pytan.handler.QuestionPoller: ID 1286: Timing: Started: 2015-08-07
12   2015-08-07 19:41:35,905 DEBUG     pytan.handler.QuestionPoller: ID 1286: Progress: Tested: 0, Passed:
13   2015-08-07 19:41:35,905 DEBUG     pytan.handler.QuestionPoller: ID 1286: Timing: Started: 2015-08-07
14   2015-08-07 19:41:40,908 DEBUG     pytan.handler.QuestionPoller: ID 1286: Progress: Tested: 0, Passed:
15   2015-08-07 19:41:40,908 DEBUG     pytan.handler.QuestionPoller: ID 1286: Timing: Started: 2015-08-07
16   2015-08-07 19:41:45,915 DEBUG     pytan.handler.QuestionPoller: ID 1286: Progress: Tested: 0, Passed:
17   2015-08-07 19:41:45,915 DEBUG     pytan.handler.QuestionPoller: ID 1286: Timing: Started: 2015-08-07
18   2015-08-07 19:41:50,919 DEBUG     pytan.handler.QuestionPoller: ID 1286: Progress: Tested: 0, Passed:
19   2015-08-07 19:41:50,919 DEBUG     pytan.handler.QuestionPoller: ID 1286: Timing: Started: 2015-08-07
20   2015-08-07 19:41:55,923 DEBUG     pytan.handler.QuestionPoller: ID 1286: Progress: Tested: 1, Passed:
21   2015-08-07 19:41:55,924 DEBUG     pytan.handler.QuestionPoller: ID 1286: Timing: Started: 2015-08-07
22   2015-08-07 19:41:55,924 INFO      pytan.handler.QuestionPoller: ID 1286: Progress Changed 50% (1 of 2
23   2015-08-07 19:42:00,928 DEBUG     pytan.handler.QuestionPoller: ID 1286: Progress: Tested: 1, Passed:
24   2015-08-07 19:42:00,929 DEBUG     pytan.handler.QuestionPoller: ID 1286: Timing: Started: 2015-08-07
25   2015-08-07 19:42:05,933 DEBUG     pytan.handler.QuestionPoller: ID 1286: Progress: Tested: 2, Passed:
26   2015-08-07 19:42:05,933 DEBUG     pytan.handler.QuestionPoller: ID 1286: Timing: Started: 2015-08-07
27   2015-08-07 19:42:05,933 INFO      pytan.handler.QuestionPoller: ID 1286: Progress Changed 100% (2 of
28   2015-08-07 19:42:05,933 INFO      pytan.handler.QuestionPoller: ID 1286: Reached Threshold of 99% (2

30   Type of response:  <type 'dict'>

32   Pretty print of response:
33   {'poller_object': <pytan.pollers.QuestionPoller object at 0x10a614e10>,
34    'poller_success': True,
35    'question_object': <taniumpy.object_types.question.Question object at 0x10a613450>,
36    'question_results': <taniumpy.object_types.result_set.ResultSet object at 0x10a614490>}

38   Equivalent Question if it were to be asked in the Tanium Console:
39   Get Computer Name[Dweedle] from all machines

41   CSV Results of response:
42   Computer Name[Dweedle]
43   [no results]
44   JTANIUM1
```

Ask manual 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
import os
import sys
sys.dont_write_bytecode = True

# Determine our script name, script dir
my_file = os.path.abspath(sys.argv[0])
my_dir = os.path.dirname(my_file)

# determine the pytan lib dir and add it to the path
parent_dir = os.path.dirname(my_dir)
```

```
11   pytan_root_dir = os.path.dirname(parent_dir)
12   lib_dir = os.path.join(pytan_root_dir, 'lib')
13   path_adds = [lib_dir]
14
15   for aa in path_adds:
16       if aa not in sys.path:
17           sys.path.append(aa)
18
19
20   # connection info for Tanium Server
21   USERNAME = "Tanium User"
22   PASSWORD = "T@n!um"
23   HOST = "172.16.31.128"
24   PORT = "443"
25
26   # Logging conrols
27   LOGLEVEL = 2
28   DEBUGFORMAT = False
29
30   import tempfile
31
32   import pytan
33   handler = pytan.Handler(
34       username=USERNAME,
35       password=PASSWORD,
36       host=HOST,
37       port=PORT,
38       loglevel=LOGLEVEL,
39       debugformat=DEBUGFORMAT,
40   )
41
42   print handler
43
44   # setup the arguments for the handler method
45   kwargs = {}
46   kwargs["sensors"] = u'Folder Name Search with RegEx Match'
47   kwargs["qtype"] = u'manual'
48
49   # call the handler with the ask method, passing in kwargs for arguments
50   response = handler.ask(**kwargs)
51   import pprint, io
52
53   print ""
54   print "Type of response: ", type(response)
55
56   print ""
57   print "Pretty print of response:"
58   print pprint.pformat(response)
59
60   print ""
61   print "Equivalent Question if it were to be asked in the Tanium Console: "
62   print response['question_object'].query_text
63
64   # create an IO stream to store CSV results to
65   out = io.BytesIO()
66
67   # call the write_csv() method to convert response to CSV and store it in out
68   response['question_results'].write_csv(out, response['question_results'])
```

```
69
70  print ""
71  print "CSV Results of response: "
72  out = out.getvalue()
73  if len(out.splitlines()) > 15:
74      out = out.splitlines()[0:15]
75      out.append('..trimmed for brevity..')
76      out = '\n'.join(out)
77  print out
```

Output from Python Code

```
1   Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2   2015-08-07 19:42:06,009 DEBUG    pytan.handler.QuestionPoller: ID 1288: id resolved to 1288
3   2015-08-07 19:42:06,010 DEBUG    pytan.handler.QuestionPoller: ID 1288: expiration resolved to 2015-
4   2015-08-07 19:42:06,010 DEBUG    pytan.handler.QuestionPoller: ID 1288: query_text resolved to Get F
5   2015-08-07 19:42:06,010 DEBUG    pytan.handler.QuestionPoller: ID 1288: id resolved to 1288
6   2015-08-07 19:42:06,010 DEBUG    pytan.handler.QuestionPoller: ID 1288: Object Info resolved to Ques
7   2015-08-07 19:42:06,013 DEBUG    pytan.handler.QuestionPoller: ID 1288: Progress: Tested: 0, Passed:
8   2015-08-07 19:42:06,013 DEBUG    pytan.handler.QuestionPoller: ID 1288: Timing: Started: 2015-08-07
9   2015-08-07 19:42:06,013 INFO     pytan.handler.QuestionPoller: ID 1288: Progress Changed 0% (0 of 2)
10  2015-08-07 19:42:11,021 DEBUG    pytan.handler.QuestionPoller: ID 1288: Progress: Tested: 0, Passed:
11  2015-08-07 19:42:11,021 DEBUG    pytan.handler.QuestionPoller: ID 1288: Timing: Started: 2015-08-07
12  2015-08-07 19:42:16,025 DEBUG    pytan.handler.QuestionPoller: ID 1288: Progress: Tested: 0, Passed:
13  2015-08-07 19:42:16,025 DEBUG    pytan.handler.QuestionPoller: ID 1288: Timing: Started: 2015-08-07
14  2015-08-07 19:42:21,032 DEBUG    pytan.handler.QuestionPoller: ID 1288: Progress: Tested: 0, Passed:
15  2015-08-07 19:42:21,032 DEBUG    pytan.handler.QuestionPoller: ID 1288: Timing: Started: 2015-08-07
16  2015-08-07 19:42:26,037 DEBUG    pytan.handler.QuestionPoller: ID 1288: Progress: Tested: 2, Passed:
17  2015-08-07 19:42:26,037 DEBUG    pytan.handler.QuestionPoller: ID 1288: Timing: Started: 2015-08-07
18  2015-08-07 19:42:26,038 INFO     pytan.handler.QuestionPoller: ID 1288: Progress Changed 100% (2 of
19  2015-08-07 19:42:26,038 INFO     pytan.handler.QuestionPoller: ID 1288: Reached Threshold of 99% (2
20
21  Type of response:  <type 'dict'>
22
23  Pretty print of response:
24  {'poller_object': <pytan.pollers.QuestionPoller object at 0x10a6147d0>,
25   'poller_success': True,
26   'question_object': <taniumpy.object_types.question.Question object at 0x10a613790>,
27   'question_results': <taniumpy.object_types.result_set.ResultSet object at 0x10a5f5190>}
28
29  Equivalent Question if it were to be asked in the Tanium Console:
30  Get Folder Name Search with RegEx Match[, , No, No] from all machines
31
32  CSV Results of response:
33  Count,"Folder Name Search with RegEx Match[, , No, No]"
34  24705,[too many results]
35  1,C:\Windows\winsxs\amd64_microsoft-windows-s..structure.resources_31bf3856ad364e35_6.1.7600.16385_e
36  1,C:\Windows\winsxs\x86_microsoft-windows-e..-host-authenticator_31bf3856ad364e35_6.1.7601.17514_non
37  1,C:\Windows\winsxs\amd64_microsoft-windows-ocspsvc_31bf3856ad364e35_6.1.7601.22807_none_3bfeae72930
38  1,C:\Windows\winsxs\amd64_microsoft-windows-c..ityclient.resources_31bf3856ad364e35_6.1.7601.22865_e
39  1,C:\Windows\assembly\NativeImages_v2.0.50727_64\System.Xml
40  1,C:\Windows\winsxs\amd64_microsoft-windows-winsetupui_31bf3856ad364e35_6.1.7601.18804_none_bd3cf1bb
41  1,C:\Windows\winsxs\amd64_microsoft-windows-scripting.resources_31bf3856ad364e35_6.1.7600.16385_en-u
42  1,C:\Windows\winsxs\x86_microsoft-windows-mlang.resources_31bf3856ad364e35_6.1.7600.16385_ru-ru_cf3a
43  1,C:\Windows\winsxs\x86_microsoft-windows-minkernelapinamespace_31bf3856ad364e35_6.1.7601.21728_none
44  1,C:\Users\Jim Olsen\AppData\Local\Google
45  1,C:\Windows\winsxs\x86_microsoft-windows-e..nt-client.resources_31bf3856ad364e35_6.1.7600.16385_en-
```

```
46   1,C:\Windows\winsxs\amd64_microsoft-windows-d..e-eashared-kjshared_31bf3856ad364e35_6.1.7600.16385_n
47   1,C:\Windows\assembly\NativeImages_v4.0.30319_32\RadLangSvc
48   ..trimmed for brevity..
```

Ask manual 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
1    import os
2    import sys
3    sys.dont_write_bytecode = True
4
5    # Determine our script name, script dir
6    my_file = os.path.abspath(sys.argv[0])
7    my_dir = os.path.dirname(my_file)
8
9    # determine the pytan lib dir and add it to the path
10   parent_dir = os.path.dirname(my_dir)
11   pytan_root_dir = os.path.dirname(parent_dir)
12   lib_dir = os.path.join(pytan_root_dir, 'lib')
13   path_adds = [lib_dir]
14
15   for aa in path_adds:
16       if aa not in sys.path:
17           sys.path.append(aa)
18
19
20   # connection info for Tanium Server
21   USERNAME = "Tanium User"
22   PASSWORD = "T@n!um"
23   HOST = "172.16.31.128"
24   PORT = "443"
25
26   # Logging conrols
27   LOGLEVEL = 2
28   DEBUGFORMAT = False
29
30   import tempfile
31
32   import pytan
33   handler = pytan.Handler(
34       username=USERNAME,
35       password=PASSWORD,
36       host=HOST,
37       port=PORT,
38       loglevel=LOGLEVEL,
39       debugformat=DEBUGFORMAT,
40   )
41
```

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

Output from Python Code

```
1   Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2   2015-08-07 19:42:26,175 DEBUG    pytan.handler.QuestionPoller: ID 1289: id resolved to 1289
3   2015-08-07 19:42:26,175 DEBUG    pytan.handler.QuestionPoller: ID 1289: expiration resolved to 2015-
4   2015-08-07 19:42:26,175 DEBUG    pytan.handler.QuestionPoller: ID 1289: query_text resolved to Get F
5   2015-08-07 19:42:26,175 DEBUG    pytan.handler.QuestionPoller: ID 1289: id resolved to 1289
6   2015-08-07 19:42:26,175 DEBUG    pytan.handler.QuestionPoller: ID 1289: Object Info resolved to Ques
7   2015-08-07 19:42:26,178 DEBUG    pytan.handler.QuestionPoller: ID 1289: Progress: Tested: 0, Passed:
8   2015-08-07 19:42:26,178 DEBUG    pytan.handler.QuestionPoller: ID 1289: Timing: Started: 2015-08-07
9   2015-08-07 19:42:26,178 INFO     pytan.handler.QuestionPoller: ID 1289: Progress Changed 0% (0 of 2)
10  2015-08-07 19:42:31,183 DEBUG    pytan.handler.QuestionPoller: ID 1289: Progress: Tested: 0, Passed:
11  2015-08-07 19:42:31,183 DEBUG    pytan.handler.QuestionPoller: ID 1289: Timing: Started: 2015-08-07
12  2015-08-07 19:42:36,189 DEBUG    pytan.handler.QuestionPoller: ID 1289: Progress: Tested: 1, Passed:
13  2015-08-07 19:42:36,189 DEBUG    pytan.handler.QuestionPoller: ID 1289: Timing: Started: 2015-08-07
14  2015-08-07 19:42:36,189 INFO     pytan.handler.QuestionPoller: ID 1289: Progress Changed 50% (1 of 2
15  2015-08-07 19:42:41,193 DEBUG    pytan.handler.QuestionPoller: ID 1289: Progress: Tested: 1, Passed:
16  2015-08-07 19:42:41,194 DEBUG    pytan.handler.QuestionPoller: ID 1289: Timing: Started: 2015-08-07
17  2015-08-07 19:42:46,197 DEBUG    pytan.handler.QuestionPoller: ID 1289: Progress: Tested: 1, Passed:
18  2015-08-07 19:42:46,197 DEBUG    pytan.handler.QuestionPoller: ID 1289: Timing: Started: 2015-08-07
```

```
19   2015-08-07 19:42:51,202 DEBUG     pytan.handler.QuestionPoller: ID 1289: Progress: Tested: 1, Passed:
20   2015-08-07 19:42:51,202 DEBUG     pytan.handler.QuestionPoller: ID 1289: Timing: Started: 2015-08-07
21   2015-08-07 19:42:56,207 DEBUG     pytan.handler.QuestionPoller: ID 1289: Progress: Tested: 2, Passed:
22   2015-08-07 19:42:56,207 DEBUG     pytan.handler.QuestionPoller: ID 1289: Timing: Started: 2015-08-07
23   2015-08-07 19:42:56,207 INFO      pytan.handler.QuestionPoller: ID 1289: Progress Changed 100% (2 of
24   2015-08-07 19:42:56,207 INFO      pytan.handler.QuestionPoller: ID 1289: Reached Threshold of 99% (2
25
26   Type of response:  <type 'dict'>
27
28   Pretty print of response:
29   {'poller_object': <pytan.pollers.QuestionPoller object at 0x10a614f50>,
30    'poller_success': True,
31    'question_object': <taniumpy.object_types.question.Question object at 0x10a5f5190>,
32    'question_results': <taniumpy.object_types.result_set.ResultSet object at 0x10a615c10>}
33
34   Equivalent Question if it were to be asked in the Tanium Console:
35   Get Folder Name Search with RegEx Match[Program Files, , No, No, Microsoft.*] containing "Shared" fr
36
37   CSV Results of response:
38   "Folder Name Search with RegEx Match[Program Files, , No, No, Microsoft.*]"
39   [no results]
40   C:\Program Files\Common Files\Microsoft Shared\VS7Debug
41   C:\Program Files\Common Files\Microsoft Shared\ink\ar-SA
42   C:\Program Files\Common Files\Microsoft Shared\ink\ru-RU
43   C:\Program Files\Common Files\Microsoft Shared\ink\fsdefinitions\keypad
44   C:\Program Files\Common Files\Microsoft Shared\ink
45   C:\Program Files\Common Files\Microsoft Shared\ink\sv-SE
46   C:\Program Files\Common Files\Microsoft Shared\ink\uk-UA
47   C:\Program Files\Common Files\Microsoft Shared\ink\sl-SI
48   C:\Program Files\Common Files\Microsoft Shared\ink\hu-HU
49   C:\Program Files\Common Files\Microsoft Shared\ink\zh-TW
50   C:\Program Files\Common Files\Microsoft Shared\ink\zh-CN
51   C:\Program Files\Common Files\Microsoft Shared\ink\fi-FI
52   C:\Program Files\Common Files\Microsoft Shared
53   ..trimmed for brevity..
```

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

```
7   my_dir = os.path.dirname(my_file)
8
9   # determine the pytan lib dir and add it to the path
10  parent_dir = os.path.dirname(my_dir)
11  pytan_root_dir = os.path.dirname(parent_dir)
12  lib_dir = os.path.join(pytan_root_dir, 'lib')
13  path_adds = [lib_dir]
14
15  for aa in path_adds:
16      if aa not in sys.path:
17          sys.path.append(aa)
18
19
20  # connection info for Tanium Server
21  USERNAME = "Tanium User"
22  PASSWORD = "T@n!um"
23  HOST = "172.16.31.128"
24  PORT = "443"
25
26  # Logging conrols
27  LOGLEVEL = 2
28  DEBUGFORMAT = False
29
30  import tempfile
31
32  import pytan
33  handler = pytan.Handler(
34      username=USERNAME,
35      password=PASSWORD,
36      host=HOST,
37      port=PORT,
38      loglevel=LOGLEVEL,
39      debugformat=DEBUGFORMAT,
40  )
41
42  print handler
43
44  # setup the arguments for the handler method
45  kwargs = {}
46  kwargs["sensors"] = u'Operating System, that contains:Windows, opt:max_data_age:3600, opt:value_type
47  kwargs["qtype"] = u'manual'
48
49  # call the handler with the ask method, passing in kwargs for arguments
50  response = handler.ask(**kwargs)
51  import pprint, io
52
53  print ""
54  print "Type of response: ", type(response)
55
56  print ""
57  print "Pretty print of response:"
58  print pprint.pformat(response)
59
60  print ""
61  print "Equivalent Question if it were to be asked in the Tanium Console: "
62  print response['question_object'].query_text
63
64  # create an IO stream to store CSV results to
```

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

Output from Python Code

```
1    Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2    2015-08-07 19:42:56,259 DEBUG     pytan.handler.QuestionPoller: ID 1290: id resolved to 1290
3    2015-08-07 19:42:56,259 DEBUG     pytan.handler.QuestionPoller: ID 1290: expiration resolved to 2015-
4    2015-08-07 19:42:56,259 DEBUG     pytan.handler.QuestionPoller: ID 1290: query_text resolved to Get C
5    2015-08-07 19:42:56,259 DEBUG     pytan.handler.QuestionPoller: ID 1290: id resolved to 1290
6    2015-08-07 19:42:56,259 DEBUG     pytan.handler.QuestionPoller: ID 1290: Object Info resolved to Ques
7    2015-08-07 19:42:56,262 DEBUG     pytan.handler.QuestionPoller: ID 1290: Progress: Tested: 0, Passed:
8    2015-08-07 19:42:56,262 DEBUG     pytan.handler.QuestionPoller: ID 1290: Timing: Started: 2015-08-07
9    2015-08-07 19:42:56,262 INFO      pytan.handler.QuestionPoller: ID 1290: Progress Changed 0% (0 of 2)
10   2015-08-07 19:43:01,266 DEBUG     pytan.handler.QuestionPoller: ID 1290: Progress: Tested: 1, Passed:
11   2015-08-07 19:43:01,266 DEBUG     pytan.handler.QuestionPoller: ID 1290: Timing: Started: 2015-08-07
12   2015-08-07 19:43:01,266 INFO      pytan.handler.QuestionPoller: ID 1290: Progress Changed 50% (1 of 2
13   2015-08-07 19:43:06,271 DEBUG     pytan.handler.QuestionPoller: ID 1290: Progress: Tested: 2, Passed:
14   2015-08-07 19:43:06,271 DEBUG     pytan.handler.QuestionPoller: ID 1290: Timing: Started: 2015-08-07
15   2015-08-07 19:43:06,271 INFO      pytan.handler.QuestionPoller: ID 1290: Progress Changed 100% (2 of
16   2015-08-07 19:43:06,271 INFO      pytan.handler.QuestionPoller: ID 1290: Reached Threshold of 99% (2
17
18   Type of response:  <type 'dict'>
19
20   Pretty print of response:
21   {'poller_object': <pytan.pollers.QuestionPoller object at 0x10a5f5f90>,
22    'poller_success': True,
23    'question_object': <taniumpy.object_types.question.Question object at 0x10a613b10>,
24    'question_results': <taniumpy.object_types.result_set.ResultSet object at 0x10a5b9cd0>}
25
26   Equivalent Question if it were to be asked in the Tanium Console:
27   Get Operating System containing "Windows" from all machines
28
29   CSV Results of response:
30   Operating System
31   [no results]
32   Windows Server 2008 R2 Standard
```

Ask manual 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

```python
import os
import sys
sys.dont_write_bytecode = True

# Determine our script name, script dir
my_file = os.path.abspath(sys.argv[0])
my_dir = os.path.dirname(my_file)

# determine the pytan lib dir and add it to the path
parent_dir = os.path.dirname(my_dir)
pytan_root_dir = os.path.dirname(parent_dir)
lib_dir = os.path.join(pytan_root_dir, 'lib')
path_adds = [lib_dir]

for aa in path_adds:
    if aa not in sys.path:
        sys.path.append(aa)


# connection info for Tanium Server
USERNAME = "Tanium User"
PASSWORD = "T@n!um"
HOST = "172.16.31.128"
PORT = "443"

# Logging conrols
LOGLEVEL = 2
DEBUGFORMAT = False

import tempfile

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'

# 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)
```

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

Output from Python Code

```
1   Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2   2015-08-07 19:43:06,319 DEBUG     pytan.handler.QuestionPoller: ID 1291: id resolved to 1291
3   2015-08-07 19:43:06,319 DEBUG     pytan.handler.QuestionPoller: ID 1291: expiration resolved to 2015-
4   2015-08-07 19:43:06,319 DEBUG     pytan.handler.QuestionPoller: ID 1291: query_text resolved to Get O
5   2015-08-07 19:43:06,319 DEBUG     pytan.handler.QuestionPoller: ID 1291: id resolved to 1291
6   2015-08-07 19:43:06,319 DEBUG     pytan.handler.QuestionPoller: ID 1291: Object Info resolved to Ques
7   2015-08-07 19:43:06,322 DEBUG     pytan.handler.QuestionPoller: ID 1291: Progress: Tested: 0, Passed:
8   2015-08-07 19:43:06,322 DEBUG     pytan.handler.QuestionPoller: ID 1291: Timing: Started: 2015-08-07
9   2015-08-07 19:43:06,322 INFO      pytan.handler.QuestionPoller: ID 1291: Progress Changed 0% (0 of 2)
10  2015-08-07 19:43:11,327 DEBUG     pytan.handler.QuestionPoller: ID 1291: Progress: Tested: 1, Passed:
11  2015-08-07 19:43:11,327 DEBUG     pytan.handler.QuestionPoller: ID 1291: Timing: Started: 2015-08-07
12  2015-08-07 19:43:11,327 INFO      pytan.handler.QuestionPoller: ID 1291: Progress Changed 50% (1 of 2
13  2015-08-07 19:43:16,332 DEBUG     pytan.handler.QuestionPoller: ID 1291: Progress: Tested: 2, Passed:
14  2015-08-07 19:43:16,332 DEBUG     pytan.handler.QuestionPoller: ID 1291: Timing: Started: 2015-08-07
15  2015-08-07 19:43:16,333 INFO      pytan.handler.QuestionPoller: ID 1291: Progress Changed 100% (2 of
16  2015-08-07 19:43:16,333 INFO      pytan.handler.QuestionPoller: ID 1291: Reached Threshold of 99% (2
17
18  Type of response:  <type 'dict'>
19
20  Pretty print of response:
21  {'poller_object': <pytan.pollers.QuestionPoller object at 0x10a614dd0>,
22   'poller_success': True,
23   'question_object': <taniumpy.object_types.question.Question object at 0x10a614190>,
24   'question_results': <taniumpy.object_types.result_set.ResultSet object at 0x10a5b9cd0>}
25
26  Equivalent Question if it were to be asked in the Tanium Console:
27  Get Operating System containing "Windows" from all machines
28
29  CSV Results of response:
30  Operating System
31  [no results]
```

```
32  Windows Server 2008 R2 Standard
```

Ask manual 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
1   import os
2   import sys
3   sys.dont_write_bytecode = True
4
5   # Determine our script name, script dir
6   my_file = os.path.abspath(sys.argv[0])
7   my_dir = os.path.dirname(my_file)
8
9   # determine the pytan lib dir and add it to the path
10  parent_dir = os.path.dirname(my_dir)
11  pytan_root_dir = os.path.dirname(parent_dir)
12  lib_dir = os.path.join(pytan_root_dir, 'lib')
13  path_adds = [lib_dir]
14
15  for aa in path_adds:
16      if aa not in sys.path:
17          sys.path.append(aa)
18
19
20  # connection info for Tanium Server
21  USERNAME = "Tanium User"
22  PASSWORD = "T@n!um"
23  HOST = "172.16.31.128"
24  PORT = "443"
25
26  # Logging conrols
27  LOGLEVEL = 2
28  DEBUGFORMAT = False
29
30  import tempfile
31
32  import pytan
33  handler = pytan.Handler(
34      username=USERNAME,
35      password=PASSWORD,
36      host=HOST,
37      port=PORT,
38      loglevel=LOGLEVEL,
39      debugformat=DEBUGFORMAT,
40  )
41
42  print handler
```

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

Output from Python Code

```
1    Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2    2015-08-07 19:43:16,405 DEBUG    pytan.handler.QuestionPoller: ID 1294: id resolved to 1294
3    2015-08-07 19:43:16,405 DEBUG    pytan.handler.QuestionPoller: ID 1294: expiration resolved to 2015-
4    2015-08-07 19:43:16,405 DEBUG    pytan.handler.QuestionPoller: ID 1294: query_text resolved to Get F
5    2015-08-07 19:43:16,405 DEBUG    pytan.handler.QuestionPoller: ID 1294: id resolved to 1294
6    2015-08-07 19:43:16,405 DEBUG    pytan.handler.QuestionPoller: ID 1294: Object Info resolved to Ques
7    2015-08-07 19:43:16,408 DEBUG    pytan.handler.QuestionPoller: ID 1294: Progress: Tested: 0, Passed:
8    2015-08-07 19:43:16,408 DEBUG    pytan.handler.QuestionPoller: ID 1294: Timing: Started: 2015-08-07
9    2015-08-07 19:43:16,408 INFO     pytan.handler.QuestionPoller: ID 1294: Progress Changed 0% (0 of 2)
10   2015-08-07 19:43:21,414 DEBUG    pytan.handler.QuestionPoller: ID 1294: Progress: Tested: 0, Passed:
11   2015-08-07 19:43:21,414 DEBUG    pytan.handler.QuestionPoller: ID 1294: Timing: Started: 2015-08-07
12   2015-08-07 19:43:26,420 DEBUG    pytan.handler.QuestionPoller: ID 1294: Progress: Tested: 0, Passed:
13   2015-08-07 19:43:26,420 DEBUG    pytan.handler.QuestionPoller: ID 1294: Timing: Started: 2015-08-07
14   2015-08-07 19:43:31,424 DEBUG    pytan.handler.QuestionPoller: ID 1294: Progress: Tested: 0, Passed:
15   2015-08-07 19:43:31,424 DEBUG    pytan.handler.QuestionPoller: ID 1294: Timing: Started: 2015-08-07
16   2015-08-07 19:43:36,428 DEBUG    pytan.handler.QuestionPoller: ID 1294: Progress: Tested: 1, Passed:
17   2015-08-07 19:43:36,428 DEBUG    pytan.handler.QuestionPoller: ID 1294: Timing: Started: 2015-08-07
18   2015-08-07 19:43:36,428 INFO     pytan.handler.QuestionPoller: ID 1294: Progress Changed 50% (1 of 2
19   2015-08-07 19:43:41,432 DEBUG    pytan.handler.QuestionPoller: ID 1294: Progress: Tested: 1, Passed:
```

```
20  2015-08-07 19:43:41,432 DEBUG     pytan.handler.QuestionPoller: ID 1294: Timing: Started: 2015-08-07
21  2015-08-07 19:43:46,441 DEBUG     pytan.handler.QuestionPoller: ID 1294: Progress: Tested: 1, Passed:
22  2015-08-07 19:43:46,442 DEBUG     pytan.handler.QuestionPoller: ID 1294: Timing: Started: 2015-08-07
23  2015-08-07 19:43:51,449 DEBUG     pytan.handler.QuestionPoller: ID 1294: Progress: Tested: 2, Passed:
24  2015-08-07 19:43:51,449 DEBUG     pytan.handler.QuestionPoller: ID 1294: Timing: Started: 2015-08-07
25  2015-08-07 19:43:51,449 INFO      pytan.handler.QuestionPoller: ID 1294: Progress Changed 100% (2 of
26  2015-08-07 19:43:51,449 INFO      pytan.handler.QuestionPoller: ID 1294: Reached Threshold of 99% (2

28  Type of response:  <type 'dict'>

30  Pretty print of response:
31  {'poller_object': <pytan.pollers.QuestionPoller object at 0x10a615f10>,
32   'poller_success': True,
33   'question_object': <taniumpy.object_types.question.Question object at 0x10a5f57d0>,
34   'question_results': <taniumpy.object_types.result_set.ResultSet object at 0x10a5b9a10>}

36  Equivalent Question if it were to be asked in the Tanium Console:
37  Get Folder Name Search with RegEx Match[Program Files, , No, No, Microsoft.*] containing "Shared" fr

39  CSV Results of response:
40  "Folder Name Search with RegEx Match[Program Files, , No, No, Microsoft.*]"
41  [no results]
42  C:\Program Files\Common Files\Microsoft Shared\VS7Debug
43  C:\Program Files\Common Files\Microsoft Shared\ink\ar-SA
44  C:\Program Files\Common Files\Microsoft Shared\ink\ru-RU
45  C:\Program Files\Common Files\Microsoft Shared\ink\fsdefinitions\keypad
46  C:\Program Files\Common Files\Microsoft Shared\ink
47  C:\Program Files\Common Files\Microsoft Shared\ink\sv-SE
48  C:\Program Files\Common Files\Microsoft Shared\ink\uk-UA
49  C:\Program Files\Common Files\Microsoft Shared\ink\sl-SI
50  C:\Program Files\Common Files\Microsoft Shared\ink\hu-HU
51  C:\Program Files\Common Files\Microsoft Shared\ink\zh-TW
52  C:\Program Files\Common Files\Microsoft Shared\ink\zh-CN
53  C:\Program Files\Common Files\Microsoft Shared\ink\fi-FI
54  C:\Program Files\Common Files\Microsoft Shared
55  ..trimmed for brevity..
```

Ask manual question sensor with filter and 3 options

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

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

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

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

Example Python Code

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

```
6   my_file = os.path.abspath(sys.argv[0])
7   my_dir = os.path.dirname(my_file)
8
9   # determine the pytan lib dir and add it to the path
10  parent_dir = os.path.dirname(my_dir)
11  pytan_root_dir = os.path.dirname(parent_dir)
12  lib_dir = os.path.join(pytan_root_dir, 'lib')
13  path_adds = [lib_dir]
14
15  for aa in path_adds:
16      if aa not in sys.path:
17          sys.path.append(aa)
18
19
20  # connection info for Tanium Server
21  USERNAME = "Tanium User"
22  PASSWORD = "T@n!um"
23  HOST = "172.16.31.128"
24  PORT = "443"
25
26  # Logging conrols
27  LOGLEVEL = 2
28  DEBUGFORMAT = False
29
30  import tempfile
31
32  import pytan
33  handler = pytan.Handler(
34      username=USERNAME,
35      password=PASSWORD,
36      host=HOST,
37      port=PORT,
38      loglevel=LOGLEVEL,
39      debugformat=DEBUGFORMAT,
40  )
41
42  print handler
43
44  # setup the arguments for the handler method
45  kwargs = {}
46  kwargs["sensors"] = u'Operating System, that contains:Windows, opt:match_all_values, opt:ignore_case
47  kwargs["qtype"] = u'manual'
48
49  # call the handler with the ask method, passing in kwargs for arguments
50  response = handler.ask(**kwargs)
51  import pprint, io
52
53  print ""
54  print "Type of response: ", type(response)
55
56  print ""
57  print "Pretty print of response:"
58  print pprint.pformat(response)
59
60  print ""
61  print "Equivalent Question if it were to be asked in the Tanium Console: "
62  print response['question_object'].query_text
63
```

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

Output from Python Code

```
1    Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2    2015-08-07 19:43:51,504 DEBUG     pytan.handler.QuestionPoller: ID 1295: id resolved to 1295
3    2015-08-07 19:43:51,504 DEBUG     pytan.handler.QuestionPoller: ID 1295: expiration resolved to 2015-
4    2015-08-07 19:43:51,504 DEBUG     pytan.handler.QuestionPoller: ID 1295: query_text resolved to Get C
5    2015-08-07 19:43:51,504 DEBUG     pytan.handler.QuestionPoller: ID 1295: id resolved to 1295
6    2015-08-07 19:43:51,504 DEBUG     pytan.handler.QuestionPoller: ID 1295: Object Info resolved to Ques
7    2015-08-07 19:43:51,508 DEBUG     pytan.handler.QuestionPoller: ID 1295: Progress: Tested: 0, Passed:
8    2015-08-07 19:43:51,508 DEBUG     pytan.handler.QuestionPoller: ID 1295: Timing: Started: 2015-08-07
9    2015-08-07 19:43:51,508 INFO      pytan.handler.QuestionPoller: ID 1295: Progress Changed 0% (0 of 2)
10   2015-08-07 19:43:56,512 DEBUG     pytan.handler.QuestionPoller: ID 1295: Progress: Tested: 0, Passed:
11   2015-08-07 19:43:56,512 DEBUG     pytan.handler.QuestionPoller: ID 1295: Timing: Started: 2015-08-07
12   2015-08-07 19:44:01,520 DEBUG     pytan.handler.QuestionPoller: ID 1295: Progress: Tested: 2, Passed:
13   2015-08-07 19:44:01,520 DEBUG     pytan.handler.QuestionPoller: ID 1295: Timing: Started: 2015-08-07
14   2015-08-07 19:44:01,520 INFO      pytan.handler.QuestionPoller: ID 1295: Progress Changed 100% (2 of
15   2015-08-07 19:44:01,520 INFO      pytan.handler.QuestionPoller: ID 1295: Reached Threshold of 99% (2
16
17   Type of response:  <type 'dict'>
18
19   Pretty print of response:
20   {'poller_object': <pytan.pollers.QuestionPoller object at 0x10a5b98d0>,
21    'poller_success': True,
22    'question_object': <taniumpy.object_types.question.Question object at 0x10a5b9810>,
23    'question_results': <taniumpy.object_types.result_set.ResultSet object at 0x10a615710>}
24
25   Equivalent Question if it were to be asked in the Tanium Console:
26   Get Operating System containing "Windows" from all machines
27
28   CSV Results of response:
29   Operating System
30   [no results]
31   Windows Server 2008 R2 Standard
```

Ask manual 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
import os
import sys
sys.dont_write_bytecode = True

# Determine our script name, script dir
my_file = os.path.abspath(sys.argv[0])
my_dir = os.path.dirname(my_file)

# determine the pytan lib dir and add it to the path
parent_dir = os.path.dirname(my_dir)
pytan_root_dir = os.path.dirname(parent_dir)
lib_dir = os.path.join(pytan_root_dir, 'lib')
path_adds = [lib_dir]

for aa in path_adds:
    if aa not in sys.path:
        sys.path.append(aa)


# connection info for Tanium Server
USERNAME = "Tanium User"
PASSWORD = "T@n!um"
HOST = "172.16.31.128"
PORT = "443"

# Logging conrols
LOGLEVEL = 2
DEBUGFORMAT = False

import tempfile

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']
```

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

Output from Python Code

```
1   Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2   2015-08-07 19:44:01,651 DEBUG    pytan.handler.QuestionPoller: ID 1296: id resolved to 1296
3   2015-08-07 19:44:01,651 DEBUG    pytan.handler.QuestionPoller: ID 1296: expiration resolved to 2015-
4   2015-08-07 19:44:01,651 DEBUG    pytan.handler.QuestionPoller: ID 1296: query_text resolved to Get C
5   2015-08-07 19:44:01,651 DEBUG    pytan.handler.QuestionPoller: ID 1296: id resolved to 1296
6   2015-08-07 19:44:01,651 DEBUG    pytan.handler.QuestionPoller: ID 1296: Object Info resolved to Ques
7   2015-08-07 19:44:01,655 DEBUG    pytan.handler.QuestionPoller: ID 1296: Progress: Tested: 0, Passed:
8   2015-08-07 19:44:01,655 DEBUG    pytan.handler.QuestionPoller: ID 1296: Timing: Started: 2015-08-07
9   2015-08-07 19:44:01,655 INFO     pytan.handler.QuestionPoller: ID 1296: Progress Changed 0% (0 of 2)
10  2015-08-07 19:44:06,659 DEBUG    pytan.handler.QuestionPoller: ID 1296: Progress: Tested: 0, Passed:
11  2015-08-07 19:44:06,659 DEBUG    pytan.handler.QuestionPoller: ID 1296: Timing: Started: 2015-08-07
12  2015-08-07 19:44:11,666 DEBUG    pytan.handler.QuestionPoller: ID 1296: Progress: Tested: 0, Passed:
13  2015-08-07 19:44:11,667 DEBUG    pytan.handler.QuestionPoller: ID 1296: Timing: Started: 2015-08-07
14  2015-08-07 19:44:16,670 DEBUG    pytan.handler.QuestionPoller: ID 1296: Progress: Tested: 0, Passed:
15  2015-08-07 19:44:16,670 DEBUG    pytan.handler.QuestionPoller: ID 1296: Timing: Started: 2015-08-07
16  2015-08-07 19:44:21,677 DEBUG    pytan.handler.QuestionPoller: ID 1296: Progress: Tested: 1, Passed:
17  2015-08-07 19:44:21,677 DEBUG    pytan.handler.QuestionPoller: ID 1296: Timing: Started: 2015-08-07
18  2015-08-07 19:44:21,677 INFO     pytan.handler.QuestionPoller: ID 1296: Progress Changed 50% (1 of 2
19  2015-08-07 19:44:26,687 DEBUG    pytan.handler.QuestionPoller: ID 1296: Progress: Tested: 1, Passed:
20  2015-08-07 19:44:26,687 DEBUG    pytan.handler.QuestionPoller: ID 1296: Timing: Started: 2015-08-07
21  2015-08-07 19:44:31,691 DEBUG    pytan.handler.QuestionPoller: ID 1296: Progress: Tested: 2, Passed:
22  2015-08-07 19:44:31,692 DEBUG    pytan.handler.QuestionPoller: ID 1296: Timing: Started: 2015-08-07
23  2015-08-07 19:44:31,692 INFO     pytan.handler.QuestionPoller: ID 1296: Progress Changed 100% (2 of
```

```
24  2015-08-07 19:44:31,692 INFO     pytan.handler.QuestionPoller: ID 1296: Reached Threshold of 99% (2
25
26  Type of response:  <type 'dict'>
27
28  Pretty print of response:
29  {'poller_object': <pytan.pollers.QuestionPoller object at 0x10a5c9690>,
30   'poller_success': True,
31   'question_object': <taniumpy.object_types.question.Question object at 0x10a5e1610>,
32   'question_results': <taniumpy.object_types.result_set.ResultSet object at 0x10a615510>}
33
34  Equivalent Question if it were to be asked in the Tanium Console:
35  Get Computer Name and Folder Name Search with RegEx Match[Program Files, , No, No, Microsoft.*, test
36
37  CSV Results of response:
38  Computer Name,"Folder Name Search with RegEx Match[Program Files, , No, No, Microsoft.*, test]"
39  Casus-Belli.local,[no results]
40  JTANIUM1.localdomain,"C:\Program Files\Common Files\Microsoft Shared\VS7Debug
41  C:\Program Files\Common Files\Microsoft Shared\ink\ar-SA
42  C:\Program Files\Common Files\Microsoft Shared\ink\ru-RU
43  C:\Program Files\Common Files\Microsoft Shared\ink\fsdefinitions\keypad
44  C:\Program Files\Common Files\Microsoft Shared\ink
45  C:\Program Files\Common Files\Microsoft Shared\ink\sv-SE
46  C:\Program Files\Common Files\Microsoft Shared\ink\uk-UA
47  C:\Program Files\Common Files\Microsoft Shared\ink\sl-SI
48  C:\Program Files\Common Files\Microsoft Shared\ink\hu-HU
49  C:\Program Files\Common Files\Microsoft Shared\ink\zh-TW
50  C:\Program Files\Common Files\Microsoft Shared\ink\zh-CN
51  C:\Program Files\Common Files\Microsoft Shared\ink\fi-FI
52  C:\Program Files\Common Files\Microsoft Shared
53  ..trimmed for brevity..
```

Ask manual 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 or Google Chrome

Example Python Code

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

```
17          sys.path.append(aa)
18
19
20   # connection info for Tanium Server
21   USERNAME = "Tanium User"
22   PASSWORD = "T@n!um"
23   HOST = "172.16.31.128"
24   PORT = "443"
25
26   # Logging conrols
27   LOGLEVEL = 2
28   DEBUGFORMAT = False
29
30   import tempfile
31
32   import pytan
33   handler = pytan.Handler(
34       username=USERNAME,
35       password=PASSWORD,
36       host=HOST,
37       port=PORT,
38       loglevel=LOGLEVEL,
39       debugformat=DEBUGFORMAT,
40   )
41
42   print handler
43
44   # setup the arguments for the handler method
45   kwargs = {}
46   kwargs["question_filters"] = [u'Installed Applications, that regex match:.*Google (Search|Chrome).*'
47   kwargs["sensors"] = [u'Computer Name',
48    u'Last Logged In User',
49    u'Installed Applications, that regex match:.*Google (Search|Chrome).*']
50   kwargs["question_options"] = [u'ignore_case', u'or']
51   kwargs["qtype"] = u'manual'
52
53   # call the handler with the ask method, passing in kwargs for arguments
54   response = handler.ask(**kwargs)
55   import pprint, io
56
57   print ""
58   print "Type of response: ", type(response)
59
60   print ""
61   print "Pretty print of response:"
62   print pprint.pformat(response)
63
64   print ""
65   print "Equivalent Question if it were to be asked in the Tanium Console: "
66   print response['question_object'].query_text
67
68   # create an IO stream to store CSV results to
69   out = io.BytesIO()
70
71   # call the write_csv() method to convert response to CSV and store it in out
72   response['question_results'].write_csv(out, response['question_results'])
73
74   print ""
```

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

Output from Python Code

```
1   Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2   2015-08-07 19:44:31,762 DEBUG    pytan.handler.QuestionPoller: ID 1297: id resolved to 1297
3   2015-08-07 19:44:31,763 DEBUG    pytan.handler.QuestionPoller: ID 1297: expiration resolved to 2015-
4   2015-08-07 19:44:31,763 DEBUG    pytan.handler.QuestionPoller: ID 1297: query_text resolved to Get C
5   2015-08-07 19:44:31,763 DEBUG    pytan.handler.QuestionPoller: ID 1297: id resolved to 1297
6   2015-08-07 19:44:31,763 DEBUG    pytan.handler.QuestionPoller: ID 1297: Object Info resolved to Ques
7   2015-08-07 19:44:31,766 DEBUG    pytan.handler.QuestionPoller: ID 1297: Progress: Tested: 0, Passed:
8   2015-08-07 19:44:31,766 DEBUG    pytan.handler.QuestionPoller: ID 1297: Timing: Started: 2015-08-07
9   2015-08-07 19:44:31,766 INFO     pytan.handler.QuestionPoller: ID 1297: Progress Changed 0% (0 of 2)
10  2015-08-07 19:44:36,774 DEBUG    pytan.handler.QuestionPoller: ID 1297: Progress: Tested: 0, Passed:
11  2015-08-07 19:44:36,774 DEBUG    pytan.handler.QuestionPoller: ID 1297: Timing: Started: 2015-08-07
12  2015-08-07 19:44:41,779 DEBUG    pytan.handler.QuestionPoller: ID 1297: Progress: Tested: 1, Passed:
13  2015-08-07 19:44:41,779 DEBUG    pytan.handler.QuestionPoller: ID 1297: Timing: Started: 2015-08-07
14  2015-08-07 19:44:41,779 INFO     pytan.handler.QuestionPoller: ID 1297: Progress Changed 50% (1 of 2
15  2015-08-07 19:44:46,783 DEBUG    pytan.handler.QuestionPoller: ID 1297: Progress: Tested: 2, Passed:
16  2015-08-07 19:44:46,783 DEBUG    pytan.handler.QuestionPoller: ID 1297: Timing: Started: 2015-08-07
17  2015-08-07 19:44:46,783 INFO     pytan.handler.QuestionPoller: ID 1297: Progress Changed 100% (2 of
18  2015-08-07 19:44:46,783 INFO     pytan.handler.QuestionPoller: ID 1297: Reached Threshold of 99% (2
19
20  Type of response:  <type 'dict'>
21
22  Pretty print of response:
23  {'poller_object': <pytan.pollers.QuestionPoller object at 0x10a615950>,
24   'poller_success': True,
25   'question_object': <taniumpy.object_types.question.Question object at 0x10a5e1bd0>,
26   'question_results': <taniumpy.object_types.result_set.ResultSet object at 0x10a5e1d90>}
27
28  Equivalent Question if it were to be asked in the Tanium Console:
29  Get Computer Name and Last Logged In User and Installed Applications containing "Google (Search|Chro
30
31  CSV Results of response:
32  Computer Name,Last Logged In User,Name,Silent Uninstall String,Uninstallable,Version
33  JTANIUM1.localdomain,Uninitialized - waiting for login,Google Chrome,"""C:\Program Files (x86)\Googl
34  Casus-Belli.local,jolsen,"Google Search
35  Google Search
36  Google Chrome","nothing
37  nothing
38  nothing","Not Uninstallable
39  Not Uninstallable
40  Not Uninstallable","42.0.2311.90
41  41.0.2272.104
42  44.0.2403.130"
```

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
import os
import sys
sys.dont_write_bytecode = True

# Determine our script name, script dir
my_file = os.path.abspath(sys.argv[0])
my_dir = os.path.dirname(my_file)

# determine the pytan lib dir and add it to the path
parent_dir = os.path.dirname(my_dir)
pytan_root_dir = os.path.dirname(parent_dir)
lib_dir = os.path.join(pytan_root_dir, 'lib')
path_adds = [lib_dir]

for aa in path_adds:
    if aa not in sys.path:
        sys.path.append(aa)


# connection info for Tanium Server
USERNAME = "Tanium User"
PASSWORD = "T@n!um"
HOST = "172.16.31.128"
PORT = "443"

# Logging conrols
LOGLEVEL = 2
DEBUGFORMAT = False

import tempfile

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

print handler
```

```
43
44   # setup the arguments for the handler method
45   kwargs = {}
46   kwargs["question_filter_defs"] = [{u'filter': {u'not_flag': 0,
47                   u'operator': u'RegexMatch',
48                   u'value': u'.*Windows.*'},
49      u'name': u'Operating System'}]
50   kwargs["sensor_defs"] = [u'Computer Name',
51      {u'filter': {u'not_flag': 0,
52                   u'operator': u'RegexMatch',
53                   u'value': u'.*Shared.*'},
54      u'name': u'Folder Name Search with RegEx Match',
55      u'options': {u'ignore_case_flag': 0,
56                   u'max_age_seconds': 3600,
57                   u'value_type': u'string'},
58      u'params': {u'dirname': u'Program Files'}}]
59   kwargs["question_option_defs"] = {u'and_flag': 0, u'ignore_case_flag': 0, u'max_age_seconds': 3600}
60   kwargs["qtype"] = u'_manual'
61
62   # call the handler with the ask method, passing in kwargs for arguments
63   response = handler.ask(**kwargs)
64   import pprint, io
65
66   print ""
67   print "Type of response: ", type(response)
68
69   print ""
70   print "Pretty print of response:"
71   print pprint.pformat(response)
72
73   print ""
74   print "Equivalent Question if it were to be asked in the Tanium Console: "
75   print response['question_object'].query_text
76
77   # create an IO stream to store CSV results to
78   out = io.BytesIO()
79
80   # call the write_csv() method to convert response to CSV and store it in out
81   response['question_results'].write_csv(out, response['question_results'])
82
83   print ""
84   print "CSV Results of response: "
85   out = out.getvalue()
86   if len(out.splitlines()) > 15:
87       out = out.splitlines()[0:15]
88       out.append('..trimmed for brevity..')
89       out = '\n'.join(out)
90   print out
```

Output from Python Code

```
1   Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2   2015-08-07 19:44:46,869 DEBUG     pytan.handler.QuestionPoller: ID 1298: id resolved to 1298
3   2015-08-07 19:44:46,869 DEBUG     pytan.handler.QuestionPoller: ID 1298: expiration resolved to 2015-
4   2015-08-07 19:44:46,869 DEBUG     pytan.handler.QuestionPoller: ID 1298: query_text resolved to Get C
5   2015-08-07 19:44:46,869 DEBUG     pytan.handler.QuestionPoller: ID 1298: id resolved to 1298
6   2015-08-07 19:44:46,869 DEBUG     pytan.handler.QuestionPoller: ID 1298: Object Info resolved to Ques
```

```
7    2015-08-07 19:44:46,872 DEBUG      pytan.handler.QuestionPoller: ID 1298: Progress: Tested: 0, Passed:
8    2015-08-07 19:44:46,873 DEBUG      pytan.handler.QuestionPoller: ID 1298: Timing: Started: 2015-08-07
9    2015-08-07 19:44:46,873 INFO       pytan.handler.QuestionPoller: ID 1298: Progress Changed 0% (0 of 2)
10   2015-08-07 19:44:51,877 DEBUG      pytan.handler.QuestionPoller: ID 1298: Progress: Tested: 1, Passed:
11   2015-08-07 19:44:51,877 DEBUG      pytan.handler.QuestionPoller: ID 1298: Timing: Started: 2015-08-07
12   2015-08-07 19:44:51,877 INFO       pytan.handler.QuestionPoller: ID 1298: Progress Changed 50% (1 of 2
13   2015-08-07 19:44:56,881 DEBUG      pytan.handler.QuestionPoller: ID 1298: Progress: Tested: 1, Passed:
14   2015-08-07 19:44:56,881 DEBUG      pytan.handler.QuestionPoller: ID 1298: Timing: Started: 2015-08-07
15   2015-08-07 19:45:01,885 DEBUG      pytan.handler.QuestionPoller: ID 1298: Progress: Tested: 1, Passed:
16   2015-08-07 19:45:01,885 DEBUG      pytan.handler.QuestionPoller: ID 1298: Timing: Started: 2015-08-07
17   2015-08-07 19:45:06,890 DEBUG      pytan.handler.QuestionPoller: ID 1298: Progress: Tested: 2, Passed:
18   2015-08-07 19:45:06,890 DEBUG      pytan.handler.QuestionPoller: ID 1298: Timing: Started: 2015-08-07
19   2015-08-07 19:45:06,890 INFO       pytan.handler.QuestionPoller: ID 1298: Progress Changed 100% (2 of
20   2015-08-07 19:45:06,890 INFO       pytan.handler.QuestionPoller: ID 1298: Reached Threshold of 99% (2
21
22   Type of response:  <type 'dict'>
23
24   Pretty print of response:
25   {'poller_object': <pytan.pollers.QuestionPoller object at 0x10a5c9c90>,
26    'poller_success': True,
27    'question_object': <taniumpy.object_types.question.Question object at 0x10a5b98d0>,
28    'question_results': <taniumpy.object_types.result_set.ResultSet object at 0x10a5e1410>}
29
30   Equivalent Question if it were to be asked in the Tanium Console:
31   Get Computer Name and Folder Name Search with RegEx Match[Program Files, , No, No] containing "Share
32
33   CSV Results of response:
34   Computer Name,"Folder Name Search with RegEx Match[Program Files, , No, No]"
35   JTANIUM1.localdomain,"C:\Program Files\Common Files\Microsoft Shared\VS7Debug
36   C:\Program Files\Common Files\Microsoft Shared\ink\ar-SA
37   C:\Program Files\Common Files\Microsoft Shared\ink\ru-RU
38   C:\Program Files\Common Files\Microsoft Shared\ink\fsdefinitions\keypad
39   C:\Program Files\Common Files\Microsoft Shared\ink
40   C:\Program Files\Common Files\Microsoft Shared\ink\sv-SE
41   C:\Program Files\Common Files\Microsoft Shared\ink\uk-UA
42   C:\Program Files\Common Files\Microsoft Shared\ink\sl-SI
43   C:\Program Files\Common Files\Microsoft Shared\ink\hu-HU
44   C:\Program Files\Common Files\Microsoft Shared\ink\zh-TW
45   C:\Program Files\Common Files\Microsoft Shared\ink\zh-CN
46   C:\Program Files\Common Files\Microsoft Shared\ink\fi-FI
47   C:\Program Files\Common Files\Microsoft Shared
48   C:\Program Files\Common Files\Microsoft Shared\ink\da-DK
49   ..trimmed for brevity..
```

PyTan API Invalid Question Examples

Invalid ask manual question sensor help

Have ask_manual() return the help for sensors

Example Python Code

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

```
4
5    # Determine our script name, script dir
6    my_file = os.path.abspath(sys.argv[0])
7    my_dir = os.path.dirname(my_file)
8
9    # determine the pytan lib dir and add it to the path
10   parent_dir = os.path.dirname(my_dir)
11   pytan_root_dir = os.path.dirname(parent_dir)
12   lib_dir = os.path.join(pytan_root_dir, 'lib')
13   path_adds = [lib_dir]
14
15   for aa in path_adds:
16       if aa not in sys.path:
17           sys.path.append(aa)
18
19
20   # connection info for Tanium Server
21   USERNAME = "Tanium User"
22   PASSWORD = "T@n!um"
23   HOST = "172.16.31.128"
24   PORT = "443"
25
26   # Logging conrols
27   LOGLEVEL = 2
28   DEBUGFORMAT = False
29
30   import tempfile
31
32   import pytan
33   handler = pytan.Handler(
34       username=USERNAME,
35       password=PASSWORD,
36       host=HOST,
37       port=PORT,
38       loglevel=LOGLEVEL,
39       debugformat=DEBUGFORMAT,
40   )
41
42   print handler
43
44   # setup the arguments for the handler method
45   kwargs = {}
46   kwargs["qtype"] = u'manual'
47   kwargs["sensors_help"] = True
48
49
50   # call the handler with the ask method, passing in kwargs for arguments
51   # this should throw an exception: pytan.exceptions.PytanHelp
52   import traceback
53   try:
54       handler.ask(**kwargs)
55   except Exception as e:
56       traceback.print_exc(file=sys.stdout)
```

Output from Python Code

```
1   Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2   Traceback (most recent call last):
3     File "<string>", line 55, in <module>
4     File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 130, in ask
5       result = getattr(self, q_obj_map['handler'])(**kwargs)
6     File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 271, in ask_manual
7       raise pytan.exceptions.PytanHelp(pytan.help.help_sensors())
8   PytanHelp:
9   Sensors Help
10  ============
11
12  Supplying sensors controls what columns will be showed when you ask a
13  question.
14
15  A sensor string is a human string that describes, at a minimum, a sensor.
16  It can also optionally define a selector for the sensor, parameters for
17  the sensor, a filter for the sensor, and options for the filter for the
18  sensor. Sensors can be provided as a string or a list of strings.
19
20  Examples for basic sensors
21  --------------------------------
22
23  Supplying a single sensor:
24
25      'Computer Name'
26
27  Supplying two sensors in a list of strings:
28
29      ['Computer Name', 'IP Route Details']
30
31  Supplying multiple sensors with selectors (name is the default
32  selector if none is supplied):
33
34      [
35          'Computer Name',
36          'name:Computer Name',
37          'id:1',
38          'hash:123456789',
39      ]
40
41  Sensor Parameters
42  -----------------
43
44  Supplying parameters to a sensor can control the arguments that are
45  supplied to a sensor, if that sensor takes any arguments.
46
47  Sensor parameters must be surrounded with curly braces '{}',
48  and must have a key and value specified that is separated by
49  an equals '='. Multiple parameters must be seperated by
50  a comma ','. The key should match up to a valid parameter key
51  for the sensor in question.
52
53  If a parameter is supplied and the sensor doesn't have a
54  corresponding key name, it will be ignored. If the sensor has
55  parameters and a parameter is NOT supplied then one of two
56  paths will be taken:
57
58      * if the parameter does not require a default value, the
```

```
59        parameter is left blank and not supplied.
60        * if the parameter does require a value (pulldowns, for
61        example), a default value is derived (for pulldowns,
62        the first value available as a pulldown entry is used).
63
64    Examples for sensors with parameters
65    ------------------------------------
66
67    Supplying a single sensor with a single parameter 'dirname':
68
69        'Sensor With Params{dirname=Program Files}'
70
71    Supplying a single sensor with two parameters, 'param1' and
72    'param2':
73
74        'Sensor With Params{param1=value1,param2=value2}'
75
76    Sensor Filters
77    --------------
78
79    Supplying a filter to a sensor controls what data will be shown in
80    those columns (sensors) you've provided.
81
82    Sensor filters can be supplied by adding ', that FILTER:VALUE',
83    where FILTER is a valid filter string, and VALUE is the string
84    that you want FILTER to match on.
85
86    See filter help for a list of all possible FILTER strings.
87
88    See options help for a list of options that can control how
89    the filter works.
90
91    Examples for sensors with filters
92    ---------------------------------
93
94    Supplying a sensor with a filter that limits the results to only
95    show column data that matches the regular expression
96    '.*Windows.*' (Tanium does a case insensitive match by default):
97
98        'Computer Name, that contains:Windows'
99
100   Supplying a sensor with a filter that limits the results to only
101   show column data that matches the regular expression
102   'Microsoft.*':
103
104       'Computer Name, that starts with:Microsoft'
105
106   Supply a sensor with a filter that limits the results to only
107   show column data that has a version greater or equal to
108   '39.0.0.0'. Since this sensor uses Version as its default result
109   type, there is no need to change the value type using filter
110   options.
111
112       'Installed Application Version' \
113       '{Application Name=Google Chrome}, that =>:39.0.0.0'
114
115   Sensor Options
116   --------------
```

```
117
118  Supplying options to a sensor can change how the filter for
119  that sensor works.
120
121  Sensor options can be supplied by adding ', opt:OPTION' or
122  ', opt:OPTION:VALUE' for those options that require values,
123  where OPTION is a valid option string, and VALUE is the
124  appropriate value required by accordant OPTION.
125
126  See options help for a list of options that can control how
127  the filter works.
128
129  Examples for sensors with options
130  ---------------------------------
131
132  Supplying a sensor with an option that forces tanium to
133  re-fetch any cached column data that is older than 1 minute:
134
135      'Computer Name, opt:max_data_age:60'
136
137  Supplying a sensor with filter and an option that causes
138  Tanium to match case for the filter value:
139
140      'Computer Name, that contains:Windows, opt:match_case'
141
142  Supplying a sensor with a filter and an option that causes
143  Tanium to match all values supplied:
144
145      'Computer Name, that contains:Windows, opt:match_all_values'
146
147  Supplying a sensor with a filter and a set of options that
148  causes Tanium to recognize the value type as String (which is
149  the default type for most sensors), re-fetch data older than
150  10 minutes, match any values, and match case:
151
152      'Computer Name', that contains:Windows, ' \
153      opt:value_type:string, opt:max_data_age:600, ' \
154      'opt:match_any_value, opt:match_case'
```

Invalid ask manual question filter help

Have ask_manual() return the help for filters

Example Python Code

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

```
11  pytan_root_dir = os.path.dirname(parent_dir)
12  lib_dir = os.path.join(pytan_root_dir, 'lib')
13  path_adds = [lib_dir]
14
15  for aa in path_adds:
16      if aa not in sys.path:
17          sys.path.append(aa)
18
19
20  # connection info for Tanium Server
21  USERNAME = "Tanium User"
22  PASSWORD = "T@n!um"
23  HOST = "172.16.31.128"
24  PORT = "443"
25
26  # Logging conrols
27  LOGLEVEL = 2
28  DEBUGFORMAT = False
29
30  import tempfile
31
32  import pytan
33  handler = pytan.Handler(
34      username=USERNAME,
35      password=PASSWORD,
36      host=HOST,
37      port=PORT,
38      loglevel=LOGLEVEL,
39      debugformat=DEBUGFORMAT,
40  )
41
42  print handler
43
44  # setup the arguments for the handler method
45  kwargs = {}
46  kwargs["filters_help"] = True
47  kwargs["qtype"] = u'manual'
48
49
50  # call the handler with the ask method, passing in kwargs for arguments
51  # this should throw an exception: pytan.exceptions.PytanHelp
52  import traceback
53  try:
54      handler.ask(**kwargs)
55  except Exception as e:
56      traceback.print_exc(file=sys.stdout)
```

Output from Python Code

```
1  Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2  Traceback (most recent call last):
3    File "<string>", line 55, in <module>
4    File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 130, in ask
5      result = getattr(self, q_obj_map['handler'])(**kwargs)
6    File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 274, in ask_manual
7      raise pytan.exceptions.PytanHelp(pytan.help.help_filters())
8  PytanHelp:
```

```
 9  | Filters Help
10  | ============
11  |
12  | Filters are used generously throughout pytan. When used as part of a
13  | sensor string, they control what data is shown for the columns that
14  | the sensor returns. When filters are used for whole question filters,
15  | they control what rows will be returned. They are used by Groups to
16  | define group membership, deploy actions to determine which machines
17  | should have the action deployed to it, and more.
18  |
19  | A filter string is a human string that describes, a sensor followed
20  | by ', that FILTER:VALUE', where FILTER is a valid filter string,
21  | and VALUE is the string that you want FILTER to match on.
22  |
23  | Valid Filters
24  | -------------
25  |
26  |     '<'
27  |         Help: Filter for less than VALUE
28  |         Example: "Sensor1, that <:VALUE"
29  |
30  |     'less'
31  |         Help: Filter for less than VALUE
32  |         Example: "Sensor1, that less:VALUE"
33  |
34  |     'lt'
35  |         Help: Filter for less than VALUE
36  |         Example: "Sensor1, that lt:VALUE"
37  |
38  |     'less than'
39  |         Help: Filter for less than VALUE
40  |         Example: "Sensor1, that less than:VALUE"
41  |
42  |     '!<'
43  |         Help: Filter for not less than VALUE
44  |         Example: "Sensor1, that !<:VALUE"
45  |
46  |     'notless'
47  |         Help: Filter for not less than VALUE
48  |         Example: "Sensor1, that notless:VALUE"
49  |
50  |     'not less'
51  |         Help: Filter for not less than VALUE
52  |         Example: "Sensor1, that not less:VALUE"
53  |
54  |     'not less than'
55  |         Help: Filter for not less than VALUE
56  |         Example: "Sensor1, that not less than:VALUE"
57  |
58  |     '<='
59  |         Help: Filter for less than or equal to VALUE
60  |         Example: "Sensor1, that <=:VALUE"
61  |
62  |     'less equal'
63  |         Help: Filter for less than or equal to VALUE
64  |         Example: "Sensor1, that less equal:VALUE"
65  |
66  |     'lessequal'
```

```
67          Help: Filter for less than or equal to VALUE
68          Example: "Sensor1, that lessequal:VALUE"
69
70      'le'
71          Help: Filter for less than or equal to VALUE
72          Example: "Sensor1, that le:VALUE"
73
74      '!<='
75          Help: Filter for not less than or equal to VALUE
76          Example: "Sensor1, that !<=:VALUE"
77
78      'not less equal'
79          Help: Filter for not less than or equal to VALUE
80          Example: "Sensor1, that not less equal:VALUE"
81
82      'not lessequal'
83          Help: Filter for not less than or equal to VALUE
84          Example: "Sensor1, that not lessequal:VALUE"
85
86      '>'
87          Help: Filter for greater than VALUE
88          Example: "Sensor1, that >:VALUE"
89
90      'greater'
91          Help: Filter for greater than VALUE
92          Example: "Sensor1, that greater:VALUE"
93
94      'gt'
95          Help: Filter for greater than VALUE
96          Example: "Sensor1, that gt:VALUE"
97
98      'greater than'
99          Help: Filter for greater than VALUE
100         Example: "Sensor1, that greater than:VALUE"
101
102     '!>'
103         Help: Filter for not greater than VALUE
104         Example: "Sensor1, that !>:VALUE"
105
106     'not greater'
107         Help: Filter for not greater than VALUE
108         Example: "Sensor1, that not greater:VALUE"
109
110     'notgreater'
111         Help: Filter for not greater than VALUE
112         Example: "Sensor1, that notgreater:VALUE"
113
114     'not greater than'
115         Help: Filter for not greater than VALUE
116         Example: "Sensor1, that not greater than:VALUE"
117
118     '=>'
119         Help: Filter for greater than or equal to VALUE
120         Example: "Sensor1, that =>:VALUE"
121
122     'greater equal'
123         Help: Filter for greater than or equal to VALUE
124         Example: "Sensor1, that greater equal:VALUE"
```

```
125
126        'greaterequal'
127            Help: Filter for greater than or equal to VALUE
128            Example: "Sensor1, that greaterequal:VALUE"
129
130        'ge'
131            Help: Filter for greater than or equal to VALUE
132            Example: "Sensor1, that ge:VALUE"
133
134        '!=>'
135            Help: Filter for not greater than VALUE
136            Example: "Sensor1, that !=>:VALUE"
137
138        'not greater equal'
139            Help: Filter for not greater than VALUE
140            Example: "Sensor1, that not greater equal:VALUE"
141
142        'notgreaterequal'
143            Help: Filter for not greater than VALUE
144            Example: "Sensor1, that notgreaterequal:VALUE"
145
146        '='
147            Help: Filter for equals to VALUE
148            Example: "Sensor1, that =:VALUE"
149
150        'equal'
151            Help: Filter for equals to VALUE
152            Example: "Sensor1, that equal:VALUE"
153
154        'equals'
155            Help: Filter for equals to VALUE
156            Example: "Sensor1, that equals:VALUE"
157
158        'eq'
159            Help: Filter for equals to VALUE
160            Example: "Sensor1, that eq:VALUE"
161
162        '!='
163            Help: Filter for not equals to VALUE
164            Example: "Sensor1, that !=:VALUE"
165
166        'not equal'
167            Help: Filter for not equals to VALUE
168            Example: "Sensor1, that not equal:VALUE"
169
170        'notequal'
171            Help: Filter for not equals to VALUE
172            Example: "Sensor1, that notequal:VALUE"
173
174        'not equals'
175            Help: Filter for not equals to VALUE
176            Example: "Sensor1, that not equals:VALUE"
177
178        'notequals'
179            Help: Filter for not equals to VALUE
180            Example: "Sensor1, that notequals:VALUE"
181
182        'ne'
```

```
183          Help: Filter for not equals to VALUE
184          Example: "Sensor1, that ne:VALUE"
185
186      'contains'
187          Help: Filter for contains VALUE (adds .* before and after VALUE)
188          Example: "Sensor1, that contains:VALUE"
189
190      'does not contain'
191          Help: Filter for does not contain VALUE (adds .* before and after VALUE)
192          Example: "Sensor1, that does not contain:VALUE"
193
194      'doesnotcontain'
195          Help: Filter for does not contain VALUE (adds .* before and after VALUE)
196          Example: "Sensor1, that doesnotcontain:VALUE"
197
198      'not contains'
199          Help: Filter for does not contain VALUE (adds .* before and after VALUE)
200          Example: "Sensor1, that not contains:VALUE"
201
202      'notcontains'
203          Help: Filter for does not contain VALUE (adds .* before and after VALUE)
204          Example: "Sensor1, that notcontains:VALUE"
205
206      'starts with'
207          Help: Filter for starts with VALUE (adds .* after VALUE)
208          Example: "Sensor1, that starts with:VALUE"
209
210      'startswith'
211          Help: Filter for starts with VALUE (adds .* after VALUE)
212          Example: "Sensor1, that startswith:VALUE"
213
214      'does not start with'
215          Help: Filter for does not start with VALUE (adds .* after VALUE)
216          Example: "Sensor1, that does not start with:VALUE"
217
218      'doesnotstartwith'
219          Help: Filter for does not start with VALUE (adds .* after VALUE)
220          Example: "Sensor1, that doesnotstartwith:VALUE"
221
222      'not starts with'
223          Help: Filter for does not start with VALUE (adds .* after VALUE)
224          Example: "Sensor1, that not starts with:VALUE"
225
226      'notstartswith'
227          Help: Filter for does not start with VALUE (adds .* after VALUE)
228          Example: "Sensor1, that notstartswith:VALUE"
229
230      'ends with'
231          Help: Filter for ends with VALUE (adds .* before VALUE)
232          Example: "Sensor1, that ends with:VALUE"
233
234      'endswith'
235          Help: Filter for ends with VALUE (adds .* before VALUE)
236          Example: "Sensor1, that endswith:VALUE"
237
238      'does not end with'
239          Help: Filter for does bit end with VALUE (adds .* before VALUE)
240          Example: "Sensor1, that does not end with:VALUE"
```

```
241
242        'doesnotendwith'
243            Help: Filter for does bit end with VALUE (adds .* before VALUE)
244            Example: "Sensor1, that doesnotendwith:VALUE"
245
246        'not ends with'
247            Help: Filter for does bit end with VALUE (adds .* before VALUE)
248            Example: "Sensor1, that not ends with:VALUE"
249
250        'notstartswith'
251            Help: Filter for does bit end with VALUE (adds .* before VALUE)
252            Example: "Sensor1, that notstartswith:VALUE"
253
254        'is not'
255            Help: Filter for non regular expression match for VALUE
256            Example: "Sensor1, that is not:VALUE"
257
258        'not regex'
259            Help: Filter for non regular expression match for VALUE
260            Example: "Sensor1, that not regex:VALUE"
261
262        'notregex'
263            Help: Filter for non regular expression match for VALUE
264            Example: "Sensor1, that notregex:VALUE"
265
266        'not regex match'
267            Help: Filter for non regular expression match for VALUE
268            Example: "Sensor1, that not regex match:VALUE"
269
270        'notregexmatch'
271            Help: Filter for non regular expression match for VALUE
272            Example: "Sensor1, that notregexmatch:VALUE"
273
274        'nre'
275            Help: Filter for non regular expression match for VALUE
276            Example: "Sensor1, that nre:VALUE"
277
278        'is'
279            Help: Filter for regular expression match for VALUE
280            Example: "Sensor1, that is:VALUE"
281
282        'regex'
283            Help: Filter for regular expression match for VALUE
284            Example: "Sensor1, that regex:VALUE"
285
286        'regex match'
287            Help: Filter for regular expression match for VALUE
288            Example: "Sensor1, that regex match:VALUE"
289
290        'regexmatch'
291            Help: Filter for regular expression match for VALUE
292            Example: "Sensor1, that regexmatch:VALUE"
293
294        're'
295            Help: Filter for regular expression match for VALUE
296            Example: "Sensor1, that re:VALUE"
```

Invalid ask manual question option help

Have ask_manual() return the help for options

Example Python Code

```python
import os
import sys
sys.dont_write_bytecode = True

# Determine our script name, script dir
my_file = os.path.abspath(sys.argv[0])
my_dir = os.path.dirname(my_file)

# determine the pytan lib dir and add it to the path
parent_dir = os.path.dirname(my_dir)
pytan_root_dir = os.path.dirname(parent_dir)
lib_dir = os.path.join(pytan_root_dir, 'lib')
path_adds = [lib_dir]

for aa in path_adds:
    if aa not in sys.path:
        sys.path.append(aa)


# connection info for Tanium Server
USERNAME = "Tanium User"
PASSWORD = "T@n!um"
HOST = "172.16.31.128"
PORT = "443"

# Logging conrols
LOGLEVEL = 2
DEBUGFORMAT = False

import tempfile

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'


# call the handler with the ask method, passing in kwargs for arguments
# this should throw an exception: pytan.exceptions.PytanHelp
import traceback
```

```
53   try:
54       handler.ask(**kwargs)
55   except Exception as e:
56       traceback.print_exc(file=sys.stdout)
```

Output from Python Code

```
1    Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2    Traceback (most recent call last):
3      File "<string>", line 55, in <module>
4      File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 130, in ask
5        result = getattr(self, q_obj_map['handler'])(**kwargs)
6      File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 277, in ask_manual
7        raise pytan.exceptions.PytanHelp(pytan.help.help_options())
8    PytanHelp:
9    Options Help
10   ============
11
12   Options are used for controlling how filters act. When options are
13   used as part of a sensor string, they change how the filters
14   supplied as part of that sensor operate. When options are used for
15   whole question options, they change how all of the question filters
16   operate.
17
18   When options are supplied for a sensor string, they must be
19   supplied as ', opt:OPTION' or ', opt:OPTION:VALUE' for options
20   that require a value.
21
22   When options are supplied for question options, they must be
23   supplied as 'OPTION' or 'OPTION:VALUE' for options that require
24   a value.
25
26   Options can be used on 'filter' or 'group', where 'group' pertains
27   to group filters or question filters. All 'filter' options are also
28   applicable to 'group' for question options.
29
30   Valid Options
31   -------------
32
33       'ignore_case'
34           Help: Make the filter do a case insensitive match
35           Usable on: filter
36           Example for sensor: "Sensor1, opt:ignore_case"
37           Example for question: "ignore_case"
38
39       'match_case'
40           Help: Make the filter do a case sensitive match
41           Usable on: filter
42           Example for sensor: "Sensor1, opt:match_case"
43           Example for question: "match_case"
44
45       'match_any_value'
46           Help: Make the filter match any value
47           Usable on: filter
48           Example for sensor: "Sensor1, opt:match_any_value"
49           Example for question: "match_any_value"
50
```

```
51        'match_all_values'
52            Help: Make the filter match all values
53            Usable on: filter
54            Example for sensor: "Sensor1, opt:match_all_values"
55            Example for question: "match_all_values"
56
57        'max_data_age'
58            Help: Re-fetch cached values older than N seconds
59            Usable on: filter
60            VALUE description and type: seconds, <type 'int'>
61            Example for sensor: "Sensor1, opt:max_data_age:seconds"
62            Example for question: "max_data_age:seconds"
63
64        'value_type'
65            Help: Make the filter consider the value type as VALUE_TYPE
66            Usable on: filter
67            VALUE description and type: value_type, <type 'str'>
68            Example for sensor: "Sensor1, opt:value_type:value_type"
69            Example for question: "value_type:value_type"
70
71        'and'
72            Help: Use 'and' for all of the filters supplied
73            Usable on: group
74            Example for sensor: "Sensor1, opt:and"
75            Example for question: "and"
76
77        'or'
78            Help: Use 'or' for all of the filters supplied
79            Usable on: group
80            Example for sensor: "Sensor1, opt:or"
81            Example for question: "or"
```

Invalid ask manual question sensor

Ask a question using a sensor that does not exist

Example Python Code

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

```
18
19
20    # connection info for Tanium Server
21    USERNAME = "Tanium User"
22    PASSWORD = "T@n!um"
23    HOST = "172.16.31.128"
24    PORT = "443"
25
26    # Logging conrols
27    LOGLEVEL = 2
28    DEBUGFORMAT = False
29
30    import tempfile
31
32    import pytan
33    handler = pytan.Handler(
34        username=USERNAME,
35        password=PASSWORD,
36        host=HOST,
37        port=PORT,
38        loglevel=LOGLEVEL,
39        debugformat=DEBUGFORMAT,
40    )
41
42    print handler
43
44    # setup the arguments for the handler method
45    kwargs = {}
46    kwargs["sensors"] = u'Dweedle Dee and Dum'
47    kwargs["qtype"] = u'manual'
48
49
50    # call the handler with the ask method, passing in kwargs for arguments
51    # this should throw an exception: pytan.exceptions.HandlerError
52    import traceback
53    try:
54        handler.ask(**kwargs)
55    except Exception as e:
56        traceback.print_exc(file=sys.stdout)
```

Output from Python Code

```
1    Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2    Traceback (most recent call last):
3      File "<string>", line 55, in <module>
4      File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 130, in ask
5        result = getattr(self, q_obj_map['handler'])(**kwargs)
6      File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 305, in ask_manual
7        **kwargs
8      File "/Users/jolsen/gh/pytan/lib/pytan/utils.py", line 2699, in wrap
9        ret = f(*args, **kwargs)
10     File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 1882, in _ask_manual
11       sensor_defs = self._get_sensor_defs(sensor_defs)
12     File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 1418, in _get_sensor_defs
13       d['sensor_obj'] = self.get('sensor', **def_search)[0]
14     File "/Users/jolsen/gh/pytan/lib/pytan/utils.py", line 2699, in wrap
15       ret = f(*args, **kwargs)
```

```
16      File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 1250, in get
17        return self._get_multi(obj_map, **kwargs)
18      File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 1363, in _get_multi
19        found = self._find(api_obj_multi, **kwargs)
20      File "/Users/jolsen/gh/pytan/lib/pytan/utils.py", line 2699, in wrap
21        ret = f(*args, **kwargs)
22      File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 1327, in _find
23        raise pytan.exceptions.HandlerError(err(search_str))
24    HandlerError: No results found searching for Sensor, name: u'Dweedle Dee and Dum'!!
```

Invalid ask manual question filter

Ask a question using an invalid filter.

Example Python Code

```python
1   import os
2   import sys
3   sys.dont_write_bytecode = True
4
5   # Determine our script name, script dir
6   my_file = os.path.abspath(sys.argv[0])
7   my_dir = os.path.dirname(my_file)
8
9   # determine the pytan lib dir and add it to the path
10  parent_dir = os.path.dirname(my_dir)
11  pytan_root_dir = os.path.dirname(parent_dir)
12  lib_dir = os.path.join(pytan_root_dir, 'lib')
13  path_adds = [lib_dir]
14
15  for aa in path_adds:
16      if aa not in sys.path:
17          sys.path.append(aa)
18
19
20  # connection info for Tanium Server
21  USERNAME = "Tanium User"
22  PASSWORD = "T@n!um"
23  HOST = "172.16.31.128"
24  PORT = "443"
25
26  # Logging conrols
27  LOGLEVEL = 2
28  DEBUGFORMAT = False
29
30  import tempfile
31
32  import pytan
33  handler = pytan.Handler(
34      username=USERNAME,
35      password=PASSWORD,
36      host=HOST,
37      port=PORT,
38      loglevel=LOGLEVEL,
39      debugformat=DEBUGFORMAT,
```

```
40   )
41
42   print handler
43
44   # setup the arguments for the handler method
45   kwargs = {}
46   kwargs["sensors"] = u'Computer name, that does not meet:little'
47   kwargs["qtype"] = u'manual'
48
49
50   # call the handler with the ask method, passing in kwargs for arguments
51   # this should throw an exception: pytan.exceptions.HumanParserError
52   import traceback
53   try:
54       handler.ask(**kwargs)
55   except Exception as e:
56       traceback.print_exc(file=sys.stdout)
```

Output from Python Code

```
1    Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2    Traceback (most recent call last):
3      File "<string>", line 55, in <module>
4      File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 130, in ask
5        result = getattr(self, q_obj_map['handler'])(**kwargs)
6      File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 297, in ask_manual
7        sensor_defs = pytan.utils.dehumanize_sensors(sensors)
8      File "/Users/jolsen/gh/pytan/lib/pytan/utils.py", line 1478, in dehumanize_sensors
9        s, parsed_filter = extract_filter(s)
10     File "/Users/jolsen/gh/pytan/lib/pytan/utils.py", line 1846, in extract_filter
11       raise pytan.exceptions.HumanParserError(err(split_filter[1]))
12   HumanParserError: Filter u' does not meet:little' is not a valid filter!
```

Invalid ask manual question paramater too many

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

Example Python Code

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

```
16        if aa not in sys.path:
17            sys.path.append(aa)
18
19
20    # connection info for Tanium Server
21    USERNAME = "Tanium User"
22    PASSWORD = "T@n!um"
23    HOST = "172.16.31.128"
24    PORT = "443"
25
26    # Logging conrols
27    LOGLEVEL = 2
28    DEBUGFORMAT = False
29
30    import tempfile
31
32    import pytan
33    handler = pytan.Handler(
34        username=USERNAME,
35        password=PASSWORD,
36        host=HOST,
37        port=PORT,
38        loglevel=LOGLEVEL,
39        debugformat=DEBUGFORMAT,
40    )
41
42    print handler
43
44    # setup the arguments for the handler method
45    kwargs = {}
46    kwargs["sensors"] = u'Folder Name Search with RegEx Match{dirname=Program Files,regex=.*}{}'
47    kwargs["qtype"] = u'manual'
48
49
50    # call the handler with the ask method, passing in kwargs for arguments
51    # this should throw an exception: pytan.exceptions.HumanParserError
52    import traceback
53    try:
54        handler.ask(**kwargs)
55    except Exception as e:
56        traceback.print_exc(file=sys.stdout)
```

Output from Python Code

```
1    Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2    Traceback (most recent call last):
3      File "<string>", line 55, in <module>
4      File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 130, in ask
5        result = getattr(self, q_obj_map['handler'])(**kwargs)
6      File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 297, in ask_manual
7        sensor_defs = pytan.utils.dehumanize_sensors(sensors)
8      File "/Users/jolsen/gh/pytan/lib/pytan/utils.py", line 1476, in dehumanize_sensors
9        s, parsed_params = extract_params(s)
10      File "/Users/jolsen/gh/pytan/lib/pytan/utils.py", line 1646, in extract_params
11        raise pytan.exceptions.HumanParserError(err(s))
12    HumanParserError: More than one parameter ({}) passed in u'Folder Name Search with RegEx Match{dirna
```

Invalid ask manual question option

Ask a question using an invalid option.

Example Python Code

```python
import os
import sys
sys.dont_write_bytecode = True

# Determine our script name, script dir
my_file = os.path.abspath(sys.argv[0])
my_dir = os.path.dirname(my_file)

# determine the pytan lib dir and add it to the path
parent_dir = os.path.dirname(my_dir)
pytan_root_dir = os.path.dirname(parent_dir)
lib_dir = os.path.join(pytan_root_dir, 'lib')
path_adds = [lib_dir]

for aa in path_adds:
    if aa not in sys.path:
        sys.path.append(aa)


# connection info for Tanium Server
USERNAME = "Tanium User"
PASSWORD = "T@n!um"
HOST = "172.16.31.128"
PORT = "443"

# Logging conrols
LOGLEVEL = 2
DEBUGFORMAT = False

import tempfile

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'


# call the handler with the ask method, passing in kwargs for arguments
# this should throw an exception: pytan.exceptions.HumanParserError
import traceback
```

```
53  try:
54      handler.ask(**kwargs)
55  except Exception as e:
56      traceback.print_exc(file=sys.stdout)
```

Output from Python Code

```
1   Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2   Traceback (most recent call last):
3     File "<string>", line 55, in <module>
4     File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 130, in ask
5       result = getattr(self, q_obj_map['handler'])(**kwargs)
6     File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 297, in ask_manual
7       sensor_defs = pytan.utils.dehumanize_sensors(sensors)
8     File "/Users/jolsen/gh/pytan/lib/pytan/utils.py", line 1477, in dehumanize_sensors
9       s, parsed_options = extract_options(s)
10    File "/Users/jolsen/gh/pytan/lib/pytan/utils.py", line 1721, in extract_options
11      parsed_options = map_options(parsed_options, ['filter'])
12    File "/Users/jolsen/gh/pytan/lib/pytan/utils.py", line 1751, in map_options
13      raise pytan.exceptions.HumanParserError(err(option))
14  HumanParserError: Option u'bad' is not a valid option!
```

Invalid ask manual question parameter split

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

Example Python Code

```python
1   import os
2   import sys
3   sys.dont_write_bytecode = True
4
5   # Determine our script name, script dir
6   my_file = os.path.abspath(sys.argv[0])
7   my_dir = os.path.dirname(my_file)
8
9   # determine the pytan lib dir and add it to the path
10  parent_dir = os.path.dirname(my_dir)
11  pytan_root_dir = os.path.dirname(parent_dir)
12  lib_dir = os.path.join(pytan_root_dir, 'lib')
13  path_adds = [lib_dir]
14
15  for aa in path_adds:
16      if aa not in sys.path:
17          sys.path.append(aa)
18
19
20  # connection info for Tanium Server
21  USERNAME = "Tanium User"
22  PASSWORD = "T@n!um"
23  HOST = "172.16.31.128"
24  PORT = "443"
25
26  # Logging conrols
```

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

Output from Python Code

```
1    Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2    Traceback (most recent call last):
3      File "<string>", line 55, in <module>
4      File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 130, in ask
5        result = getattr(self, q_obj_map['handler'])(**kwargs)
6      File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 297, in ask_manual
7        sensor_defs = pytan.utils.dehumanize_sensors(sensors)
8      File "/Users/jolsen/gh/pytan/lib/pytan/utils.py", line 1476, in dehumanize_sensors
9        s, parsed_params = extract_params(s)
10     File "/Users/jolsen/gh/pytan/lib/pytan/utils.py", line 1664, in extract_params
11       raise pytan.exceptions.HumanParserError(err(sp, pytan.constants.PARAM_KEY_SPLIT))
12   HumanParserError: Parameter Dweedle missing key/value seperator (=)
```

PyTan API Valid Get Object Examples

Get action by id

Get an action by id

Example Python Code

```python
1   import os
2   import sys
3   sys.dont_write_bytecode = True
4
5   # Determine our script name, script dir
6   my_file = os.path.abspath(sys.argv[0])
7   my_dir = os.path.dirname(my_file)
8
9   # determine the pytan lib dir and add it to the path
10  parent_dir = os.path.dirname(my_dir)
11  pytan_root_dir = os.path.dirname(parent_dir)
12  lib_dir = os.path.join(pytan_root_dir, 'lib')
13  path_adds = [lib_dir]
14
15  for aa in path_adds:
16      if aa not in sys.path:
17          sys.path.append(aa)
18
19
20  # connection info for Tanium Server
21  USERNAME = "Tanium User"
22  PASSWORD = "T@n!um"
23  HOST = "172.16.31.128"
24  PORT = "443"
25
26  # Logging conrols
27  LOGLEVEL = 2
28  DEBUGFORMAT = False
29
30  import tempfile
31
32  import pytan
33  handler = pytan.Handler(
34      username=USERNAME,
35      password=PASSWORD,
36      host=HOST,
37      port=PORT,
38      loglevel=LOGLEVEL,
39      debugformat=DEBUGFORMAT,
40  )
41
42  print handler
43
44  # setup the arguments for the handler method
45  kwargs = {}
46  kwargs["objtype"] = u'action'
47  kwargs["id"] = 1
48
49  # call the handler with the get method, passing in kwargs for arguments
50  response = handler.get(**kwargs)
51
52  print ""
53  print "Type of response: ", type(response)
54
55  print ""
56  print "print of response:"
57  print response
58
```

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

Output from Python Code

```
1   Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2
3   Type of response:  <class 'taniumpy.object_types.action_list.ActionList'>
4
5   print of response:
6   ActionList, len: 1
7
8   length of response (number of objects returned):
9   1
10
11  print the first object returned in JSON format:
12  {
13    "_type": "action",
14    "action_group": {
15      "_type": "group",
16      "id": 0,
17      "name": "Default"
18    },
19    "approver": {
20      "_type": "user",
21      "id": 1,
22      "name": "Jim Olsen"
23    },
24    "comment": "Distribute Tanium Standard Utilities",
25    "creation_time": "2015-08-07T13:22:26",
26    "distribute_seconds": 3200,
27  ..trimmed for brevity..
```

Get question by id

Get a question by id

Example Python Code

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

```
5    # Determine our script name, script dir
6    my_file = os.path.abspath(sys.argv[0])
7    my_dir = os.path.dirname(my_file)
8
9    # determine the pytan lib dir and add it to the path
10   parent_dir = os.path.dirname(my_dir)
11   pytan_root_dir = os.path.dirname(parent_dir)
12   lib_dir = os.path.join(pytan_root_dir, 'lib')
13   path_adds = [lib_dir]
14
15   for aa in path_adds:
16       if aa not in sys.path:
17           sys.path.append(aa)
18
19
20   # connection info for Tanium Server
21   USERNAME = "Tanium User"
22   PASSWORD = "T@n!um"
23   HOST = "172.16.31.128"
24   PORT = "443"
25
26   # Logging conrols
27   LOGLEVEL = 2
28   DEBUGFORMAT = False
29
30   import tempfile
31
32   import pytan
33   handler = pytan.Handler(
34       username=USERNAME,
35       password=PASSWORD,
36       host=HOST,
37       port=PORT,
38       loglevel=LOGLEVEL,
39       debugformat=DEBUGFORMAT,
40   )
41
42   print handler
43
44   # setup the arguments for the handler method
45   kwargs = {}
46   kwargs["objtype"] = u'question'
47   kwargs["id"] = 1
48
49   # call the handler with the get method, passing in kwargs for arguments
50   response = handler.get(**kwargs)
51
52   print ""
53   print "Type of response: ", type(response)
54
55   print ""
56   print "print of response:"
57   print response
58
59   print ""
60   print "length of response (number of objects returned): "
61   print len(response)
62
```

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

Output from Python Code

```
1   Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2
3   Type of response:  <class 'taniumpy.object_types.question_list.QuestionList'>
4
5   print of response:
6   QuestionList, len: 1
7
8   length of response (number of objects returned):
9   1
10
11  print the first object returned in JSON format:
12  {
13    "_type": "question",
14    "action_tracking_flag": 0,
15    "context_group": {
16      "_type": "group",
17      "id": 0
18    },
19    "expiration": "2015-08-07T13:31:47",
20    "expire_seconds": 0,
21    "force_computer_id_flag": 1,
22    "hidden_flag": 0,
23    "id": 1,
24    "management_rights_group": {
25      "_type": "group",
26      "id": 0
27  ..trimmed for brevity..
```

Get saved question by names

Get two saved questions by name

Example Python Code

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

```
9    # determine the pytan lib dir and add it to the path
10   parent_dir = os.path.dirname(my_dir)
11   pytan_root_dir = os.path.dirname(parent_dir)
12   lib_dir = os.path.join(pytan_root_dir, 'lib')
13   path_adds = [lib_dir]
14
15   for aa in path_adds:
16       if aa not in sys.path:
17           sys.path.append(aa)
18
19
20   # connection info for Tanium Server
21   USERNAME = "Tanium User"
22   PASSWORD = "T@n!um"
23   HOST = "172.16.31.128"
24   PORT = "443"
25
26   # Logging conrols
27   LOGLEVEL = 2
28   DEBUGFORMAT = False
29
30   import tempfile
31
32   import pytan
33   handler = pytan.Handler(
34       username=USERNAME,
35       password=PASSWORD,
36       host=HOST,
37       port=PORT,
38       loglevel=LOGLEVEL,
39       debugformat=DEBUGFORMAT,
40   )
41
42   print handler
43
44   # setup the arguments for the handler method
45   kwargs = {}
46   kwargs["objtype"] = u'saved_question'
47   kwargs["name"] = [u'Installed Applications', u'Computer Name']
48
49   # call the handler with the get method, passing in kwargs for arguments
50   response = handler.get(**kwargs)
51
52   print ""
53   print "Type of response: ", type(response)
54
55   print ""
56   print "print of response:"
57   print response
58
59   print ""
60   print "length of response (number of objects returned): "
61   print len(response)
62
63   print ""
64   print "print the first object returned in JSON format:"
65   out = response.to_json(response[0])
66   if len(out.splitlines()) > 15:
```

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

Output from Python Code

```
1    Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2
3    Type of response:  <class 'taniumpy.object_types.saved_question_list.SavedQuestionList'>
4
5    print of response:
6    SavedQuestionList, len: 2
7
8    length of response (number of objects returned):
9    2
10
11   print the first object returned in JSON format:
12   {
13     "_type": "saved_question",
14     "action_tracking_flag": 0,
15     "archive_enabled_flag": 0,
16     "archive_owner": {
17       "_type": "user"
18     },
19     "expire_seconds": 600,
20     "hidden_flag": 0,
21     "id": 64,
22     "issue_seconds": 120,
23     "issue_seconds_never_flag": 0,
24     "keep_seconds": 0,
25     "metadata": {
26       "_type": "metadata",
27   ..trimmed for brevity..
```

Get userrole by id

Get a user role by id.

Example Python Code

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

```
13    path_adds = [lib_dir]

14

15    for aa in path_adds:
16        if aa not in sys.path:
17            sys.path.append(aa)

18

19

20    # connection info for Tanium Server
21    USERNAME = "Tanium User"
22    PASSWORD = "T@n!um"
23    HOST = "172.16.31.128"
24    PORT = "443"

25

26    # Logging conrols
27    LOGLEVEL = 2
28    DEBUGFORMAT = False

29

30    import tempfile

31

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

41

42    print handler

43

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

48

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

51

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

54

55    print ""
56    print "print of response:"
57    print response

58

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

62

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

70
```

```
71    print out
```

Output from Python Code

```
1    Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2
3    Type of response:   <class 'taniumpy.object_types.user_role_list.UserRoleList'>
4
5    print of response:
6    UserRoleList, len: 1
7
8    length of response (number of objects returned):
9    1
10
11   print the first object returned in JSON format:
12   {
13     "_type": "role",
14     "description": "Administrators can perform all functions in the system, including creating other u
15     "id": 1,
16     "name": "Administrator",
17     "permissions": {
18       "_type": "permissions",
19       "permission": [
20         "admin",
21         "sensor_read",
22         "sensor_write",
23         "question_read",
24         "question_write",
25         "action_read",
26         "action_write",
27   ..trimmed for brevity..
```

Get leader clients

Get all clients that are Leader status

Example Python Code

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

```
17          sys.path.append(aa)
18
19
20   # connection info for Tanium Server
21   USERNAME = "Tanium User"
22   PASSWORD = "T@n!um"
23   HOST = "172.16.31.128"
24   PORT = "443"
25
26   # Logging conrols
27   LOGLEVEL = 2
28   DEBUGFORMAT = False
29
30   import tempfile
31
32   import pytan
33   handler = pytan.Handler(
34       username=USERNAME,
35       password=PASSWORD,
36       host=HOST,
37       port=PORT,
38       loglevel=LOGLEVEL,
39       debugformat=DEBUGFORMAT,
40   )
41
42   print handler
43
44   # setup the arguments for the handler method
45   kwargs = {}
46   kwargs["objtype"] = u'client'
47   kwargs["status"] = u'Leader'
48
49   # call the handler with the get method, passing in kwargs for arguments
50   response = handler.get(**kwargs)
51
52   print ""
53   print "Type of response: ", type(response)
54
55   print ""
56   print "print of response:"
57   print response
58
59   print ""
60   print "length of response (number of objects returned): "
61   print len(response)
62
63   print ""
64   print "print the first object returned in JSON format:"
65   out = response.to_json(response[0])
66   if len(out.splitlines()) > 15:
67       out = out.splitlines()[0:15]
68       out.append('..trimmed for brevity..')
69       out = '\n'.join(out)
70
71   print out
```

Output from Python Code

```
1   Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2
3   Type of response:  <class 'taniumpy.object_types.system_status_list.SystemStatusList'>
4
5   print of response:
6   SystemStatusList, len: 1
7
8   length of response (number of objects returned):
9   1
10
11  print the first object returned in JSON format:
12  {
13    "_type": "client_status",
14    "cache_row_id": 1,
15    "computer_id": "3741604154",
16    "full_version": "6.0.314.1195",
17    "host_name": "JTANIUM1.localdomain",
18    "ipaddress_client": "172.16.31.128",
19    "ipaddress_server": "172.16.31.128",
20    "last_registration": "2015-08-07T19:45:00",
21    "port_number": 17473,
22    "protocol_version": 314,
23    "public_key_valid": 1,
24    "receive_state": "Previous Only",
25    "send_state": "Backward Only",
26    "status": "Leader"
27  ..trimmed for brevity..
```

Get setting by name

Get a system setting by name

Example Python Code

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

```
22   PASSWORD = "T@n!um"
23   HOST = "172.16.31.128"
24   PORT = "443"
25
26   # Logging conrols
27   LOGLEVEL = 2
28   DEBUGFORMAT = False
29
30   import tempfile
31
32   import pytan
33   handler = pytan.Handler(
34       username=USERNAME,
35       password=PASSWORD,
36       host=HOST,
37       port=PORT,
38       loglevel=LOGLEVEL,
39       debugformat=DEBUGFORMAT,
40   )
41
42   print handler
43
44   # setup the arguments for the handler method
45   kwargs = {}
46   kwargs["objtype"] = u'setting'
47   kwargs["name"] = u'control_address'
48
49   # call the handler with the get method, passing in kwargs for arguments
50   response = handler.get(**kwargs)
51
52   print ""
53   print "Type of response: ", type(response)
54
55   print ""
56   print "print of response:"
57   print response
58
59   print ""
60   print "length of response (number of objects returned): "
61   print len(response)
62
63   print ""
64   print "print the first object returned in JSON format:"
65   out = response.to_json(response[0])
66   if len(out.splitlines()) > 15:
67       out = out.splitlines()[0:15]
68       out.append('..trimmed for brevity..')
69       out = '\n'.join(out)
70
71   print out
```

Output from Python Code

```
1   Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2
3   Type of response:  <class 'taniumpy.object_types.system_setting_list.SystemSettingList'>
4
```

```
5   print of response:
6   SystemSettingList, len: 1
7
8   length of response (number of objects returned):
9   1
10
11  print the first object returned in JSON format:
12  {
13    "_type": "system_setting",
14    "default_value": "512:17473:127.0.0.1",
15    "hidden_flag": 0,
16    "id": 58,
17    "name": "control_address",
18    "read_only_flag": 0,
19    "setting_type": "Server",
20    "value": "512:17473:127.0.0.1",
21    "value_type": "Text"
22  }
```

Get user by name

Get a user by name

Example Python Code

```python
1   import os
2   import sys
3   sys.dont_write_bytecode = True
4
5   # Determine our script name, script dir
6   my_file = os.path.abspath(sys.argv[0])
7   my_dir = os.path.dirname(my_file)
8
9   # determine the pytan lib dir and add it to the path
10  parent_dir = os.path.dirname(my_dir)
11  pytan_root_dir = os.path.dirname(parent_dir)
12  lib_dir = os.path.join(pytan_root_dir, 'lib')
13  path_adds = [lib_dir]
14
15  for aa in path_adds:
16      if aa not in sys.path:
17          sys.path.append(aa)
18
19
20  # connection info for Tanium Server
21  USERNAME = "Tanium User"
22  PASSWORD = "T@n!um"
23  HOST = "172.16.31.128"
24  PORT = "443"
25
26  # Logging conrols
27  LOGLEVEL = 2
28  DEBUGFORMAT = False
29
30  import tempfile
```

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

Output from Python Code

```
1   Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2
3   Type of response:  <class 'taniumpy.object_types.user_list.UserList'>
4
5   print of response:
6   UserList, len: 1
7
8   length of response (number of objects returned):
9   1
10
11  print the first object returned in JSON format:
12  {
13    "_type": "user",
```

```
14    "deleted_flag": 0,
15    "group_id": 0,
16    "id": 2,
17    "last_login": "2015-08-07T19:45:07",
18    "local_admin_flag": 1,
19    "name": "Tanium User",
20    "permissions": {
21      "_type": "permissions",
22      "permission": [
23        "admin",
24        "sensor_read",
25        "sensor_write",
26        "question_read",
27  ..trimmed for brevity..
```

Get sensor by id

Get a sensor by id

Example Python Code

```python
1   import os
2   import sys
3   sys.dont_write_bytecode = True
4
5   # Determine our script name, script dir
6   my_file = os.path.abspath(sys.argv[0])
7   my_dir = os.path.dirname(my_file)
8
9   # determine the pytan lib dir and add it to the path
10  parent_dir = os.path.dirname(my_dir)
11  pytan_root_dir = os.path.dirname(parent_dir)
12  lib_dir = os.path.join(pytan_root_dir, 'lib')
13  path_adds = [lib_dir]
14
15  for aa in path_adds:
16      if aa not in sys.path:
17          sys.path.append(aa)
18
19
20  # connection info for Tanium Server
21  USERNAME = "Tanium User"
22  PASSWORD = "T@n!um"
23  HOST = "172.16.31.128"
24  PORT = "443"
25
26  # Logging conrols
27  LOGLEVEL = 2
28  DEBUGFORMAT = False
29
30  import tempfile
31
32  import pytan
33  handler = pytan.Handler(
34      username=USERNAME,
```

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

Output from Python Code

```
1   Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2
3   Type of response:  <class 'taniumpy.object_types.sensor_list.SensorList'>
4
5   print of response:
6   SensorList, len: 1
7
8   length of response (number of objects returned):
9   1
10
11  print the first object returned in JSON format:
12  {
13    "_type": "sensor",
14    "category": "Reserved",
15    "description": "The recorded state of each action a client has taken recently in the form of id:st
16    "exclude_from_parse_flag": 1,
17    "hash": 1792443391,
```

```
18     "hidden_flag": 0,
19     "id": 1,
20     "ignore_case_flag": 1,
21     "max_age_seconds": 3600,
22     "name": "Action Statuses",
23     "queries": {
24       "_type": "queries",
25       "query": [
26           {
27   ..trimmed for brevity..
```

Get sensor by mixed

Get multiple sensors by id, name, and hash

Example Python Code

```python
1   import os
2   import sys
3   sys.dont_write_bytecode = True
4
5   # Determine our script name, script dir
6   my_file = os.path.abspath(sys.argv[0])
7   my_dir = os.path.dirname(my_file)
8
9   # determine the pytan lib dir and add it to the path
10  parent_dir = os.path.dirname(my_dir)
11  pytan_root_dir = os.path.dirname(parent_dir)
12  lib_dir = os.path.join(pytan_root_dir, 'lib')
13  path_adds = [lib_dir]
14
15  for aa in path_adds:
16      if aa not in sys.path:
17          sys.path.append(aa)
18
19
20  # connection info for Tanium Server
21  USERNAME = "Tanium User"
22  PASSWORD = "T@n!um"
23  HOST = "172.16.31.128"
24  PORT = "443"
25
26  # Logging conrols
27  LOGLEVEL = 2
28  DEBUGFORMAT = False
29
30  import tempfile
31
32  import pytan
33  handler = pytan.Handler(
34      username=USERNAME,
35      password=PASSWORD,
36      host=HOST,
37      port=PORT,
38      loglevel=LOGLEVEL,
```

```
39        debugformat=DEBUGFORMAT,
40    )
41
42    print handler
43
44    # setup the arguments for the handler method
45    kwargs = {}
46    kwargs["objtype"] = u'sensor'
47    kwargs["hash"] = [u'322086833']
48    kwargs["name"] = [u'Computer Name']
49    kwargs["id"] = [1, 2]
50
51    # call the handler with the get method, passing in kwargs for arguments
52    response = handler.get(**kwargs)
53
54    print ""
55    print "Type of response: ", type(response)
56
57    print ""
58    print "print of response:"
59    print response
60
61    print ""
62    print "length of response (number of objects returned): "
63    print len(response)
64
65    print ""
66    print "print the first object returned in JSON format:"
67    out = response.to_json(response[0])
68    if len(out.splitlines()) > 15:
69        out = out.splitlines()[0:15]
70        out.append('..trimmed for brevity..')
71        out = '\n'.join(out)
72
73    print out
```

Output from Python Code

```
1    Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2
3    Type of response:  <class 'taniumpy.object_types.sensor_list.SensorList'>
4
5    print of response:
6    SensorList, len: 4
7
8    length of response (number of objects returned):
9    4
10
11   print the first object returned in JSON format:
12   {
13     "_type": "sensor",
14     "category": "Reserved",
15     "description": "The recorded state of each download a client has made recently in the form of hash
16     "exclude_from_parse_flag": 0,
17     "hash": 322086833,
18     "hidden_flag": 0,
19     "id": 4,
```

```
20    "ignore_case_flag": 1,
21    "max_age_seconds": 900,
22    "name": "Download Statuses",
23    "queries": {
24      "_type": "queries",
25      "query": [
26        {
27  ..trimmed for brevity..
```

Get whitelisted url by id

Get a whitelisted url by id

Example Python Code

```python
1   import os
2   import sys
3   sys.dont_write_bytecode = True
4
5   # Determine our script name, script dir
6   my_file = os.path.abspath(sys.argv[0])
7   my_dir = os.path.dirname(my_file)
8
9   # determine the pytan lib dir and add it to the path
10  parent_dir = os.path.dirname(my_dir)
11  pytan_root_dir = os.path.dirname(parent_dir)
12  lib_dir = os.path.join(pytan_root_dir, 'lib')
13  path_adds = [lib_dir]
14
15  for aa in path_adds:
16      if aa not in sys.path:
17          sys.path.append(aa)
18
19
20  # connection info for Tanium Server
21  USERNAME = "Tanium User"
22  PASSWORD = "T@n!um"
23  HOST = "172.16.31.128"
24  PORT = "443"
25
26  # Logging conrols
27  LOGLEVEL = 2
28  DEBUGFORMAT = False
29
30  import tempfile
31
32  import pytan
33  handler = pytan.Handler(
34      username=USERNAME,
35      password=PASSWORD,
36      host=HOST,
37      port=PORT,
38      loglevel=LOGLEVEL,
39      debugformat=DEBUGFORMAT,
40  )
```

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

Output from Python Code

```
1    Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2
3    Type of response:  <class 'taniumpy.object_types.white_listed_url_list.WhiteListedUrlList'>
4
5    print of response:
6    WhiteListedUrlList, len: 1
7
8    length of response (number of objects returned):
9    1
10
11   print the first object returned in JSON format:
12   {
13     "_type": "white_listed_url",
14     "download_seconds": 86400,
15     "id": 1,
16     "url_regex": "test1"
17   }
```

Get group by name

Get a group by name

Example Python Code

```python
import os
import sys
sys.dont_write_bytecode = True

# Determine our script name, script dir
my_file = os.path.abspath(sys.argv[0])
my_dir = os.path.dirname(my_file)

# determine the pytan lib dir and add it to the path
parent_dir = os.path.dirname(my_dir)
pytan_root_dir = os.path.dirname(parent_dir)
lib_dir = os.path.join(pytan_root_dir, 'lib')
path_adds = [lib_dir]

for aa in path_adds:
    if aa not in sys.path:
        sys.path.append(aa)


# connection info for Tanium Server
USERNAME = "Tanium User"
PASSWORD = "T@n!um"
HOST = "172.16.31.128"
PORT = "443"

# Logging conrols
LOGLEVEL = 2
DEBUGFORMAT = False

import tempfile

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
```

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

Output from Python Code

```
1   Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2
3   Type of response:  <class 'taniumpy.object_types.group_list.GroupList'>
4
5   print of response:
6   GroupList, len: 1
7
8   length of response (number of objects returned):
9   1
10
11  print the first object returned in JSON format:
12  {
13    "_type": "group",
14    "and_flag": 0,
15    "deleted_flag": 0,
16    "filters": {
17      "_type": "filters",
18      "filter": []
19    },
20    "id": 64,
21    "name": "All Computers",
22    "not_flag": 0,
23    "sub_groups": {
24      "_type": "groups",
25      "group": []
26    },
27  ..trimmed for brevity..
```

Get sensor by hash

Get a sensor by hash

Example Python Code

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

```
4
5   # Determine our script name, script dir
6   my_file = os.path.abspath(sys.argv[0])
7   my_dir = os.path.dirname(my_file)
8
9   # determine the pytan lib dir and add it to the path
10  parent_dir = os.path.dirname(my_dir)
11  pytan_root_dir = os.path.dirname(parent_dir)
12  lib_dir = os.path.join(pytan_root_dir, 'lib')
13  path_adds = [lib_dir]
14
15  for aa in path_adds:
16      if aa not in sys.path:
17          sys.path.append(aa)
18
19
20  # connection info for Tanium Server
21  USERNAME = "Tanium User"
22  PASSWORD = "T@n!um"
23  HOST = "172.16.31.128"
24  PORT = "443"
25
26  # Logging conrols
27  LOGLEVEL = 2
28  DEBUGFORMAT = False
29
30  import tempfile
31
32  import pytan
33  handler = pytan.Handler(
34      username=USERNAME,
35      password=PASSWORD,
36      host=HOST,
37      port=PORT,
38      loglevel=LOGLEVEL,
39      debugformat=DEBUGFORMAT,
40  )
41
42  print handler
43
44  # setup the arguments for the handler method
45  kwargs = {}
46  kwargs["objtype"] = u'sensor'
47  kwargs["hash"] = u'322086833'
48
49  # call the handler with the get method, passing in kwargs for arguments
50  response = handler.get(**kwargs)
51
52  print ""
53  print "Type of response: ", type(response)
54
55  print ""
56  print "print of response:"
57  print response
58
59  print ""
60  print "length of response (number of objects returned): "
61  print len(response)
```

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

Output from Python Code

```
1   Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2
3   Type of response:  <class 'taniumpy.object_types.sensor_list.SensorList'>
4
5   print of response:
6   SensorList, len: 1
7
8   length of response (number of objects returned):
9   1
10
11  print the first object returned in JSON format:
12  {
13    "_type": "sensor",
14    "category": "Reserved",
15    "description": "The recorded state of each download a client has made recently in the form of hash
16    "exclude_from_parse_flag": 0,
17    "hash": 322086833,
18    "hidden_flag": 0,
19    "id": 4,
20    "ignore_case_flag": 1,
21    "max_age_seconds": 900,
22    "name": "Download Statuses",
23    "queries": {
24      "_type": "queries",
25      "query": [
26        {
27  ..trimmed for brevity..
```

Get package by name

Get a package by name

Example Python Code

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

```
8
9    # determine the pytan lib dir and add it to the path
10   parent_dir = os.path.dirname(my_dir)
11   pytan_root_dir = os.path.dirname(parent_dir)
12   lib_dir = os.path.join(pytan_root_dir, 'lib')
13   path_adds = [lib_dir]
14
15   for aa in path_adds:
16       if aa not in sys.path:
17           sys.path.append(aa)
18
19
20   # connection info for Tanium Server
21   USERNAME = "Tanium User"
22   PASSWORD = "T@n!um"
23   HOST = "172.16.31.128"
24   PORT = "443"
25
26   # Logging conrols
27   LOGLEVEL = 2
28   DEBUGFORMAT = False
29
30   import tempfile
31
32   import pytan
33   handler = pytan.Handler(
34       username=USERNAME,
35       password=PASSWORD,
36       host=HOST,
37       port=PORT,
38       loglevel=LOGLEVEL,
39       debugformat=DEBUGFORMAT,
40   )
41
42   print handler
43
44   # setup the arguments for the handler method
45   kwargs = {}
46   kwargs["objtype"] = u'package'
47   kwargs["name"] = u'Distribute Tanium Standard Utilities'
48
49   # call the handler with the get method, passing in kwargs for arguments
50   response = handler.get(**kwargs)
51
52   print ""
53   print "Type of response: ", type(response)
54
55   print ""
56   print "print of response:"
57   print response
58
59   print ""
60   print "length of response (number of objects returned): "
61   print len(response)
62
63   print ""
64   print "print the first object returned in JSON format:"
65   out = response.to_json(response[0])
```

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

Output from Python Code

```
1   Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2
3   Type of response:  <class 'taniumpy.object_types.package_spec_list.PackageSpecList'>
4
5   print of response:
6   PackageSpecList, len: 1
7
8   length of response (number of objects returned):
9   1
10
11  print the first object returned in JSON format:
12  {
13    "_type": "package_spec",
14    "available_time": "2015-08-07T13:23:22",
15    "command": "cmd /c cscript install-standard-utils.vbs \"Tools\\StdUtils\"",
16    "command_timeout": 2700,
17    "creation_time": "2015-08-07T13:22:19",
18    "deleted_flag": 0,
19    "display_name": "Distribute Tanium Standard Utilities",
20    "expire_seconds": 3300,
21    "files": {
22      "_type": "package_files",
23      "file": [
24        {
25          "_type": "file",
26          "bytes_downloaded": 0,
27  ..trimmed for brevity..
```

Get sensor by names

Get multiple sensors by name

Example Python Code

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

```python
12  lib_dir = os.path.join(pytan_root_dir, 'lib')
13  path_adds = [lib_dir]
14
15  for aa in path_adds:
16      if aa not in sys.path:
17          sys.path.append(aa)
18
19
20  # connection info for Tanium Server
21  USERNAME = "Tanium User"
22  PASSWORD = "T@n!um"
23  HOST = "172.16.31.128"
24  PORT = "443"
25
26  # Logging conrols
27  LOGLEVEL = 2
28  DEBUGFORMAT = False
29
30  import tempfile
31
32  import pytan
33  handler = pytan.Handler(
34      username=USERNAME,
35      password=PASSWORD,
36      host=HOST,
37      port=PORT,
38      loglevel=LOGLEVEL,
39      debugformat=DEBUGFORMAT,
40  )
41
42  print handler
43
44  # setup the arguments for the handler method
45  kwargs = {}
46  kwargs["objtype"] = u'sensor'
47  kwargs["name"] = [u'Computer Name', u'Action Statuses']
48
49  # call the handler with the get method, passing in kwargs for arguments
50  response = handler.get(**kwargs)
51
52  print ""
53  print "Type of response: ", type(response)
54
55  print ""
56  print "print of response:"
57  print response
58
59  print ""
60  print "length of response (number of objects returned): "
61  print len(response)
62
63  print ""
64  print "print the first object returned in JSON format:"
65  out = response.to_json(response[0])
66  if len(out.splitlines()) > 15:
67      out = out.splitlines()[0:15]
68      out.append('..trimmed for brevity..')
69      out = '\n'.join(out)
```

```
70
71  print out
```

Output from Python Code

```
1   Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2
3   Type of response:  <class 'taniumpy.object_types.sensor_list.SensorList'>
4
5   print of response:
6   SensorList, len: 2
7
8   length of response (number of objects returned):
9   2
10
11  print the first object returned in JSON format:
12  {
13    "_type": "sensor",
14    "category": "Reserved",
15    "description": "The assigned name of the client machine.\nExample: workstation-1.company.com",
16    "exclude_from_parse_flag": 0,
17    "hash": 3409330187,
18    "hidden_flag": 0,
19    "id": 3,
20    "ignore_case_flag": 1,
21    "max_age_seconds": 86400,
22    "name": "Computer Name",
23    "queries": {
24      "_type": "queries",
25      "query": [
26        {
27  ..trimmed for brevity..
```

Get saved question by name

Get saved question by name

Example Python Code

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

```
16      if aa not in sys.path:
17          sys.path.append(aa)
18
19
20  # connection info for Tanium Server
21  USERNAME = "Tanium User"
22  PASSWORD = "T@n!um"
23  HOST = "172.16.31.128"
24  PORT = "443"
25
26  # Logging conrols
27  LOGLEVEL = 2
28  DEBUGFORMAT = False
29
30  import tempfile
31
32  import pytan
33  handler = pytan.Handler(
34      username=USERNAME,
35      password=PASSWORD,
36      host=HOST,
37      port=PORT,
38      loglevel=LOGLEVEL,
39      debugformat=DEBUGFORMAT,
40  )
41
42  print handler
43
44  # setup the arguments for the handler method
45  kwargs = {}
46  kwargs["objtype"] = u'saved_question'
47  kwargs["name"] = u'Installed Applications'
48
49  # call the handler with the get method, passing in kwargs for arguments
50  response = handler.get(**kwargs)
51
52  print ""
53  print "Type of response: ", type(response)
54
55  print ""
56  print "print of response:"
57  print response
58
59  print ""
60  print "length of response (number of objects returned): "
61  print len(response)
62
63  print ""
64  print "print the first object returned in JSON format:"
65  out = response.to_json(response[0])
66  if len(out.splitlines()) > 15:
67      out = out.splitlines()[0:15]
68      out.append('..trimmed for brevity..')
69      out = '\n'.join(out)
70
71  print out
```

Output from Python Code

```
1   Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2
3   Type of response:  <class 'taniumpy.object_types.saved_question_list.SavedQuestionList'>
4
5   print of response:
6   SavedQuestionList, len: 1
7
8   length of response (number of objects returned):
9   1
10
11  print the first object returned in JSON format:
12  {
13    "_type": "saved_question",
14    "action_tracking_flag": 0,
15    "archive_enabled_flag": 0,
16    "archive_owner": {
17      "_type": "user"
18    },
19    "expire_seconds": 600,
20    "hidden_flag": 0,
21    "id": 64,
22    "issue_seconds": 120,
23    "issue_seconds_never_flag": 0,
24    "keep_seconds": 0,
25    "metadata": {
26      "_type": "metadata",
27  ..trimmed for brevity..
```

Get user by id

Get a user by id

Example Python Code

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

```
21   USERNAME = "Tanium User"
22   PASSWORD = "T@n!um"
23   HOST = "172.16.31.128"
24   PORT = "443"
25
26   # Logging conrols
27   LOGLEVEL = 2
28   DEBUGFORMAT = False
29
30   import tempfile
31
32   import pytan
33   handler = pytan.Handler(
34       username=USERNAME,
35       password=PASSWORD,
36       host=HOST,
37       port=PORT,
38       loglevel=LOGLEVEL,
39       debugformat=DEBUGFORMAT,
40   )
41
42   print handler
43
44   # setup the arguments for the handler method
45   kwargs = {}
46   kwargs["objtype"] = u'user'
47   kwargs["id"] = 1
48
49   # call the handler with the get method, passing in kwargs for arguments
50   response = handler.get(**kwargs)
51
52   print ""
53   print "Type of response: ", type(response)
54
55   print ""
56   print "print of response:"
57   print response
58
59   print ""
60   print "length of response (number of objects returned): "
61   print len(response)
62
63   print ""
64   print "print the first object returned in JSON format:"
65   out = response.to_json(response[0])
66   if len(out.splitlines()) > 15:
67       out = out.splitlines()[0:15]
68       out.append('..trimmed for brevity..')
69       out = '\n'.join(out)
70
71   print out
```

Output from Python Code

```
1   Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2
3   Type of response:  <class 'taniumpy.object_types.user_list.UserList'>
```

```
4
5   print of response:
6   UserList, len: 1
7
8   length of response (number of objects returned):
9   1
10
11  print the first object returned in JSON format:
12  {
13    "_type": "user",
14    "deleted_flag": 0,
15    "group_id": 0,
16    "id": 1,
17    "last_login": "2015-08-07T13:21:59",
18    "local_admin_flag": -1,
19    "name": "Jim Olsen",
20    "permissions": {
21      "_type": "permissions",
22      "permission": [
23        "admin",
24        "sensor_read",
25        "sensor_write",
26        "question_read",
27  ..trimmed for brevity..
```

Get sensor by name

Get a sensor by name

Example Python Code

```python
1   import os
2   import sys
3   sys.dont_write_bytecode = True
4
5   # Determine our script name, script dir
6   my_file = os.path.abspath(sys.argv[0])
7   my_dir = os.path.dirname(my_file)
8
9   # determine the pytan lib dir and add it to the path
10  parent_dir = os.path.dirname(my_dir)
11  pytan_root_dir = os.path.dirname(parent_dir)
12  lib_dir = os.path.join(pytan_root_dir, 'lib')
13  path_adds = [lib_dir]
14
15  for aa in path_adds:
16      if aa not in sys.path:
17          sys.path.append(aa)
18
19
20  # connection info for Tanium Server
21  USERNAME = "Tanium User"
22  PASSWORD = "T@n!um"
23  HOST = "172.16.31.128"
24  PORT = "443"
```

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

Output from Python Code

```
1   Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2
3   Type of response:  <class 'taniumpy.object_types.sensor_list.SensorList'>
4
5   print of response:
6   SensorList, len: 1
7
```

```
8   length of response (number of objects returned):
9   1
10
11  print the first object returned in JSON format:
12  {
13    "_type": "sensor",
14    "category": "Reserved",
15    "description": "The assigned name of the client machine.\nExample: workstation-1.company.com",
16    "exclude_from_parse_flag": 0,
17    "hash": 3409330187,
18    "hidden_flag": 0,
19    "id": 3,
20    "ignore_case_flag": 1,
21    "max_age_seconds": 86400,
22    "name": "Computer Name",
23    "queries": {
24      "_type": "queries",
25      "query": [
26          {
27  ..trimmed for brevity..
```

Get saved action by name

Get a saved action by name

Example Python Code

```python
1   import os
2   import sys
3   sys.dont_write_bytecode = True
4
5   # Determine our script name, script dir
6   my_file = os.path.abspath(sys.argv[0])
7   my_dir = os.path.dirname(my_file)
8
9   # determine the pytan lib dir and add it to the path
10  parent_dir = os.path.dirname(my_dir)
11  pytan_root_dir = os.path.dirname(parent_dir)
12  lib_dir = os.path.join(pytan_root_dir, 'lib')
13  path_adds = [lib_dir]
14
15  for aa in path_adds:
16      if aa not in sys.path:
17          sys.path.append(aa)
18
19
20  # connection info for Tanium Server
21  USERNAME = "Tanium User"
22  PASSWORD = "T@n!um"
23  HOST = "172.16.31.128"
24  PORT = "443"
25
26  # Logging conrols
27  LOGLEVEL = 2
28  DEBUGFORMAT = False
```

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

Output from Python Code

```
1    Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2
3    Type of response:  <class 'taniumpy.object_types.saved_action_list.SavedActionList'>
4
5    print of response:
6    SavedActionList, len: 1
7
8    length of response (number of objects returned):
9    1
10
11   print the first object returned in JSON format:
```

```
12  {
13    "_type": "saved_action",
14    "action_group_id": 0,
15    "approved_flag": 1,
16    "approver": {
17      "_type": "user",
18      "id": 1
19    },
20    "comment": "Distribute Tanium Standard Utilities",
21    "creation_time": "2015-08-07T13:22:26",
22    "distribute_seconds": 3200,
23    "end_time": "Never",
24    "expire_seconds": 3300,
25    "id": 1,
26    "issue_count": 0,
27  ..trimmed for brevity..
```

Get all users

Get all users

Example Python Code

```python
1   import os
2   import sys
3   sys.dont_write_bytecode = True
4
5   # Determine our script name, script dir
6   my_file = os.path.abspath(sys.argv[0])
7   my_dir = os.path.dirname(my_file)
8
9   # determine the pytan lib dir and add it to the path
10  parent_dir = os.path.dirname(my_dir)
11  pytan_root_dir = os.path.dirname(parent_dir)
12  lib_dir = os.path.join(pytan_root_dir, 'lib')
13  path_adds = [lib_dir]
14
15  for aa in path_adds:
16      if aa not in sys.path:
17          sys.path.append(aa)
18
19
20  # connection info for Tanium Server
21  USERNAME = "Tanium User"
22  PASSWORD = "T@n!um"
23  HOST = "172.16.31.128"
24  PORT = "443"
25
26  # Logging conrols
27  LOGLEVEL = 2
28  DEBUGFORMAT = False
29
30  import tempfile
31
32  import pytan
```

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

Output from Python Code

```
1   Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2
3   Type of response:  <class 'taniumpy.object_types.user_list.UserList'>
4
5   print of response:
6   UserList, len: 6
7
8   length of response (number of objects returned):
9   6
10
11  print the first object returned in JSON format:
12  {
13    "_type": "user",
14    "deleted_flag": 0,
15    "group_id": 0,
16    "id": 1,
```

```
17      "last_login": "2015-08-07T13:21:59",
18      "local_admin_flag": -1,
19      "name": "Jim Olsen",
20      "permissions": {
21        "_type": "permissions",
22        "permission": [
23          "admin",
24          "sensor_read",
25          "sensor_write",
26          "question_read",
27   ..trimmed for brevity..
```

Get all saved actions

Get all saved actions

Example Python Code

```python
1    import os
2    import sys
3    sys.dont_write_bytecode = True
4
5    # Determine our script name, script dir
6    my_file = os.path.abspath(sys.argv[0])
7    my_dir = os.path.dirname(my_file)
8
9    # determine the pytan lib dir and add it to the path
10   parent_dir = os.path.dirname(my_dir)
11   pytan_root_dir = os.path.dirname(parent_dir)
12   lib_dir = os.path.join(pytan_root_dir, 'lib')
13   path_adds = [lib_dir]
14
15   for aa in path_adds:
16       if aa not in sys.path:
17           sys.path.append(aa)
18
19
20   # connection info for Tanium Server
21   USERNAME = "Tanium User"
22   PASSWORD = "T@n!um"
23   HOST = "172.16.31.128"
24   PORT = "443"
25
26   # Logging conrols
27   LOGLEVEL = 2
28   DEBUGFORMAT = False
29
30   import tempfile
31
32   import pytan
33   handler = pytan.Handler(
34       username=USERNAME,
35       password=PASSWORD,
36       host=HOST,
37       port=PORT,
```

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

Output from Python Code

```
1    Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2
3    Type of response:  <class 'taniumpy.object_types.saved_action_list.SavedActionList'>
4
5    print of response:
6    SavedActionList, len: 4
7
8    length of response (number of objects returned):
9    4
10
11   print the first object returned in JSON format:
12   {
13     "_type": "saved_action",
14     "action_group_id": 0,
15     "approved_flag": 1,
16     "approver": {
17       "_type": "user",
18       "id": 1
19     },
20     "cache_row_id": 0,
21     "comment": "Distribute Tanium Standard Utilities",
```

```
22      "creation_time": "2015-08-07T13:22:26",
23      "distribute_seconds": 3200,
24      "end_time": "Never",
25      "expire_seconds": 3300,
26      "id": 1,
27   ..trimmed for brevity..
```

Get all settings

Get all system settings

Example Python Code

```python
1   import os
2   import sys
3   sys.dont_write_bytecode = True
4
5   # Determine our script name, script dir
6   my_file = os.path.abspath(sys.argv[0])
7   my_dir = os.path.dirname(my_file)
8
9   # determine the pytan lib dir and add it to the path
10  parent_dir = os.path.dirname(my_dir)
11  pytan_root_dir = os.path.dirname(parent_dir)
12  lib_dir = os.path.join(pytan_root_dir, 'lib')
13  path_adds = [lib_dir]
14
15  for aa in path_adds:
16      if aa not in sys.path:
17          sys.path.append(aa)
18
19
20  # connection info for Tanium Server
21  USERNAME = "Tanium User"
22  PASSWORD = "T@n!um"
23  HOST = "172.16.31.128"
24  PORT = "443"
25
26  # Logging conrols
27  LOGLEVEL = 2
28  DEBUGFORMAT = False
29
30  import tempfile
31
32  import pytan
33  handler = pytan.Handler(
34      username=USERNAME,
35      password=PASSWORD,
36      host=HOST,
37      port=PORT,
38      loglevel=LOGLEVEL,
39      debugformat=DEBUGFORMAT,
40  )
41
42  print handler
```

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

Output from Python Code

```
1    Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2
3    Type of response:  <class 'taniumpy.object_types.system_setting_list.SystemSettingList'>
4
5    print of response:
6    SystemSettingList, len: 91
7
8    length of response (number of objects returned):
9    91
10
11   print the first object returned in JSON format:
12   {
13     "_type": "system_setting",
14     "audit_data": {
15       "_type": "audit_data",
16       "creation_time": "2015-08-07T13:22:35",
17       "last_modified_by": "Jim Olsen",
18       "modification_time": "2015-08-07T13:22:35"
19     },
20     "cache_row_id": 0,
21     "default_value": "0",
22     "hidden_flag": 0,
23     "id": 1,
24     "name": "load_initial_content",
25     "read_only_flag": 0,
26     "setting_type": "Server",
```

```
27    ..trimmed for brevity..
```

Get all saved questions

Get all saved questions

Example Python Code

```python
1   import os
2   import sys
3   sys.dont_write_bytecode = True
4
5   # Determine our script name, script dir
6   my_file = os.path.abspath(sys.argv[0])
7   my_dir = os.path.dirname(my_file)
8
9   # determine the pytan lib dir and add it to the path
10  parent_dir = os.path.dirname(my_dir)
11  pytan_root_dir = os.path.dirname(parent_dir)
12  lib_dir = os.path.join(pytan_root_dir, 'lib')
13  path_adds = [lib_dir]
14
15  for aa in path_adds:
16      if aa not in sys.path:
17          sys.path.append(aa)
18
19
20  # connection info for Tanium Server
21  USERNAME = "Tanium User"
22  PASSWORD = "T@n!um"
23  HOST = "172.16.31.128"
24  PORT = "443"
25
26  # Logging conrols
27  LOGLEVEL = 2
28  DEBUGFORMAT = False
29
30  import tempfile
31
32  import pytan
33  handler = pytan.Handler(
34      username=USERNAME,
35      password=PASSWORD,
36      host=HOST,
37      port=PORT,
38      loglevel=LOGLEVEL,
39      debugformat=DEBUGFORMAT,
40  )
41
42  print handler
43
44  # setup the arguments for the handler method
45  kwargs = {}
46  kwargs["objtype"] = u'saved_question'
47
```

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

Output from Python Code

```
1   Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2
3   Type of response:  <class 'taniumpy.object_types.saved_question_list.SavedQuestionList'>
4
5   print of response:
6   SavedQuestionList, len: 107
7
8   length of response (number of objects returned):
9   107
10
11  print the first object returned in JSON format:
12  {
13    "_type": "saved_question",
14    "action_tracking_flag": 0,
15    "archive_enabled_flag": 0,
16    "archive_owner": {
17      "_type": "user"
18    },
19    "cache_row_id": 0,
20    "expire_seconds": 600,
21    "hidden_flag": 0,
22    "id": 1,
23    "issue_seconds": 120,
24    "issue_seconds_never_flag": 0,
25    "keep_seconds": 0,
26    "mod_time": "2015-08-07T13:22:22",
27  ..trimmed for brevity..
```

Get all userroless

Get all user roles

Example Python Code

```python
import os
import sys
sys.dont_write_bytecode = True

# Determine our script name, script dir
my_file = os.path.abspath(sys.argv[0])
my_dir = os.path.dirname(my_file)

# determine the pytan lib dir and add it to the path
parent_dir = os.path.dirname(my_dir)
pytan_root_dir = os.path.dirname(parent_dir)
lib_dir = os.path.join(pytan_root_dir, 'lib')
path_adds = [lib_dir]

for aa in path_adds:
    if aa not in sys.path:
        sys.path.append(aa)


# connection info for Tanium Server
USERNAME = "Tanium User"
PASSWORD = "T@n!um"
HOST = "172.16.31.128"
PORT = "443"

# Logging conrols
LOGLEVEL = 2
DEBUGFORMAT = False

import tempfile

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)
```

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

Output from Python Code

```
1   Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2
3   Type of response:  <class 'taniumpy.object_types.user_role_list.UserRoleList'>
4
5   print of response:
6   UserRoleList, len: 9
7
8   length of response (number of objects returned):
9   9
10
11  print the first object returned in JSON format:
12  {
13    "_type": "role",
14    "description": "Administrators can perform all functions in the system, including creating other u
15    "id": 1,
16    "name": "Administrator",
17    "permissions": {
18      "_type": "permissions",
19      "permission": [
20        "admin",
21        "sensor_read",
22        "sensor_write",
23        "question_read",
24        "question_write",
25        "action_read",
26        "action_write",
27  ..trimmed for brevity..
```

Get all questions

Get all questions

Example Python Code

```
1   import os
2   import sys
3   sys.dont_write_bytecode = True
4
5   # Determine our script name, script dir
6   my_file = os.path.abspath(sys.argv[0])
7   my_dir = os.path.dirname(my_file)
8
9   # determine the pytan lib dir and add it to the path
10  parent_dir = os.path.dirname(my_dir)
11  pytan_root_dir = os.path.dirname(parent_dir)
12  lib_dir = os.path.join(pytan_root_dir, 'lib')
13  path_adds = [lib_dir]
14
15  for aa in path_adds:
16      if aa not in sys.path:
17          sys.path.append(aa)
18
19
20  # connection info for Tanium Server
21  USERNAME = "Tanium User"
22  PASSWORD = "T@n!um"
23  HOST = "172.16.31.128"
24  PORT = "443"
25
26  # Logging conrols
27  LOGLEVEL = 2
28  DEBUGFORMAT = False
29
30  import tempfile
31
32  import pytan
33  handler = pytan.Handler(
34      username=USERNAME,
35      password=PASSWORD,
36      host=HOST,
37      port=PORT,
38      loglevel=LOGLEVEL,
39      debugformat=DEBUGFORMAT,
40  )
41
42  print handler
43
44  # setup the arguments for the handler method
45  kwargs = {}
46  kwargs["objtype"] = u'question'
47
48  # call the handler with the get_all method, passing in kwargs for arguments
49  response = handler.get_all(**kwargs)
50
51  print ""
52  print "Type of response: ", type(response)
53
54  print ""
55  print "print of response:"
56  print response
57
58  print ""
```

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

Output from Python Code

```
1    Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2
3    Type of response:  <class 'taniumpy.object_types.question_list.QuestionList'>
4
5    print of response:
6    QuestionList, len: 174
7
8    length of response (number of objects returned):
9    174
10
11   print the first object returned in JSON format:
12   {
13     "_type": "question",
14     "action_tracking_flag": 0,
15     "cache_row_id": 1,
16     "context_group": {
17       "_type": "group",
18       "id": 0
19     },
20     "expiration": "2015-08-07T13:32:29",
21     "expire_seconds": 600,
22     "hidden_flag": 0,
23     "id": 104,
24     "management_rights_group": {
25       "_type": "group",
26       "id": 0
27   ..trimmed for brevity..
```

Get all groups

Get all groups

Example Python Code

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

```
6   my_file = os.path.abspath(sys.argv[0])
7   my_dir = os.path.dirname(my_file)
8
9   # determine the pytan lib dir and add it to the path
10  parent_dir = os.path.dirname(my_dir)
11  pytan_root_dir = os.path.dirname(parent_dir)
12  lib_dir = os.path.join(pytan_root_dir, 'lib')
13  path_adds = [lib_dir]
14
15  for aa in path_adds:
16      if aa not in sys.path:
17          sys.path.append(aa)
18
19
20  # connection info for Tanium Server
21  USERNAME = "Tanium User"
22  PASSWORD = "T@n!um"
23  HOST = "172.16.31.128"
24  PORT = "443"
25
26  # Logging conrols
27  LOGLEVEL = 2
28  DEBUGFORMAT = False
29
30  import tempfile
31
32  import pytan
33  handler = pytan.Handler(
34      username=USERNAME,
35      password=PASSWORD,
36      host=HOST,
37      port=PORT,
38      loglevel=LOGLEVEL,
39      debugformat=DEBUGFORMAT,
40  )
41
42  print handler
43
44  # setup the arguments for the handler method
45  kwargs = {}
46  kwargs["objtype"] = u'group'
47
48  # call the handler with the get_all method, passing in kwargs for arguments
49  response = handler.get_all(**kwargs)
50
51  print ""
52  print "Type of response: ", type(response)
53
54  print ""
55  print "print of response:"
56  print response
57
58  print ""
59  print "length of response (number of objects returned): "
60  print len(response)
61
62  print ""
63  print "print the first object returned in JSON format:"
```

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

Output from Python Code

```
1   Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2
3   Type of response:  <class 'taniumpy.object_types.group_list.GroupList'>
4
5   print of response:
6   GroupList, len: 2
7
8   length of response (number of objects returned):
9   2
10
11  print the first object returned in JSON format:
12  {
13    "_type": "group",
14    "and_flag": 0,
15    "deleted_flag": 0,
16    "filters": {
17      "_type": "filters",
18      "filter": []
19    },
20    "id": 64,
21    "name": "All Computers",
22    "not_flag": 0,
23    "sub_groups": {
24      "_type": "groups",
25      "group": []
26    },
27  ..trimmed for brevity..
```

Get all sensors

Get all sensors

Example Python Code

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

```
11  pytan_root_dir = os.path.dirname(parent_dir)
12  lib_dir = os.path.join(pytan_root_dir, 'lib')
13  path_adds = [lib_dir]
14
15  for aa in path_adds:
16      if aa not in sys.path:
17          sys.path.append(aa)
18
19
20  # connection info for Tanium Server
21  USERNAME = "Tanium User"
22  PASSWORD = "T@n!um"
23  HOST = "172.16.31.128"
24  PORT = "443"
25
26  # Logging conrols
27  LOGLEVEL = 2
28  DEBUGFORMAT = False
29
30  import tempfile
31
32  import pytan
33  handler = pytan.Handler(
34      username=USERNAME,
35      password=PASSWORD,
36      host=HOST,
37      port=PORT,
38      loglevel=LOGLEVEL,
39      debugformat=DEBUGFORMAT,
40  )
41
42  print handler
43
44  # setup the arguments for the handler method
45  kwargs = {}
46  kwargs["objtype"] = u'sensor'
47
48  # call the handler with the get_all method, passing in kwargs for arguments
49  response = handler.get_all(**kwargs)
50
51  print ""
52  print "Type of response: ", type(response)
53
54  print ""
55  print "print of response:"
56  print response
57
58  print ""
59  print "length of response (number of objects returned): "
60  print len(response)
61
62  print ""
63  print "print the first object returned in JSON format:"
64  out = response.to_json(response[0])
65  if len(out.splitlines()) > 15:
66      out = out.splitlines()[0:15]
67      out.append('..trimmed for brevity..')
68      out = '\n'.join(out)
```

```
69
70   print out
```

Output from Python Code

```
1   Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2
3   Type of response:  <class 'taniumpy.object_types.sensor_list.SensorList'>
4
5   print of response:
6   SensorList, len: 419
7
8   length of response (number of objects returned):
9   419
10
11  print the first object returned in JSON format:
12  {
13    "_type": "sensor",
14    "cache_row_id": 0,
15    "category": "Reserved",
16    "description": "The recorded state of each action a client has taken recently in the form of id:st
17    "exclude_from_parse_flag": 1,
18    "hash": 1792443391,
19    "hidden_flag": 0,
20    "id": 1,
21    "ignore_case_flag": 1,
22    "max_age_seconds": 3600,
23    "name": "Action Statuses",
24    "queries": {
25      "_type": "queries",
26      "query": [
27  ..trimmed for brevity..
```

Get all whitelisted urls

Get all whitelisted urls

Example Python Code

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

```
16        if aa not in sys.path:
17            sys.path.append(aa)
18
19
20    # connection info for Tanium Server
21    USERNAME = "Tanium User"
22    PASSWORD = "T@n!um"
23    HOST = "172.16.31.128"
24    PORT = "443"
25
26    # Logging conrols
27    LOGLEVEL = 2
28    DEBUGFORMAT = False
29
30    import tempfile
31
32    import pytan
33    handler = pytan.Handler(
34        username=USERNAME,
35        password=PASSWORD,
36        host=HOST,
37        port=PORT,
38        loglevel=LOGLEVEL,
39        debugformat=DEBUGFORMAT,
40    )
41
42    print handler
43
44    # setup the arguments for the handler method
45    kwargs = {}
46    kwargs["objtype"] = u'whitelisted_url'
47
48    # call the handler with the get_all method, passing in kwargs for arguments
49    response = handler.get_all(**kwargs)
50
51    print ""
52    print "Type of response: ", type(response)
53
54    print ""
55    print "print of response:"
56    print response
57
58    print ""
59    print "length of response (number of objects returned): "
60    print len(response)
61
62    print ""
63    print "print the first object returned in JSON format:"
64    out = response.to_json(response[0])
65    if len(out.splitlines()) > 15:
66        out = out.splitlines()[0:15]
67        out.append('..trimmed for brevity..')
68        out = '\n'.join(out)
69
70    print out
```

Output from Python Code

```
1   Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2
3   Type of response:  <class 'taniumpy.object_types.white_listed_url_list.WhiteListedUrlList'>
4
5   print of response:
6   WhiteListedUrlList, len: 46
7
8   length of response (number of objects returned):
9   46
10
11  print the first object returned in JSON format:
12  {
13    "_type": "white_listed_url",
14    "download_seconds": 86400,
15    "id": 1,
16    "url_regex": "test1"
17  }
```

Get all clients

Get all clients

Example Python Code

```python
1   import os
2   import sys
3   sys.dont_write_bytecode = True
4
5   # Determine our script name, script dir
6   my_file = os.path.abspath(sys.argv[0])
7   my_dir = os.path.dirname(my_file)
8
9   # determine the pytan lib dir and add it to the path
10  parent_dir = os.path.dirname(my_dir)
11  pytan_root_dir = os.path.dirname(parent_dir)
12  lib_dir = os.path.join(pytan_root_dir, 'lib')
13  path_adds = [lib_dir]
14
15  for aa in path_adds:
16      if aa not in sys.path:
17          sys.path.append(aa)
18
19
20  # connection info for Tanium Server
21  USERNAME = "Tanium User"
22  PASSWORD = "T@n!um"
23  HOST = "172.16.31.128"
24  PORT = "443"
25
26  # Logging conrols
27  LOGLEVEL = 2
28  DEBUGFORMAT = False
29
30  import tempfile
31
```

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

Output from Python Code

```
1   Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2
3   Type of response:  <class 'taniumpy.object_types.system_status_list.SystemStatusList'>
4
5   print of response:
6   SystemStatusList, len: 2
7
8   length of response (number of objects returned):
9   2
10
11  print the first object returned in JSON format:
12  {
13    "_type": "client_status",
14    "cache_row_id": 0,
15    "computer_id": "3888017885",
```

```
16      "full_version": "5.1.314.7724",
17      "host_name": "Casus-Belli.local",
18      "ipaddress_client": "172.16.31.1",
19      "ipaddress_server": "172.16.31.1",
20      "last_registration": "2015-08-07T19:44:58",
21      "port_number": 17472,
22      "protocol_version": 314,
23      "public_key_valid": 1,
24      "send_state": "Forward Only",
25      "status": "Leader, Slow Link Behind"
26   }
```

Get all packages

Get all packages

Example Python Code

```python
1    import os
2    import sys
3    sys.dont_write_bytecode = True
4
5    # Determine our script name, script dir
6    my_file = os.path.abspath(sys.argv[0])
7    my_dir = os.path.dirname(my_file)
8
9    # determine the pytan lib dir and add it to the path
10   parent_dir = os.path.dirname(my_dir)
11   pytan_root_dir = os.path.dirname(parent_dir)
12   lib_dir = os.path.join(pytan_root_dir, 'lib')
13   path_adds = [lib_dir]
14
15   for aa in path_adds:
16       if aa not in sys.path:
17           sys.path.append(aa)
18
19
20   # connection info for Tanium Server
21   USERNAME = "Tanium User"
22   PASSWORD = "T@n!um"
23   HOST = "172.16.31.128"
24   PORT = "443"
25
26   # Logging conrols
27   LOGLEVEL = 2
28   DEBUGFORMAT = False
29
30   import tempfile
31
32   import pytan
33   handler = pytan.Handler(
34       username=USERNAME,
35       password=PASSWORD,
36       host=HOST,
37       port=PORT,
```

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

Output from Python Code

```
1     Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2
3     Type of response:  <class 'taniumpy.object_types.package_spec_list.PackageSpecList'>
4
5     print of response:
6     PackageSpecList, len: 72
7
8     length of response (number of objects returned):
9     72
10
11    print the first object returned in JSON format:
12    {
13      "_type": "package_spec",
14      "available_time": "2015-08-07T13:22:50",
15      "cache_row_id": 0,
16      "command": "cmd /c cscript //T:900 java-installer.vbs /KillAppsUsingJava:Yes /RebootIfNeeded:Yes /
17      "command_timeout": 900,
18      "creation_time": "2015-08-07T13:22:16",
19      "deleted_flag": 0,
20      "display_name": "Update Java 64-bit - Kill / Reboot",
21      "expire_seconds": 1500,
```

```
22    "files": {
23      "_type": "package_files",
24      "file": [
25        {
26          "_type": "file",
27  ..trimmed for brevity..
```

Get all actions

Get all actions

Example Python Code

```python
1   import os
2   import sys
3   sys.dont_write_bytecode = True
4
5   # Determine our script name, script dir
6   my_file = os.path.abspath(sys.argv[0])
7   my_dir = os.path.dirname(my_file)
8
9   # determine the pytan lib dir and add it to the path
10  parent_dir = os.path.dirname(my_dir)
11  pytan_root_dir = os.path.dirname(parent_dir)
12  lib_dir = os.path.join(pytan_root_dir, 'lib')
13  path_adds = [lib_dir]
14
15  for aa in path_adds:
16      if aa not in sys.path:
17          sys.path.append(aa)
18
19
20  # connection info for Tanium Server
21  USERNAME = "Tanium User"
22  PASSWORD = "T@n!um"
23  HOST = "172.16.31.128"
24  PORT = "443"
25
26  # Logging conrols
27  LOGLEVEL = 2
28  DEBUGFORMAT = False
29
30  import tempfile
31
32  import pytan
33  handler = pytan.Handler(
34      username=USERNAME,
35      password=PASSWORD,
36      host=HOST,
37      port=PORT,
38      loglevel=LOGLEVEL,
39      debugformat=DEBUGFORMAT,
40  )
41
42  print handler
```

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

Output from Python Code

```
1   Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2
3   Type of response:  <class 'taniumpy.object_types.action_list.ActionList'>
4
5   print of response:
6   ActionList, len: 38
7
8   length of response (number of objects returned):
9   38
10
11  print the first object returned in JSON format:
12  {
13    "_type": "action",
14    "action_group": {
15      "_type": "group",
16      "id": 0,
17      "name": "Default"
18    },
19    "approver": {
20      "_type": "user",
21      "id": 2,
22      "name": "Tanium User"
23    },
24    "cache_row_id": 0,
25    "comment": "Distribute Tanium Standard Utilities",
26    "creation_time": "2015-08-07T13:26:19",
```

```
27  ..trimmed for brevity..
```

PyTan API Invalid Get Object Examples

Invalid get action single by name

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

Example Python Code

```python
1   import os
2   import sys
3   sys.dont_write_bytecode = True
4
5   # Determine our script name, script dir
6   my_file = os.path.abspath(sys.argv[0])
7   my_dir = os.path.dirname(my_file)
8
9   # determine the pytan lib dir and add it to the path
10  parent_dir = os.path.dirname(my_dir)
11  pytan_root_dir = os.path.dirname(parent_dir)
12  lib_dir = os.path.join(pytan_root_dir, 'lib')
13  path_adds = [lib_dir]
14
15  for aa in path_adds:
16      if aa not in sys.path:
17          sys.path.append(aa)
18
19
20  # connection info for Tanium Server
21  USERNAME = "Tanium User"
22  PASSWORD = "T@n!um"
23  HOST = "172.16.31.128"
24  PORT = "443"
25
26  # Logging conrols
27  LOGLEVEL = 2
28  DEBUGFORMAT = False
29
30  import tempfile
31
32  import pytan
33  handler = pytan.Handler(
34      username=USERNAME,
35      password=PASSWORD,
36      host=HOST,
37      port=PORT,
38      loglevel=LOGLEVEL,
39      debugformat=DEBUGFORMAT,
40  )
41
42  print handler
43
44  # setup the arguments for the handler method
45  kwargs = {}
```

```
46   kwargs["objtype"] = u'action'
47   kwargs["name"] = u'Distribute Tanium Standard Utilities'
48
49
50   # call the handler with the get method, passing in kwargs for arguments
51   # this should throw an exception: pytan.exceptions.HandlerError
52   import traceback
53   try:
54       handler.get(**kwargs)
55   except Exception as e:
56       traceback.print_exc(file=sys.stdout)
```

Output from Python Code

```
1   Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2   Traceback (most recent call last):
3     File "<string>", line 55, in <module>
4     File "/Users/jolsen/gh/pytan/lib/pytan/utils.py", line 2699, in wrap
5       ret = f(*args, **kwargs)
6     File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 1245, in get
7       raise pytan.exceptions.HandlerError(err(objtype, api_attrs))
8   HandlerError: Getting a action requires at least one filter: ['id']
```

Invalid get question by name

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

Example Python Code

```
1   import os
2   import sys
3   sys.dont_write_bytecode = True
4
5   # Determine our script name, script dir
6   my_file = os.path.abspath(sys.argv[0])
7   my_dir = os.path.dirname(my_file)
8
9   # determine the pytan lib dir and add it to the path
10  parent_dir = os.path.dirname(my_dir)
11  pytan_root_dir = os.path.dirname(parent_dir)
12  lib_dir = os.path.join(pytan_root_dir, 'lib')
13  path_adds = [lib_dir]
14
15  for aa in path_adds:
16      if aa not in sys.path:
17          sys.path.append(aa)
18
19
20  # connection info for Tanium Server
21  USERNAME = "Tanium User"
22  PASSWORD = "T@n!um"
23  HOST = "172.16.31.128"
24  PORT = "443"
25
```

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

Output from Python Code

```
1   Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2   Traceback (most recent call last):
3     File "<string>", line 55, in <module>
4     File "/Users/jolsen/gh/pytan/lib/pytan/utils.py", line 2699, in wrap
5       ret = f(*args, **kwargs)
6     File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 1245, in get
7       raise pytan.exceptions.HandlerError(err(objtype, api_attrs))
8   HandlerError: Getting a question requires at least one filter: ['id']
```

PyTan API Valid Deploy Action Examples

Deploy action simple

Deploy an action against all computers using human strings.

Example Python Code

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

```
 4
 5   # Determine our script name, script dir
 6   my_file = os.path.abspath(sys.argv[0])
 7   my_dir = os.path.dirname(my_file)
 8
 9   # determine the pytan lib dir and add it to the path
10   parent_dir = os.path.dirname(my_dir)
11   pytan_root_dir = os.path.dirname(parent_dir)
12   lib_dir = os.path.join(pytan_root_dir, 'lib')
13   path_adds = [lib_dir]
14
15   for aa in path_adds:
16       if aa not in sys.path:
17           sys.path.append(aa)
18
19
20   # connection info for Tanium Server
21   USERNAME = "Tanium User"
22   PASSWORD = "T@n!um"
23   HOST = "172.16.31.128"
24   PORT = "443"
25
26   # Logging conrols
27   LOGLEVEL = 2
28   DEBUGFORMAT = False
29
30   import tempfile
31
32   import pytan
33   handler = pytan.Handler(
34       username=USERNAME,
35       password=PASSWORD,
36       host=HOST,
37       port=PORT,
38       loglevel=LOGLEVEL,
39       debugformat=DEBUGFORMAT,
40   )
41
42   print handler
43
44   # setup the arguments for the handler method
45   kwargs = {}
46   kwargs["run"] = True
47   kwargs["package"] = u'Distribute Tanium Standard Utilities'
48
49   # call the handler with the deploy_action method, passing in kwargs for arguments
50   response = handler.deploy_action(**kwargs)
51   import pprint, io
52
53   print ""
54   print "Type of response: ", type(response)
55
56   print ""
57   print "Pretty print of response:"
58   print pprint.pformat(response)
59
60   print ""
61   print "Print of action object: "
```

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

Output from Python Code

```
1   Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2   2015-08-07 19:45:08,345 DEBUG    pytan.handler.ActionPoller: ID 56: id resolved to 56
3   2015-08-07 19:45:08,345 DEBUG    pytan.handler.ActionPoller: ID 56: package_spec resolved to Package
4   2015-08-07 19:45:08,353 DEBUG    pytan.handler.ActionPoller: ID 56: target_group resolved to Group,
5   2015-08-07 19:45:08,353 DEBUG    pytan.handler.ActionPoller: ID 56: Result Map resolved to {'failed'
6   2015-08-07 19:45:08,353 DEBUG    pytan.handler.ActionPoller: ID 56: expiration_time resolved to 2015
7   2015-08-07 19:45:08,353 DEBUG    pytan.handler.ActionPoller: ID 56: status resolved to Open
8   2015-08-07 19:45:08,353 DEBUG    pytan.handler.ActionPoller: ID 56: stopped_flag resolved to 0
9   2015-08-07 19:45:08,353 DEBUG    pytan.handler.ActionPoller: ID 56: Object Info resolved to ID 56: P
10  2015-08-07 19:45:08,353 DEBUG    pytan.handler.ActionPoller: ID 56: Adding Question to derive passed
11  2015-08-07 19:45:08,365 DEBUG    pytan.handler.QuestionPoller: ID 1299: id resolved to 1299
12  2015-08-07 19:45:08,365 DEBUG    pytan.handler.QuestionPoller: ID 1299: expiration resolved to 2015-
13  2015-08-07 19:45:08,365 DEBUG    pytan.handler.QuestionPoller: ID 1299: query_text resolved to Get n
14  2015-08-07 19:45:08,365 DEBUG    pytan.handler.QuestionPoller: ID 1299: id resolved to 1299
15  2015-08-07 19:45:08,365 DEBUG    pytan.handler.QuestionPoller: ID 1299: Object Info resolved to Ques
16  2015-08-07 19:45:08,368 DEBUG    pytan.handler.QuestionPoller: ID 1299: Progress: Tested: 0, Passed:
17  2015-08-07 19:45:08,368 DEBUG    pytan.handler.QuestionPoller: ID 1299: Timing: Started: 2015-08-07
18  2015-08-07 19:45:08,368 INFO     pytan.handler.QuestionPoller: ID 1299: Progress Changed 0% (0 of 2)
19  2015-08-07 19:45:13,372 DEBUG    pytan.handler.QuestionPoller: ID 1299: Progress: Tested: 1, Passed:
20  2015-08-07 19:45:13,372 DEBUG    pytan.handler.QuestionPoller: ID 1299: Timing: Started: 2015-08-07
21  2015-08-07 19:45:13,372 INFO     pytan.handler.QuestionPoller: ID 1299: Progress Changed 50% (1 of 2
22  2015-08-07 19:45:18,379 DEBUG    pytan.handler.QuestionPoller: ID 1299: Progress: Tested: 2, Passed:
23  2015-08-07 19:45:18,379 DEBUG    pytan.handler.QuestionPoller: ID 1299: Timing: Started: 2015-08-07
24  2015-08-07 19:45:18,380 INFO     pytan.handler.QuestionPoller: ID 1299: Progress Changed 100% (2 of
25  2015-08-07 19:45:18,380 INFO     pytan.handler.QuestionPoller: ID 1299: Reached Threshold of 99% (2
26  2015-08-07 19:45:18,380 DEBUG    pytan.handler.ActionPoller: ID 56: Passed Count resolved to 2
27  2015-08-07 19:45:18,390 DEBUG    pytan.handler.ActionPoller: ID 56: Progress: Seen Action: 0, Expect
28  2015-08-07 19:45:18,390 DEBUG    pytan.handler.ActionPoller: ID 56: Timing: Started: 2015-08-07 19:4
29  2015-08-07 19:45:18,390 INFO     pytan.handler.ActionPoller: ID 56: Progress Changed for Seen Count
30  2015-08-07 19:45:18,397 DEBUG    pytan.handler.ActionPoller: ID 56: stopped_flag resolved to 0
31  2015-08-07 19:45:18,397 DEBUG    pytan.handler.ActionPoller: ID 56: status resolved to Open
32  2015-08-07 19:45:23,411 DEBUG    pytan.handler.ActionPoller: ID 56: Progress: Seen Action: 1, Expect
33  2015-08-07 19:45:23,411 DEBUG    pytan.handler.ActionPoller: ID 56: Timing: Started: 2015-08-07 19:4
34  2015-08-07 19:45:23,411 INFO     pytan.handler.ActionPoller: ID 56: Progress Changed for Seen Count
35  2015-08-07 19:45:23,417 DEBUG    pytan.handler.ActionPoller: ID 56: stopped_flag resolved to 0
36  2015-08-07 19:45:23,417 DEBUG    pytan.handler.ActionPoller: ID 56: status resolved to Open
37  2015-08-07 19:45:28,427 DEBUG    pytan.handler.ActionPoller: ID 56: Progress: Seen Action: 2, Expect
38  2015-08-07 19:45:28,427 DEBUG    pytan.handler.ActionPoller: ID 56: Timing: Started: 2015-08-07 19:4
39  2015-08-07 19:45:28,427 INFO     pytan.handler.ActionPoller: ID 56: Progress Changed for Seen Count
40  2015-08-07 19:45:28,433 DEBUG    pytan.handler.ActionPoller: ID 56: stopped_flag resolved to 0
41  2015-08-07 19:45:28,433 DEBUG    pytan.handler.ActionPoller: ID 56: status resolved to Open
```

```
42  2015-08-07 19:45:28,433 INFO     pytan.handler.ActionPoller: ID 56: Reached Threshold for Seen Count
43  2015-08-07 19:45:28,443 DEBUG    pytan.handler.ActionPoller: ID 56: failed: 0, finished: 2, running:
44  2015-08-07 19:45:28,443 DEBUG    pytan.handler.ActionPoller: ID 56: Timing: Started: 2015-08-07 19:4
45  2015-08-07 19:45:28,443 INFO     pytan.handler.ActionPoller: ID 56: Progress Changed for Finished Co
46  2015-08-07 19:45:28,448 DEBUG    pytan.handler.ActionPoller: ID 56: stopped_flag resolved to 0
47  2015-08-07 19:45:28,448 DEBUG    pytan.handler.ActionPoller: ID 56: status resolved to Open
48  2015-08-07 19:45:28,448 INFO     pytan.handler.ActionPoller: ID 56: Reached Threshold for Finished C
49
50  Type of response:  <type 'dict'>
51
52  Pretty print of response:
53  {'action_info': <taniumpy.object_types.result_info.ResultInfo object at 0x10c063810>,
54   'action_object': <taniumpy.object_types.action.Action object at 0x10c063a50>,
55   'action_result_map': {'failed': {'56:Expired.': [],
56                                    '56:Failed.': [],
57                                    '56:NotSucceeded.': [],
58                                    '56:Stopped.': [],
59                                    'total': 0},
60                         'finished': {'56:Completed.': ['Casus-Belli.local',
61                                                        'JTANIUM1.localdomain'],
62                                      '56:Expired.': [],
63                                      '56:Failed.': [],
64                                      '56:NotSucceeded.': [],
65                                      '56:Stopped.': [],
66                                      '56:Succeeded.': [],
67                                      '56:Verified.': [],
68                                      'total': 2},
69                         'running': {'56:Copying.': [],
70                                     '56:Downloading.': [],
71                                     '56:PendingVerification.': [],
72                                     '56:Running.': [],
73                                     '56:Waiting.': [],
74                                     'total': 0},
75                         'success': {'56:Completed.': ['Casus-Belli.local',
76                                                       'JTANIUM1.localdomain'],
77                                     '56:Verified.': [],
78                                     'total': 2},
79                         'unknown': {'total': 0}},
80   'action_results': <taniumpy.object_types.result_set.ResultSet object at 0x11acb14d0>,
81   'group_object': None,
82   'package_object': <taniumpy.object_types.package_spec.PackageSpec object at 0x10c03d0d0>,
83   'poller_object': <pytan.pollers.ActionPoller object at 0x10c063b10>,
84   'poller_success': True,
85   'saved_action_object': <taniumpy.object_types.saved_action.SavedAction object at 0x10bf69750>}
86
87  Print of action object:
88  Action, name: 'API Deploy Distribute Tanium Standard Utilities', id: 56
89
90  CSV Results of response:
91  Action Statuses,Computer Name
92  56:Completed.,Casus-Belli.local
93  56:Completed.,JTANIUM1.localdomain
```

Deploy action simple without results

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

Example Python Code

```python
import os
import sys
sys.dont_write_bytecode = True

# Determine our script name, script dir
my_file = os.path.abspath(sys.argv[0])
my_dir = os.path.dirname(my_file)

# determine the pytan lib dir and add it to the path
parent_dir = os.path.dirname(my_dir)
pytan_root_dir = os.path.dirname(parent_dir)
lib_dir = os.path.join(pytan_root_dir, 'lib')
path_adds = [lib_dir]

for aa in path_adds:
    if aa not in sys.path:
        sys.path.append(aa)


# connection info for Tanium Server
USERNAME = "Tanium User"
PASSWORD = "T@n!um"
HOST = "172.16.31.128"
PORT = "443"

# Logging conrols
LOGLEVEL = 2
DEBUGFORMAT = False

import tempfile

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 method, passing in kwargs for arguments
```

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

Output from Python Code

```
1   Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2   2015-08-07 19:45:28,497 DEBUG    pytan.handler.ActionPoller: ID 57: id resolved to 57
3   2015-08-07 19:45:28,497 DEBUG    pytan.handler.ActionPoller: ID 57: package_spec resolved to Package
4   2015-08-07 19:45:28,503 DEBUG    pytan.handler.ActionPoller: ID 57: target_group resolved to Group,
5   2015-08-07 19:45:28,504 DEBUG    pytan.handler.ActionPoller: ID 57: Result Map resolved to {'failed'
6   2015-08-07 19:45:28,504 DEBUG    pytan.handler.ActionPoller: ID 57: expiration_time resolved to 2015
7   2015-08-07 19:45:28,504 DEBUG    pytan.handler.ActionPoller: ID 57: status resolved to Open
8   2015-08-07 19:45:28,504 DEBUG    pytan.handler.ActionPoller: ID 57: stopped_flag resolved to 0
9   2015-08-07 19:45:28,504 DEBUG    pytan.handler.ActionPoller: ID 57: Object Info resolved to ID 57: P
10
11  Type of response:  <type 'dict'>
12
13  Pretty print of response:
14  {'action_info': <taniumpy.object_types.result_info.ResultInfo object at 0x10be950d0>,
15   'action_object': <taniumpy.object_types.action.Action object at 0x10bf70fd0>,
16   'action_result_map': None,
17   'action_results': None,
18   'group_object': None,
19   'package_object': <taniumpy.object_types.package_spec.PackageSpec object at 0x11aae6050>,
20   'poller_object': <pytan.pollers.ActionPoller object at 0x11aae6090>,
21   'poller_success': None,
22   'saved_action_object': <taniumpy.object_types.saved_action.SavedAction object at 0x10c063a50>}
23
24  Print of action object:
25  Action, name: 'API Deploy Distribute Tanium Standard Utilities', id: 57
```

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

```python
import os
import sys
sys.dont_write_bytecode = True

# Determine our script name, script dir
my_file = os.path.abspath(sys.argv[0])
my_dir = os.path.dirname(my_file)

# determine the pytan lib dir and add it to the path
parent_dir = os.path.dirname(my_dir)
pytan_root_dir = os.path.dirname(parent_dir)
lib_dir = os.path.join(pytan_root_dir, 'lib')
path_adds = [lib_dir]

for aa in path_adds:
    if aa not in sys.path:
        sys.path.append(aa)


# connection info for Tanium Server
USERNAME = "Tanium User"
PASSWORD = "T@n!um"
HOST = "172.16.31.128"
PORT = "443"

# Logging conrols
LOGLEVEL = 2
DEBUGFORMAT = False

import tempfile

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 method, passing in kwargs for arguments
response = handler.deploy_action(**kwargs)
import pprint, io
```

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

Output from Python Code

```
1    Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2    2015-08-07 19:45:28,560 DEBUG     pytan.handler.ActionPoller: ID 58: id resolved to 58
3    2015-08-07 19:45:28,561 DEBUG     pytan.handler.ActionPoller: ID 58: package_spec resolved to Package
4    2015-08-07 19:45:28,567 DEBUG     pytan.handler.ActionPoller: ID 58: target_group resolved to Group,
5    2015-08-07 19:45:28,583 DEBUG     pytan.handler.ActionPoller: ID 58: Result Map resolved to {'failed'
6    2015-08-07 19:45:28,583 DEBUG     pytan.handler.ActionPoller: ID 58: expiration_time resolved to 2015
7    2015-08-07 19:45:28,583 DEBUG     pytan.handler.ActionPoller: ID 58: status resolved to Open
8    2015-08-07 19:45:28,584 DEBUG     pytan.handler.ActionPoller: ID 58: stopped_flag resolved to 0
9    2015-08-07 19:45:28,584 DEBUG     pytan.handler.ActionPoller: ID 58: Object Info resolved to ID 58: P
10   2015-08-07 19:45:28,584 DEBUG     pytan.handler.ActionPoller: ID 58: Adding Question to derive passed
11   2015-08-07 19:45:28,615 DEBUG     pytan.handler.QuestionPoller: ID 1302: id resolved to 1302
12   2015-08-07 19:45:28,615 DEBUG     pytan.handler.QuestionPoller: ID 1302: expiration resolved to 2015-
13   2015-08-07 19:45:28,615 DEBUG     pytan.handler.QuestionPoller: ID 1302: query_text resolved to Get n
14   2015-08-07 19:45:28,615 DEBUG     pytan.handler.QuestionPoller: ID 1302: id resolved to 1302
15   2015-08-07 19:45:28,615 DEBUG     pytan.handler.QuestionPoller: ID 1302: Object Info resolved to Ques
16   2015-08-07 19:45:28,618 DEBUG     pytan.handler.QuestionPoller: ID 1302: Progress: Tested: 0, Passed:
17   2015-08-07 19:45:28,618 DEBUG     pytan.handler.QuestionPoller: ID 1302: Timing: Started: 2015-08-07
18   2015-08-07 19:45:28,618 INFO      pytan.handler.QuestionPoller: ID 1302: Progress Changed 0% (0 of 2)
19   2015-08-07 19:45:33,623 DEBUG     pytan.handler.QuestionPoller: ID 1302: Progress: Tested: 1, Passed:
20   2015-08-07 19:45:33,623 DEBUG     pytan.handler.QuestionPoller: ID 1302: Timing: Started: 2015-08-07
21   2015-08-07 19:45:33,623 INFO      pytan.handler.QuestionPoller: ID 1302: Progress Changed 50% (1 of 2
22   2015-08-07 19:45:38,626 DEBUG     pytan.handler.QuestionPoller: ID 1302: Progress: Tested: 2, Passed:
23   2015-08-07 19:45:38,626 DEBUG     pytan.handler.QuestionPoller: ID 1302: Timing: Started: 2015-08-07
24   2015-08-07 19:45:38,626 INFO      pytan.handler.QuestionPoller: ID 1302: Progress Changed 100% (2 of
25   2015-08-07 19:45:38,626 INFO      pytan.handler.QuestionPoller: ID 1302: Reached Threshold of 99% (2
26   2015-08-07 19:45:38,626 DEBUG     pytan.handler.ActionPoller: ID 58: Passed Count resolved to 1
27   2015-08-07 19:45:38,638 DEBUG     pytan.handler.ActionPoller: ID 58: Progress: Seen Action: 0, Expect
28   2015-08-07 19:45:38,638 DEBUG     pytan.handler.ActionPoller: ID 58: Timing: Started: 2015-08-07 19:4
29   2015-08-07 19:45:38,638 INFO      pytan.handler.ActionPoller: ID 58: Progress Changed for Seen Count
30   2015-08-07 19:45:38,645 DEBUG     pytan.handler.ActionPoller: ID 58: stopped_flag resolved to 0
31   2015-08-07 19:45:38,645 DEBUG     pytan.handler.ActionPoller: ID 58: status resolved to Open
```

```
32  2015-08-07 19:45:43,660 DEBUG     pytan.handler.ActionPoller: ID 58: Progress: Seen Action: 0, Expect
33  2015-08-07 19:45:43,660 DEBUG     pytan.handler.ActionPoller: ID 58: Timing: Started: 2015-08-07 19:4
34  2015-08-07 19:45:43,667 DEBUG     pytan.handler.ActionPoller: ID 58: stopped_flag resolved to 0
35  2015-08-07 19:45:43,667 DEBUG     pytan.handler.ActionPoller: ID 58: status resolved to Open
36  2015-08-07 19:45:48,677 DEBUG     pytan.handler.ActionPoller: ID 58: Progress: Seen Action: 1, Expect
37  2015-08-07 19:45:48,677 DEBUG     pytan.handler.ActionPoller: ID 58: Timing: Started: 2015-08-07 19:4
38  2015-08-07 19:45:48,677 INFO      pytan.handler.ActionPoller: ID 58: Progress Changed for Seen Count
39  2015-08-07 19:45:48,684 DEBUG     pytan.handler.ActionPoller: ID 58: stopped_flag resolved to 0
40  2015-08-07 19:45:48,684 DEBUG     pytan.handler.ActionPoller: ID 58: status resolved to Open
41  2015-08-07 19:45:48,684 INFO      pytan.handler.ActionPoller: ID 58: Reached Threshold for Seen Count
42  2015-08-07 19:45:48,692 DEBUG     pytan.handler.ActionPoller: ID 58: failed: 0, finished: 1, running:
43  2015-08-07 19:45:48,692 DEBUG     pytan.handler.ActionPoller: ID 58: Timing: Started: 2015-08-07 19:4
44  2015-08-07 19:45:48,692 INFO      pytan.handler.ActionPoller: ID 58: Progress Changed for Finished Co
45  2015-08-07 19:45:48,698 DEBUG     pytan.handler.ActionPoller: ID 58: stopped_flag resolved to 0
46  2015-08-07 19:45:48,698 DEBUG     pytan.handler.ActionPoller: ID 58: status resolved to Open
47  2015-08-07 19:45:48,698 INFO      pytan.handler.ActionPoller: ID 58: Reached Threshold for Finished C
48
49  Type of response:  <type 'dict'>
50
51  Pretty print of response:
52  {'action_info': <taniumpy.object_types.result_info.ResultInfo object at 0x11aadbcd0>,
53   'action_object': <taniumpy.object_types.action.Action object at 0x11aae6090>,
54   'action_result_map': {'failed': {'58:Expired.': [],
55                                    '58:Failed.': [],
56                                    '58:NotSucceeded.': [],
57                                    '58:Stopped.': [],
58                                    'total': 0},
59                         'finished': {'58:Completed.': ['JTANIUM1.localdomain'],
60                                      '58:Expired.': [],
61                                      '58:Failed.': [],
62                                      '58:NotSucceeded.': [],
63                                      '58:Stopped.': [],
64                                      '58:Succeeded.': [],
65                                      '58:Verified.': [],
66                                      'total': 1},
67                         'running': {'58:Copying.': [],
68                                     '58:Downloading.': [],
69                                     '58:PendingVerification.': [],
70                                     '58:Running.': [],
71                                     '58:Waiting.': [],
72                                     'total': 0},
73                         'success': {'58:Completed.': ['JTANIUM1.localdomain'],
74                                     '58:Verified.': [],
75                                     'total': 1},
76                         'unknown': {'total': 0}},
77   'action_results': <taniumpy.object_types.result_set.ResultSet object at 0x11ac77c90>,
78   'group_object': <taniumpy.object_types.group.Group object at 0x10c03db90>,
79   'package_object': <taniumpy.object_types.package_spec.PackageSpec object at 0x11aae6050>,
80   'poller_object': <pytan.pollers.ActionPoller object at 0x10bea6950>,
81   'poller_success': True,
82   'saved_action_object': <taniumpy.object_types.saved_action.SavedAction object at 0x11acb14d0>}
83
84  Print of action object:
85  Action, name: 'API Deploy Distribute Tanium Standard Utilities', id: 58
86
87  CSV Results of response:
88  Action Statuses,Computer Name
```

```
89  58:Completed.,JTANIUM1.localdomain
```

Deploy action with params against windows computers

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

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

Example Python Code

```python
1   import os
2   import sys
3   sys.dont_write_bytecode = True
4
5   # Determine our script name, script dir
6   my_file = os.path.abspath(sys.argv[0])
7   my_dir = os.path.dirname(my_file)
8
9   # determine the pytan lib dir and add it to the path
10  parent_dir = os.path.dirname(my_dir)
11  pytan_root_dir = os.path.dirname(parent_dir)
12  lib_dir = os.path.join(pytan_root_dir, 'lib')
13  path_adds = [lib_dir]
14
15  for aa in path_adds:
16      if aa not in sys.path:
17          sys.path.append(aa)
18
19
20  # connection info for Tanium Server
21  USERNAME = "Tanium User"
22  PASSWORD = "T@n!um"
23  HOST = "172.16.31.128"
24  PORT = "443"
25
26  # Logging conrols
27  LOGLEVEL = 2
28  DEBUGFORMAT = False
29
30  import tempfile
31
32  import pytan
33  handler = pytan.Handler(
34      username=USERNAME,
35      password=PASSWORD,
36      host=HOST,
37      port=PORT,
38      loglevel=LOGLEVEL,
39      debugformat=DEBUGFORMAT,
40  )
41
42  print handler
43
44  # setup the arguments for the handler method
45  kwargs = {}
```

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

Output from Python Code

```
1   Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2   2015-08-07 19:45:48,755 DEBUG    pytan.handler.ActionPoller: ID 59: id resolved to 59
3   2015-08-07 19:45:48,755 DEBUG    pytan.handler.ActionPoller: ID 59: package_spec resolved to Package
4   2015-08-07 19:45:48,759 DEBUG    pytan.handler.ActionPoller: ID 59: target_group resolved to Group,
5   2015-08-07 19:45:48,773 DEBUG    pytan.handler.ActionPoller: ID 59: Result Map resolved to {'failed'
6   2015-08-07 19:45:48,773 DEBUG    pytan.handler.ActionPoller: ID 59: expiration_time resolved to 2015
7   2015-08-07 19:45:48,773 DEBUG    pytan.handler.ActionPoller: ID 59: status resolved to Open
8   2015-08-07 19:45:48,773 DEBUG    pytan.handler.ActionPoller: ID 59: stopped_flag resolved to 0
9   2015-08-07 19:45:48,773 DEBUG    pytan.handler.ActionPoller: ID 59: Object Info resolved to ID 59: P
10  2015-08-07 19:45:48,773 DEBUG    pytan.handler.ActionPoller: ID 59: Adding Question to derive passed
11  2015-08-07 19:45:48,802 DEBUG    pytan.handler.QuestionPoller: ID 1303: id resolved to 1303
12  2015-08-07 19:45:48,802 DEBUG    pytan.handler.QuestionPoller: ID 1303: expiration resolved to 2015-
13  2015-08-07 19:45:48,802 DEBUG    pytan.handler.QuestionPoller: ID 1303: query_text resolved to Get n
14  2015-08-07 19:45:48,802 DEBUG    pytan.handler.QuestionPoller: ID 1303: id resolved to 1303
15  2015-08-07 19:45:48,802 DEBUG    pytan.handler.QuestionPoller: ID 1303: Object Info resolved to Ques
16  2015-08-07 19:45:48,805 DEBUG    pytan.handler.QuestionPoller: ID 1303: Progress: Tested: 0, Passed:
17  2015-08-07 19:45:48,805 DEBUG    pytan.handler.QuestionPoller: ID 1303: Timing: Started: 2015-08-07
18  2015-08-07 19:45:48,805 INFO     pytan.handler.QuestionPoller: ID 1303: Progress Changed 0% (0 of 2)
19  2015-08-07 19:45:53,809 DEBUG    pytan.handler.QuestionPoller: ID 1303: Progress: Tested: 2, Passed:
20  2015-08-07 19:45:53,809 DEBUG    pytan.handler.QuestionPoller: ID 1303: Timing: Started: 2015-08-07
21  2015-08-07 19:45:53,809 INFO     pytan.handler.QuestionPoller: ID 1303: Progress Changed 100% (2 of
22  2015-08-07 19:45:53,809 INFO     pytan.handler.QuestionPoller: ID 1303: Reached Threshold of 99% (2
23  2015-08-07 19:45:53,809 DEBUG    pytan.handler.ActionPoller: ID 59: Passed Count resolved to 1
24  2015-08-07 19:45:53,819 DEBUG    pytan.handler.ActionPoller: ID 59: Progress: Seen Action: 0, Expect
```

```
25   2015-08-07 19:45:53,819 DEBUG    pytan.handler.ActionPoller: ID 59: Timing: Started: 2015-08-07 19:4
26   2015-08-07 19:45:53,819 INFO     pytan.handler.ActionPoller: ID 59: Progress Changed for Seen Count
27   2015-08-07 19:45:53,824 DEBUG    pytan.handler.ActionPoller: ID 59: stopped_flag resolved to 0
28   2015-08-07 19:45:53,824 DEBUG    pytan.handler.ActionPoller: ID 59: status resolved to Open
29   2015-08-07 19:45:58,837 DEBUG    pytan.handler.ActionPoller: ID 59: Progress: Seen Action: 1, Expect
30   2015-08-07 19:45:58,837 DEBUG    pytan.handler.ActionPoller: ID 59: Timing: Started: 2015-08-07 19:4
31   2015-08-07 19:45:58,837 INFO     pytan.handler.ActionPoller: ID 59: Progress Changed for Seen Count
32   2015-08-07 19:45:58,843 DEBUG    pytan.handler.ActionPoller: ID 59: stopped_flag resolved to 0
33   2015-08-07 19:45:58,844 DEBUG    pytan.handler.ActionPoller: ID 59: status resolved to Open
34   2015-08-07 19:45:58,844 INFO     pytan.handler.ActionPoller: ID 59: Reached Threshold for Seen Count
35   2015-08-07 19:45:58,852 DEBUG    pytan.handler.ActionPoller: ID 59: failed: 0, finished: 0, running:
36   2015-08-07 19:45:58,852 DEBUG    pytan.handler.ActionPoller: ID 59: Timing: Started: 2015-08-07 19:4
37   2015-08-07 19:45:58,852 INFO     pytan.handler.ActionPoller: ID 59: Progress Changed for Finished Co
38   2015-08-07 19:45:58,858 DEBUG    pytan.handler.ActionPoller: ID 59: stopped_flag resolved to 0
39   2015-08-07 19:45:58,858 DEBUG    pytan.handler.ActionPoller: ID 59: status resolved to Open
40   2015-08-07 19:46:03,873 DEBUG    pytan.handler.ActionPoller: ID 59: failed: 0, finished: 0, running:
41   2015-08-07 19:46:03,873 DEBUG    pytan.handler.ActionPoller: ID 59: Timing: Started: 2015-08-07 19:4
42   2015-08-07 19:46:03,879 DEBUG    pytan.handler.ActionPoller: ID 59: stopped_flag resolved to 0
43   2015-08-07 19:46:03,879 DEBUG    pytan.handler.ActionPoller: ID 59: status resolved to Open
44   2015-08-07 19:46:08,892 DEBUG    pytan.handler.ActionPoller: ID 59: failed: 0, finished: 1, running:
45   2015-08-07 19:46:08,892 DEBUG    pytan.handler.ActionPoller: ID 59: Timing: Started: 2015-08-07 19:4
46   2015-08-07 19:46:08,892 INFO     pytan.handler.ActionPoller: ID 59: Progress Changed for Finished Co
47   2015-08-07 19:46:08,898 DEBUG    pytan.handler.ActionPoller: ID 59: stopped_flag resolved to 0
48   2015-08-07 19:46:08,898 DEBUG    pytan.handler.ActionPoller: ID 59: status resolved to Open
49   2015-08-07 19:46:08,898 INFO     pytan.handler.ActionPoller: ID 59: Reached Threshold for Finished C
50
51   Type of response:  <type 'dict'>
52
53   Pretty print of response:
54   {'action_info': <taniumpy.object_types.result_info.ResultInfo object at 0x10c063090>,
55    'action_object': <taniumpy.object_types.action.Action object at 0x10c17ecd0>,
56    'action_result_map': {'failed': {'59:Expired.': [],
57                                     '59:Failed.': [],
58                                     '59:NotSucceeded.': [],
59                                     '59:Stopped.': [],
60                                     'total': 0},
61                          'finished': {'59:Completed.': ['JTANIUM1.localdomain'],
62                                       '59:Expired.': [],
63                                       '59:Failed.': [],
64                                       '59:NotSucceeded.': [],
65                                       '59:Stopped.': [],
66                                       '59:Succeeded.': [],
67                                       '59:Verified.': [],
68                                       'total': 1},
69                          'running': {'59:Copying.': [],
70                                      '59:Downloading.': ['JTANIUM1.localdomain'],
71                                      '59:PendingVerification.': [],
72                                      '59:Running.': [],
73                                      '59:Waiting.': [],
74                                      'total': 1},
75                          'success': {'59:Completed.': ['JTANIUM1.localdomain'],
76                                      '59:Verified.': [],
77                                      'total': 1},
78                          'unknown': {'total': 0}},
79    'action_results': <taniumpy.object_types.result_set.ResultSet object at 0x10c17ec10>,
80    'group_object': <taniumpy.object_types.group.Group object at 0x10c03d210>,
81    'package_object': <taniumpy.object_types.package_spec.PackageSpec object at 0x10bf69dd0>,
82    'poller_object': <pytan.pollers.ActionPoller object at 0x10c063250>,
```

```
83    'poller_success': True,
84    'saved_action_object': <taniumpy.object_types.saved_action.SavedAction object at 0x10c03ddd0>}
85
86  Print of action object:
87  Action, name: 'API Deploy Custom Tagging - Add Tags', id: 59
88
89  CSV Results of response:
90  Action Statuses,Computer Name
91  59:Completed.,JTANIUM1.localdomain
```

PyTan API Invalid Deploy Action Examples

Invalid deploy action run false

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

Example Python Code

```python
1   import os
2   import sys
3   sys.dont_write_bytecode = True
4
5   # Determine our script name, script dir
6   my_file = os.path.abspath(sys.argv[0])
7   my_dir = os.path.dirname(my_file)
8
9   # determine the pytan lib dir and add it to the path
10  parent_dir = os.path.dirname(my_dir)
11  pytan_root_dir = os.path.dirname(parent_dir)
12  lib_dir = os.path.join(pytan_root_dir, 'lib')
13  path_adds = [lib_dir]
14
15  for aa in path_adds:
16      if aa not in sys.path:
17          sys.path.append(aa)
18
19
20  # connection info for Tanium Server
21  USERNAME = "Tanium User"
22  PASSWORD = "T@n!um"
23  HOST = "172.16.31.128"
24  PORT = "443"
25
26  # Logging conrols
27  LOGLEVEL = 2
28  DEBUGFORMAT = False
29
30  import tempfile
31
32  import pytan
33  handler = pytan.Handler(
34      username=USERNAME,
35      password=PASSWORD,
36      host=HOST,
```

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

Output from Python Code

```
1    Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2    2015-08-07 19:46:08,957 DEBUG     pytan.handler.QuestionPoller: ID 1304: id resolved to 1304
3    2015-08-07 19:46:08,957 DEBUG     pytan.handler.QuestionPoller: ID 1304: expiration resolved to 2015-
4    2015-08-07 19:46:08,957 DEBUG     pytan.handler.QuestionPoller: ID 1304: query_text resolved to Get C
5    2015-08-07 19:46:08,957 DEBUG     pytan.handler.QuestionPoller: ID 1304: id resolved to 1304
6    2015-08-07 19:46:08,957 DEBUG     pytan.handler.QuestionPoller: ID 1304: Object Info resolved to Ques
7    2015-08-07 19:46:08,962 DEBUG     pytan.handler.QuestionPoller: ID 1304: Progress: Tested: 0, Passed:
8    2015-08-07 19:46:08,962 DEBUG     pytan.handler.QuestionPoller: ID 1304: Timing: Started: 2015-08-07
9    2015-08-07 19:46:08,962 INFO      pytan.handler.QuestionPoller: ID 1304: Progress Changed 0% (0 of 2)
10   2015-08-07 19:46:13,969 DEBUG     pytan.handler.QuestionPoller: ID 1304: Progress: Tested: 2, Passed:
11   2015-08-07 19:46:13,969 DEBUG     pytan.handler.QuestionPoller: ID 1304: Timing: Started: 2015-08-07
12   2015-08-07 19:46:13,969 INFO      pytan.handler.QuestionPoller: ID 1304: Progress Changed 100% (2 of
13   2015-08-07 19:46:13,969 INFO      pytan.handler.QuestionPoller: ID 1304: Reached Threshold of 99% (2
14   2015-08-07 19:46:13,974 INFO      pytan.handler: Report file '/var/folders/dk/vjr1r_c53yx6k6gzp2bbt_c
15   Traceback (most recent call last):
16     File "<string>", line 55, in <module>
17     File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 404, in deploy_action
18       **kwargs
19     File "/Users/jolsen/gh/pytan/lib/pytan/utils.py", line 2699, in wrap
20       ret = f(*args, **kwargs)
21     File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 1662, in _deploy_action
22       raise pytan.exceptions.RunFalse(m(report_path, len(result)))
23   RunFalse: 'Run' is not True!!
24   View and verify the contents of /var/folders/dk/vjr1r_c53yx6k6gzp2bbt_c40000gn/T/VERIFY_BEFORE_DEPLO
25   Re-run this deploy action with run=True after verifying
```

Invalid deploy action package help

Have deploy_action() return the help for package

Example Python Code

```
1   import os
2   import sys
3   sys.dont_write_bytecode = True
4
5   # Determine our script name, script dir
6   my_file = os.path.abspath(sys.argv[0])
7   my_dir = os.path.dirname(my_file)
8
9   # determine the pytan lib dir and add it to the path
10  parent_dir = os.path.dirname(my_dir)
11  pytan_root_dir = os.path.dirname(parent_dir)
12  lib_dir = os.path.join(pytan_root_dir, 'lib')
13  path_adds = [lib_dir]
14
15  for aa in path_adds:
16      if aa not in sys.path:
17          sys.path.append(aa)
18
19
20  # connection info for Tanium Server
21  USERNAME = "Tanium User"
22  PASSWORD = "T@n!um"
23  HOST = "172.16.31.128"
24  PORT = "443"
25
26  # Logging conrols
27  LOGLEVEL = 2
28  DEBUGFORMAT = False
29
30  import tempfile
31
32  import pytan
33  handler = pytan.Handler(
34      username=USERNAME,
35      password=PASSWORD,
36      host=HOST,
37      port=PORT,
38      loglevel=LOGLEVEL,
39      debugformat=DEBUGFORMAT,
40  )
41
42  print handler
43
44  # setup the arguments for the handler method
45  kwargs = {}
46  kwargs['report_dir'] = tempfile.gettempdir()
47  kwargs["package_help"] = True
48
49
50  # call the handler with the deploy_action method, passing in kwargs for arguments
51  # this should throw an exception: pytan.exceptions.PytanHelp
52  import traceback
53  try:
54      handler.deploy_action(**kwargs)
55  except Exception as e:
56      traceback.print_exc(file=sys.stdout)
```

Output from Python Code

```
1   Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2   Traceback (most recent call last):
3     File "<string>", line 55, in <module>
4     File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 374, in deploy_action
5       raise pytan.exceptions.PytanHelp(pytan.help.help_package())
6   PytanHelp:
7   Package Help
8   ============
9
10  Supplying package defines what package will be deployed as part of the
11  action.
12
13  A package string is a human string that describes, at a minimum, a
14  package. It can also optionally define a selector for the package,
15  and/or parameters for the package. A package must be provided as a string.
16
17  Examples for package
18  --------------------------------
19
20  Supplying a package:
21
22      'Distribute Tanium Standard Utilities'
23
24  Supplying a package by id:
25
26      'id:1'
27
28  Supplying a package by hash:
29
30      'hash:123456789'
31
32  Supplying a package by name:
33
34      'name:Distribute Tanium Standard Utilities'
35
36  Package Parameters
37  ------------------
38
39  Supplying parameters to a package can control the arguments
40  that are supplied to a package, if that package takes any arguments.
41
42  Package parameters must be surrounded with curly braces '{}',
43  and must have a key and value specified that is separated by
44  an equals '='. Multiple parameters must be seperated by
45  a comma ','. The key should match up to a valid parameter key
46  for the package in question.
47
48  If a parameter is supplied and the package doesn't have a
49  corresponding key name, it will be ignored. If the package has
50  parameters and a parameter is NOT supplied then an exception
51  will be raised, printing out the JSON of the missing paramater
52  for the package in question.
53
54  Examples for package with parameters
55  ------------------------------------
56
57  Supplying a package with a single parameter '$1':
```

```
58
59        'Package With Params{$1=value1}'
60
61   Supplying a package with two parameters, '$1' and '$2':
62
63        'Package With Params{$1=value1,$2=value2}'
```

Invalid deploy action package

Deploy an action using a non-existing package.

Example Python Code

```python
1    import os
2    import sys
3    sys.dont_write_bytecode = True
4
5    # Determine our script name, script dir
6    my_file = os.path.abspath(sys.argv[0])
7    my_dir = os.path.dirname(my_file)
8
9    # determine the pytan lib dir and add it to the path
10   parent_dir = os.path.dirname(my_dir)
11   pytan_root_dir = os.path.dirname(parent_dir)
12   lib_dir = os.path.join(pytan_root_dir, 'lib')
13   path_adds = [lib_dir]
14
15   for aa in path_adds:
16       if aa not in sys.path:
17           sys.path.append(aa)
18
19
20   # connection info for Tanium Server
21   USERNAME = "Tanium User"
22   PASSWORD = "T@n!um"
23   HOST = "172.16.31.128"
24   PORT = "443"
25
26   # Logging conrols
27   LOGLEVEL = 2
28   DEBUGFORMAT = False
29
30   import tempfile
31
32   import pytan
33   handler = pytan.Handler(
34       username=USERNAME,
35       password=PASSWORD,
36       host=HOST,
37       port=PORT,
38       loglevel=LOGLEVEL,
39       debugformat=DEBUGFORMAT,
40   )
41
42   print handler
```

```
43
44   # setup the arguments for the handler method
45   kwargs = {}
46   kwargs['report_dir'] = tempfile.gettempdir()
47   kwargs["run"] = True
48   kwargs["package"] = u'Invalid Package'
49
50
51   # call the handler with the deploy_action method, passing in kwargs for arguments
52   # this should throw an exception: pytan.exceptions.HandlerError
53   import traceback
54   try:
55       handler.deploy_action(**kwargs)
56   except Exception as e:
57       traceback.print_exc(file=sys.stdout)
```

Output from Python Code

```
1    Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2    Traceback (most recent call last):
3      File "<string>", line 56, in <module>
4      File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 404, in deploy_action
5        **kwargs
6      File "/Users/jolsen/gh/pytan/lib/pytan/utils.py", line 2699, in wrap
7        ret = f(*args, **kwargs)
8      File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 1611, in _deploy_action
9        package_def = self._get_package_def(package_def)
10     File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 1430, in _get_package_def
11       d['package_obj'] = self.get('package', **def_search)[0]
12     File "/Users/jolsen/gh/pytan/lib/pytan/utils.py", line 2699, in wrap
13       ret = f(*args, **kwargs)
14     File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 1255, in get
15       return self._get_single(obj_map, **kwargs)
16     File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 1390, in _get_single
17       for x in self._single_find(obj_map, k, v, **kwargs):
18     File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 1400, in _single_find
19       obj_ret = self._find(api_obj_single, **kwargs)
20     File "/Users/jolsen/gh/pytan/lib/pytan/utils.py", line 2699, in wrap
21       ret = f(*args, **kwargs)
22     File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 1327, in _find
23       raise pytan.exceptions.HandlerError(err(search_str))
24   HandlerError: No results found searching for PackageSpec, name: u'Invalid Package'!!
```

Invalid deploy action options help

Have deploy_action() return the help for options

Example Python Code

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

```
6   my_file = os.path.abspath(sys.argv[0])
7   my_dir = os.path.dirname(my_file)
8
9   # determine the pytan lib dir and add it to the path
10  parent_dir = os.path.dirname(my_dir)
11  pytan_root_dir = os.path.dirname(parent_dir)
12  lib_dir = os.path.join(pytan_root_dir, 'lib')
13  path_adds = [lib_dir]
14
15  for aa in path_adds:
16      if aa not in sys.path:
17          sys.path.append(aa)
18
19
20  # connection info for Tanium Server
21  USERNAME = "Tanium User"
22  PASSWORD = "T@n!um"
23  HOST = "172.16.31.128"
24  PORT = "443"
25
26  # Logging conrols
27  LOGLEVEL = 2
28  DEBUGFORMAT = False
29
30  import tempfile
31
32  import pytan
33  handler = pytan.Handler(
34      username=USERNAME,
35      password=PASSWORD,
36      host=HOST,
37      port=PORT,
38      loglevel=LOGLEVEL,
39      debugformat=DEBUGFORMAT,
40  )
41
42  print handler
43
44  # setup the arguments for the handler method
45  kwargs = {}
46  kwargs['report_dir'] = tempfile.gettempdir()
47  kwargs["options_help"] = True
48
49
50  # call the handler with the deploy_action method, passing in kwargs for arguments
51  # this should throw an exception: pytan.exceptions.PytanHelp
52  import traceback
53  try:
54      handler.deploy_action(**kwargs)
55  except Exception as e:
56      traceback.print_exc(file=sys.stdout)
```

Output from Python Code

```
1   Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2   Traceback (most recent call last):
3     File "<string>", line 55, in <module>
```

```
4     File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 380, in deploy_action
5       raise pytan.exceptions.PytanHelp(pytan.help.help_options())
6   PytanHelp:
7   Options Help
8   ============
9
10  Options are used for controlling how filters act. When options are
11  used as part of a sensor string, they change how the filters
12  supplied as part of that sensor operate. When options are used for
13  whole question options, they change how all of the question filters
14  operate.
15
16  When options are supplied for a sensor string, they must be
17  supplied as ', opt:OPTION' or ', opt:OPTION:VALUE' for options
18  that require a value.
19
20  When options are supplied for question options, they must be
21  supplied as 'OPTION' or 'OPTION:VALUE' for options that require
22  a value.
23
24  Options can be used on 'filter' or 'group', where 'group' pertains
25  to group filters or question filters. All 'filter' options are also
26  applicable to 'group' for question options.
27
28  Valid Options
29  -------------
30
31      'ignore_case'
32          Help: Make the filter do a case insensitive match
33          Usable on: filter
34          Example for sensor: "Sensor1, opt:ignore_case"
35          Example for question: "ignore_case"
36
37      'match_case'
38          Help: Make the filter do a case sensitive match
39          Usable on: filter
40          Example for sensor: "Sensor1, opt:match_case"
41          Example for question: "match_case"
42
43      'match_any_value'
44          Help: Make the filter match any value
45          Usable on: filter
46          Example for sensor: "Sensor1, opt:match_any_value"
47          Example for question: "match_any_value"
48
49      'match_all_values'
50          Help: Make the filter match all values
51          Usable on: filter
52          Example for sensor: "Sensor1, opt:match_all_values"
53          Example for question: "match_all_values"
54
55      'max_data_age'
56          Help: Re-fetch cached values older than N seconds
57          Usable on: filter
58          VALUE description and type: seconds, <type 'int'>
59          Example for sensor: "Sensor1, opt:max_data_age:seconds"
60          Example for question: "max_data_age:seconds"
61
```

```
62     'value_type'
63         Help: Make the filter consider the value type as VALUE_TYPE
64         Usable on: filter
65         VALUE description and type: value_type, <type 'str'>
66         Example for sensor: "Sensor1, opt:value_type:value_type"
67         Example for question: "value_type:value_type"
68
69     'and'
70         Help: Use 'and' for all of the filters supplied
71         Usable on: group
72         Example for sensor: "Sensor1, opt:and"
73         Example for question: "and"
74
75     'or'
76         Help: Use 'or' for all of the filters supplied
77         Usable on: group
78         Example for sensor: "Sensor1, opt:or"
79         Example for question: "or"
80
81     'ignore_case'
82         Help: Make the filter do a case insensitive match
83         Usable on: filter
84         Example for sensor: "Sensor1, opt:ignore_case"
85         Example for question: "ignore_case"
86
87     'match_case'
88         Help: Make the filter do a case sensitive match
89         Usable on: filter
90         Example for sensor: "Sensor1, opt:match_case"
91         Example for question: "match_case"
92
93     'match_any_value'
94         Help: Make the filter match any value
95         Usable on: filter
96         Example for sensor: "Sensor1, opt:match_any_value"
97         Example for question: "match_any_value"
98
99     'match_all_values'
100        Help: Make the filter match all values
101        Usable on: filter
102        Example for sensor: "Sensor1, opt:match_all_values"
103        Example for question: "match_all_values"
104
105    'max_data_age'
106        Help: Re-fetch cached values older than N seconds
107        Usable on: filter
108        VALUE description and type: seconds, <type 'int'>
109        Example for sensor: "Sensor1, opt:max_data_age:seconds"
110        Example for question: "max_data_age:seconds"
111
112    'value_type'
113        Help: Make the filter consider the value type as VALUE_TYPE
114        Usable on: filter
115        VALUE description and type: value_type, <type 'str'>
116        Example for sensor: "Sensor1, opt:value_type:value_type"
117        Example for question: "value_type:value_type"
118
119    'and'
```

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

Invalid deploy action empty package

Deploy an action using an empty package string.

Example Python Code

```python
import os
import sys
sys.dont_write_bytecode = True

# Determine our script name, script dir
my_file = os.path.abspath(sys.argv[0])
my_dir = os.path.dirname(my_file)

# determine the pytan lib dir and add it to the path
parent_dir = os.path.dirname(my_dir)
pytan_root_dir = os.path.dirname(parent_dir)
lib_dir = os.path.join(pytan_root_dir, 'lib')
path_adds = [lib_dir]

for aa in path_adds:
    if aa not in sys.path:
        sys.path.append(aa)


# connection info for Tanium Server
USERNAME = "Tanium User"
PASSWORD = "T@n!um"
HOST = "172.16.31.128"
PORT = "443"

# Logging conrols
LOGLEVEL = 2
DEBUGFORMAT = False

import tempfile

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

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

Output from Python Code

```
1    Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2    Traceback (most recent call last):
3      File "<string>", line 56, in <module>
4      File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 398, in deploy_action
5        package_def = pytan.utils.dehumanize_package(package)
6      File "/Users/jolsen/gh/pytan/lib/pytan/utils.py", line 1508, in dehumanize_package
7        raise pytan.exceptions.HumanParserError(err(package))
8    HumanParserError: u'' must be a string supplied as 'package'
```

Invalid deploy action filters help

Have deploy_action() return the help for filters

Example Python Code

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

```
18
19
20   # connection info for Tanium Server
21   USERNAME = "Tanium User"
22   PASSWORD = "T@n!um"
23   HOST = "172.16.31.128"
24   PORT = "443"
25
26   # Logging conrols
27   LOGLEVEL = 2
28   DEBUGFORMAT = False
29
30   import tempfile
31
32   import pytan
33   handler = pytan.Handler(
34       username=USERNAME,
35       password=PASSWORD,
36       host=HOST,
37       port=PORT,
38       loglevel=LOGLEVEL,
39       debugformat=DEBUGFORMAT,
40   )
41
42   print handler
43
44   # setup the arguments for the handler method
45   kwargs = {}
46   kwargs['report_dir'] = tempfile.gettempdir()
47   kwargs["filters_help"] = True
48
49
50   # call the handler with the deploy_action method, passing in kwargs for arguments
51   # this should throw an exception: pytan.exceptions.PytanHelp
52   import traceback
53   try:
54       handler.deploy_action(**kwargs)
55   except Exception as e:
56       traceback.print_exc(file=sys.stdout)
```

Output from Python Code

```
1    Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2    Traceback (most recent call last):
3      File "<string>", line 55, in <module>
4      File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 377, in deploy_action
5        raise pytan.exceptions.PytanHelp(pytan.help.help_filters())
6    PytanHelp:
7    Filters Help
8    ============
9
10   Filters are used generously throughout pytan. When used as part of a
11   sensor string, they control what data is shown for the columns that
12   the sensor returns. When filters are used for whole question filters,
13   they control what rows will be returned. They are used by Groups to
14   define group membership, deploy actions to determine which machines
15   should have the action deployed to it, and more.
```

```
16
17   A filter string is a human string that describes, a sensor followed
18   by ', that FILTER:VALUE', where FILTER is a valid filter string,
19   and VALUE is the string that you want FILTER to match on.
20
21   Valid Filters
22   -------------
23
24       '<'
25           Help: Filter for less than VALUE
26           Example: "Sensor1, that <:VALUE"
27
28       'less'
29           Help: Filter for less than VALUE
30           Example: "Sensor1, that less:VALUE"
31
32       'lt'
33           Help: Filter for less than VALUE
34           Example: "Sensor1, that lt:VALUE"
35
36       'less than'
37           Help: Filter for less than VALUE
38           Example: "Sensor1, that less than:VALUE"
39
40       '!<'
41           Help: Filter for not less than VALUE
42           Example: "Sensor1, that !<:VALUE"
43
44       'notless'
45           Help: Filter for not less than VALUE
46           Example: "Sensor1, that notless:VALUE"
47
48       'not less'
49           Help: Filter for not less than VALUE
50           Example: "Sensor1, that not less:VALUE"
51
52       'not less than'
53           Help: Filter for not less than VALUE
54           Example: "Sensor1, that not less than:VALUE"
55
56       '<='
57           Help: Filter for less than or equal to VALUE
58           Example: "Sensor1, that <=:VALUE"
59
60       'less equal'
61           Help: Filter for less than or equal to VALUE
62           Example: "Sensor1, that less equal:VALUE"
63
64       'lessequal'
65           Help: Filter for less than or equal to VALUE
66           Example: "Sensor1, that lessequal:VALUE"
67
68       'le'
69           Help: Filter for less than or equal to VALUE
70           Example: "Sensor1, that le:VALUE"
71
72       '!<='
73           Help: Filter for not less than or equal to VALUE
```

```
74          Example: "Sensor1, that !<=:VALUE"
75
76      'not less equal'
77          Help: Filter for not less than or equal to VALUE
78          Example: "Sensor1, that not less equal:VALUE"
79
80      'not lessequal'
81          Help: Filter for not less than or equal to VALUE
82          Example: "Sensor1, that not lessequal:VALUE"
83
84      '>'
85          Help: Filter for greater than VALUE
86          Example: "Sensor1, that >:VALUE"
87
88      'greater'
89          Help: Filter for greater than VALUE
90          Example: "Sensor1, that greater:VALUE"
91
92      'gt'
93          Help: Filter for greater than VALUE
94          Example: "Sensor1, that gt:VALUE"
95
96      'greater than'
97          Help: Filter for greater than VALUE
98          Example: "Sensor1, that greater than:VALUE"
99
100     '!>'
101         Help: Filter for not greater than VALUE
102         Example: "Sensor1, that !>:VALUE"
103
104     'not greater'
105         Help: Filter for not greater than VALUE
106         Example: "Sensor1, that not greater:VALUE"
107
108     'notgreater'
109         Help: Filter for not greater than VALUE
110         Example: "Sensor1, that notgreater:VALUE"
111
112     'not greater than'
113         Help: Filter for not greater than VALUE
114         Example: "Sensor1, that not greater than:VALUE"
115
116     '=>'
117         Help: Filter for greater than or equal to VALUE
118         Example: "Sensor1, that =>:VALUE"
119
120     'greater equal'
121         Help: Filter for greater than or equal to VALUE
122         Example: "Sensor1, that greater equal:VALUE"
123
124     'greaterequal'
125         Help: Filter for greater than or equal to VALUE
126         Example: "Sensor1, that greaterequal:VALUE"
127
128     'ge'
129         Help: Filter for greater than or equal to VALUE
130         Example: "Sensor1, that ge:VALUE"
131
```

```
132     '!=>'
133         Help: Filter for not greater than VALUE
134         Example: "Sensor1, that !=>:VALUE"
135
136     'not greater equal'
137         Help: Filter for not greater than VALUE
138         Example: "Sensor1, that not greater equal:VALUE"
139
140     'notgreaterequal'
141         Help: Filter for not greater than VALUE
142         Example: "Sensor1, that notgreaterequal:VALUE"
143
144     '='
145         Help: Filter for equals to VALUE
146         Example: "Sensor1, that =:VALUE"
147
148     'equal'
149         Help: Filter for equals to VALUE
150         Example: "Sensor1, that equal:VALUE"
151
152     'equals'
153         Help: Filter for equals to VALUE
154         Example: "Sensor1, that equals:VALUE"
155
156     'eq'
157         Help: Filter for equals to VALUE
158         Example: "Sensor1, that eq:VALUE"
159
160     '!='
161         Help: Filter for not equals to VALUE
162         Example: "Sensor1, that !=:VALUE"
163
164     'not equal'
165         Help: Filter for not equals to VALUE
166         Example: "Sensor1, that not equal:VALUE"
167
168     'notequal'
169         Help: Filter for not equals to VALUE
170         Example: "Sensor1, that notequal:VALUE"
171
172     'not equals'
173         Help: Filter for not equals to VALUE
174         Example: "Sensor1, that not equals:VALUE"
175
176     'notequals'
177         Help: Filter for not equals to VALUE
178         Example: "Sensor1, that notequals:VALUE"
179
180     'ne'
181         Help: Filter for not equals to VALUE
182         Example: "Sensor1, that ne:VALUE"
183
184     'contains'
185         Help: Filter for contains VALUE (adds .* before and after VALUE)
186         Example: "Sensor1, that contains:VALUE"
187
188     'does not contain'
189         Help: Filter for does not contain VALUE (adds .* before and after VALUE)
```

```
190          Example: "Sensor1, that does not contain:VALUE"
191
192      'doesnotcontain'
193          Help: Filter for does not contain VALUE (adds .* before and after VALUE)
194          Example: "Sensor1, that doesnotcontain:VALUE"
195
196      'not contains'
197          Help: Filter for does not contain VALUE (adds .* before and after VALUE)
198          Example: "Sensor1, that not contains:VALUE"
199
200      'notcontains'
201          Help: Filter for does not contain VALUE (adds .* before and after VALUE)
202          Example: "Sensor1, that notcontains:VALUE"
203
204      'starts with'
205          Help: Filter for starts with VALUE (adds .* after VALUE)
206          Example: "Sensor1, that starts with:VALUE"
207
208      'startswith'
209          Help: Filter for starts with VALUE (adds .* after VALUE)
210          Example: "Sensor1, that startswith:VALUE"
211
212      'does not start with'
213          Help: Filter for does not start with VALUE (adds .* after VALUE)
214          Example: "Sensor1, that does not start with:VALUE"
215
216      'doesnotstartwith'
217          Help: Filter for does not start with VALUE (adds .* after VALUE)
218          Example: "Sensor1, that doesnotstartwith:VALUE"
219
220      'not starts with'
221          Help: Filter for does not start with VALUE (adds .* after VALUE)
222          Example: "Sensor1, that not starts with:VALUE"
223
224      'notstartswith'
225          Help: Filter for does not start with VALUE (adds .* after VALUE)
226          Example: "Sensor1, that notstartswith:VALUE"
227
228      'ends with'
229          Help: Filter for ends with VALUE (adds .* before VALUE)
230          Example: "Sensor1, that ends with:VALUE"
231
232      'endswith'
233          Help: Filter for ends with VALUE (adds .* before VALUE)
234          Example: "Sensor1, that endswith:VALUE"
235
236      'does not end with'
237          Help: Filter for does bit end with VALUE (adds .* before VALUE)
238          Example: "Sensor1, that does not end with:VALUE"
239
240      'doesnotendwith'
241          Help: Filter for does bit end with VALUE (adds .* before VALUE)
242          Example: "Sensor1, that doesnotendwith:VALUE"
243
244      'not ends with'
245          Help: Filter for does bit end with VALUE (adds .* before VALUE)
246          Example: "Sensor1, that not ends with:VALUE"
247
```

```
248         'notstartswith'
249             Help: Filter for does bit end with VALUE (adds .* before VALUE)
250             Example: "Sensor1, that notstartswith:VALUE"
251
252         'is not'
253             Help: Filter for non regular expression match for VALUE
254             Example: "Sensor1, that is not:VALUE"
255
256         'not regex'
257             Help: Filter for non regular expression match for VALUE
258             Example: "Sensor1, that not regex:VALUE"
259
260         'notregex'
261             Help: Filter for non regular expression match for VALUE
262             Example: "Sensor1, that notregex:VALUE"
263
264         'not regex match'
265             Help: Filter for non regular expression match for VALUE
266             Example: "Sensor1, that not regex match:VALUE"
267
268         'notregexmatch'
269             Help: Filter for non regular expression match for VALUE
270             Example: "Sensor1, that notregexmatch:VALUE"
271
272         'nre'
273             Help: Filter for non regular expression match for VALUE
274             Example: "Sensor1, that nre:VALUE"
275
276         'is'
277             Help: Filter for regular expression match for VALUE
278             Example: "Sensor1, that is:VALUE"
279
280         'regex'
281             Help: Filter for regular expression match for VALUE
282             Example: "Sensor1, that regex:VALUE"
283
284         'regex match'
285             Help: Filter for regular expression match for VALUE
286             Example: "Sensor1, that regex match:VALUE"
287
288         'regexmatch'
289             Help: Filter for regular expression match for VALUE
290             Example: "Sensor1, that regexmatch:VALUE"
291
292         're'
293             Help: Filter for regular expression match for VALUE
294             Example: "Sensor1, that re:VALUE"
295
296         '<'
297             Help: Filter for less than VALUE
298             Example: "Sensor1, that <:VALUE"
299
300         'less'
301             Help: Filter for less than VALUE
302             Example: "Sensor1, that less:VALUE"
303
304         'lt'
305             Help: Filter for less than VALUE
```

```
306          Example: "Sensor1, that lt:VALUE"
307
308      'less than'
309          Help: Filter for less than VALUE
310          Example: "Sensor1, that less than:VALUE"
311
312      '!<'
313          Help: Filter for not less than VALUE
314          Example: "Sensor1, that !<:VALUE"
315
316      'notless'
317          Help: Filter for not less than VALUE
318          Example: "Sensor1, that notless:VALUE"
319
320      'not less'
321          Help: Filter for not less than VALUE
322          Example: "Sensor1, that not less:VALUE"
323
324      'not less than'
325          Help: Filter for not less than VALUE
326          Example: "Sensor1, that not less than:VALUE"
327
328      '<='
329          Help: Filter for less than or equal to VALUE
330          Example: "Sensor1, that <=:VALUE"
331
332      'less equal'
333          Help: Filter for less than or equal to VALUE
334          Example: "Sensor1, that less equal:VALUE"
335
336      'lessequal'
337          Help: Filter for less than or equal to VALUE
338          Example: "Sensor1, that lessequal:VALUE"
339
340      'le'
341          Help: Filter for less than or equal to VALUE
342          Example: "Sensor1, that le:VALUE"
343
344      '!<='
345          Help: Filter for not less than or equal to VALUE
346          Example: "Sensor1, that !<=:VALUE"
347
348      'not less equal'
349          Help: Filter for not less than or equal to VALUE
350          Example: "Sensor1, that not less equal:VALUE"
351
352      'not lessequal'
353          Help: Filter for not less than or equal to VALUE
354          Example: "Sensor1, that not lessequal:VALUE"
355
356      '>'
357          Help: Filter for greater than VALUE
358          Example: "Sensor1, that >:VALUE"
359
360      'greater'
361          Help: Filter for greater than VALUE
362          Example: "Sensor1, that greater:VALUE"
363
```

```
364        'gt'
365            Help: Filter for greater than VALUE
366            Example: "Sensor1, that gt:VALUE"
367
368        'greater than'
369            Help: Filter for greater than VALUE
370            Example: "Sensor1, that greater than:VALUE"
371
372        '!>'
373            Help: Filter for not greater than VALUE
374            Example: "Sensor1, that !>:VALUE"
375
376        'not greater'
377            Help: Filter for not greater than VALUE
378            Example: "Sensor1, that not greater:VALUE"
379
380        'notgreater'
381            Help: Filter for not greater than VALUE
382            Example: "Sensor1, that notgreater:VALUE"
383
384        'not greater than'
385            Help: Filter for not greater than VALUE
386            Example: "Sensor1, that not greater than:VALUE"
387
388        '=>'
389            Help: Filter for greater than or equal to VALUE
390            Example: "Sensor1, that =>:VALUE"
391
392        'greater equal'
393            Help: Filter for greater than or equal to VALUE
394            Example: "Sensor1, that greater equal:VALUE"
395
396        'greaterequal'
397            Help: Filter for greater than or equal to VALUE
398            Example: "Sensor1, that greaterequal:VALUE"
399
400        'ge'
401            Help: Filter for greater than or equal to VALUE
402            Example: "Sensor1, that ge:VALUE"
403
404        '!=>'
405            Help: Filter for not greater than VALUE
406            Example: "Sensor1, that !=>:VALUE"
407
408        'not greater equal'
409            Help: Filter for not greater than VALUE
410            Example: "Sensor1, that not greater equal:VALUE"
411
412        'notgreaterequal'
413            Help: Filter for not greater than VALUE
414            Example: "Sensor1, that notgreaterequal:VALUE"
415
416        '='
417            Help: Filter for equals to VALUE
418            Example: "Sensor1, that =:VALUE"
419
420        'equal'
421            Help: Filter for equals to VALUE
```

```
422         Example: "Sensor1, that equal:VALUE"
423
424     'equals'
425         Help: Filter for equals to VALUE
426         Example: "Sensor1, that equals:VALUE"
427
428     'eq'
429         Help: Filter for equals to VALUE
430         Example: "Sensor1, that eq:VALUE"
431
432     '!='
433         Help: Filter for not equals to VALUE
434         Example: "Sensor1, that !=:VALUE"
435
436     'not equal'
437         Help: Filter for not equals to VALUE
438         Example: "Sensor1, that not equal:VALUE"
439
440     'notequal'
441         Help: Filter for not equals to VALUE
442         Example: "Sensor1, that notequal:VALUE"
443
444     'not equals'
445         Help: Filter for not equals to VALUE
446         Example: "Sensor1, that not equals:VALUE"
447
448     'notequals'
449         Help: Filter for not equals to VALUE
450         Example: "Sensor1, that notequals:VALUE"
451
452     'ne'
453         Help: Filter for not equals to VALUE
454         Example: "Sensor1, that ne:VALUE"
455
456     'contains'
457         Help: Filter for contains VALUE (adds .* before and after VALUE)
458         Example: "Sensor1, that contains:VALUE"
459
460     'does not contain'
461         Help: Filter for does not contain VALUE (adds .* before and after VALUE)
462         Example: "Sensor1, that does not contain:VALUE"
463
464     'doesnotcontain'
465         Help: Filter for does not contain VALUE (adds .* before and after VALUE)
466         Example: "Sensor1, that doesnotcontain:VALUE"
467
468     'not contains'
469         Help: Filter for does not contain VALUE (adds .* before and after VALUE)
470         Example: "Sensor1, that not contains:VALUE"
471
472     'notcontains'
473         Help: Filter for does not contain VALUE (adds .* before and after VALUE)
474         Example: "Sensor1, that notcontains:VALUE"
475
476     'starts with'
477         Help: Filter for starts with VALUE (adds .* after VALUE)
478         Example: "Sensor1, that starts with:VALUE"
479
```

```
480        'startswith'
481            Help: Filter for starts with VALUE (adds .* after VALUE)
482            Example: "Sensor1, that startswith:VALUE"
483
484        'does not start with'
485            Help: Filter for does not start with VALUE (adds .* after VALUE)
486            Example: "Sensor1, that does not start with:VALUE"
487
488        'doesnotstartwith'
489            Help: Filter for does not start with VALUE (adds .* after VALUE)
490            Example: "Sensor1, that doesnotstartwith:VALUE"
491
492        'not starts with'
493            Help: Filter for does not start with VALUE (adds .* after VALUE)
494            Example: "Sensor1, that not starts with:VALUE"
495
496        'notstartswith'
497            Help: Filter for does not start with VALUE (adds .* after VALUE)
498            Example: "Sensor1, that notstartswith:VALUE"
499
500        'ends with'
501            Help: Filter for ends with VALUE (adds .* before VALUE)
502            Example: "Sensor1, that ends with:VALUE"
503
504        'endswith'
505            Help: Filter for ends with VALUE (adds .* before VALUE)
506            Example: "Sensor1, that endswith:VALUE"
507
508        'does not end with'
509            Help: Filter for does bit end with VALUE (adds .* before VALUE)
510            Example: "Sensor1, that does not end with:VALUE"
511
512        'doesnotendwith'
513            Help: Filter for does bit end with VALUE (adds .* before VALUE)
514            Example: "Sensor1, that doesnotendwith:VALUE"
515
516        'not ends with'
517            Help: Filter for does bit end with VALUE (adds .* before VALUE)
518            Example: "Sensor1, that not ends with:VALUE"
519
520        'notstartswith'
521            Help: Filter for does bit end with VALUE (adds .* before VALUE)
522            Example: "Sensor1, that notstartswith:VALUE"
523
524        'is not'
525            Help: Filter for non regular expression match for VALUE
526            Example: "Sensor1, that is not:VALUE"
527
528        'not regex'
529            Help: Filter for non regular expression match for VALUE
530            Example: "Sensor1, that not regex:VALUE"
531
532        'notregex'
533            Help: Filter for non regular expression match for VALUE
534            Example: "Sensor1, that notregex:VALUE"
535
536        'not regex match'
537            Help: Filter for non regular expression match for VALUE
```

```
538          Example: "Sensor1, that not regex match:VALUE"

539

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

543

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

547

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

551

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

555

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

559

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

563

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

Invalid deploy action missing parameters

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

Example Python Code

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

```
20    # connection info for Tanium Server
21    USERNAME = "Tanium User"
22    PASSWORD = "T@n!um"
23    HOST = "172.16.31.128"
24    PORT = "443"
25
26    # Logging conrols
27    LOGLEVEL = 2
28    DEBUGFORMAT = False
29
30    import tempfile
31
32    import pytan
33    handler = pytan.Handler(
34        username=USERNAME,
35        password=PASSWORD,
36        host=HOST,
37        port=PORT,
38        loglevel=LOGLEVEL,
39        debugformat=DEBUGFORMAT,
40    )
41
42    print handler
43
44    # setup the arguments for the handler method
45    kwargs = {}
46    kwargs['report_dir'] = tempfile.gettempdir()
47    kwargs["run"] = True
48    kwargs["package"] = u'Custom Tagging - Add Tags'
49
50
51    # call the handler with the deploy_action method, passing in kwargs for arguments
52    # this should throw an exception: pytan.exceptions.HandlerError
53    import traceback
54    try:
55        handler.deploy_action(**kwargs)
56    except Exception as e:
57        traceback.print_exc(file=sys.stdout)
```

Output from Python Code

```
1     Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2     Traceback (most recent call last):
3       File "<string>", line 56, in <module>
4       File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 404, in deploy_action
5         **kwargs
6       File "/Users/jolsen/gh/pytan/lib/pytan/utils.py", line 2699, in wrap
7         ret = f(*args, **kwargs)
8       File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 1670, in _deploy_action
9         empty_ok=False,
10      File "/Users/jolsen/gh/pytan/lib/pytan/utils.py", line 2329, in build_param_objlist
11        raise pytan.exceptions.HandlerError(err(obj_name, p_key, jsonify(obj_param)))
12    HandlerError: PackageSpec, name: 'Custom Tagging - Add Tags', id: 27 parameter key '$1' requires a v
13    {
14      "defaultValue": "",
15      "helpString": "Enter tags space-delimited.",
16      "key": "$1",
```

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

PyTan API Valid Create Object Examples

Create user

Create a user called API Test User

Example Python Code

```python
1   import os
2   import sys
3   sys.dont_write_bytecode = True
4
5   # Determine our script name, script dir
6   my_file = os.path.abspath(sys.argv[0])
7   my_dir = os.path.dirname(my_file)
8
9   # determine the pytan lib dir and add it to the path
10  parent_dir = os.path.dirname(my_dir)
11  pytan_root_dir = os.path.dirname(parent_dir)
12  lib_dir = os.path.join(pytan_root_dir, 'lib')
13  path_adds = [lib_dir]
14
15  for aa in path_adds:
16      if aa not in sys.path:
17          sys.path.append(aa)
18
19
20  # connection info for Tanium Server
21  USERNAME = "Tanium User"
22  PASSWORD = "T@n!um"
23  HOST = "172.16.31.128"
24  PORT = "443"
25
26  # Logging conrols
27  LOGLEVEL = 2
28  DEBUGFORMAT = False
29
```

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

Output from Python Code

```
1  Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2  No results found searching for user with {'name': 'API Test User'}!!
```

```
3   2015-08-07 19:46:14,085 INFO     pytan.handler: New user 'API Test User' created with ID 15, roles:
4
5   Type of response:  <class 'taniumpy.object_types.user.User'>
6
7   print of response:
8   User, name: 'API Test User', id: 15
9
10  print the object returned in JSON format:
11  {
12    "_type": "user",
13    "deleted_flag": 0,
14    "group_id": 0,
15    "id": 15,
16    "last_login": "2001-01-01T00:00:00",
17    "local_admin_flag": -1,
18    "metadata": {
19      "_type": "metadata",
20      "item": [
21        {
22          "_type": "item",
23          "admin_flag": 0,
24          "name": "TConsole.User.Property.property1",
25          "value": "value1"
26        }
27      ]
28    },
29    "name": "API Test User",
30    "permissions": {
31      "_type": "permissions",
32      "permission": [
33        "admin",
34        "sensor_read",
35        "sensor_write",
36        "question_read",
37        "question_write",
38        "action_read",
39        "action_write",
40        "action_approval",
41        "notification_write",
42        "clients_read",
43        "question_log_read",
44        "content_admin"
45      ]
46    },
47    "roles": {
48      "_type": "roles",
49      "role": [
50        {
51          "_type": "role",
52          "description": "Administrators can perform all functions in the system, including creating o
53          "id": 1,
54          "name": "Administrator",
55          "permissions": {
56            "_type": "permissions",
57            "permission": [
58              "admin",
59              "sensor_read",
60              "sensor_write",
```

```
61              "question_read",
62              "question_write",
63              "action_read",
64              "action_write",
65              "action_approval",
66              "notification_write",
67              "clients_read",
68              "question_log_read",
69              "content_admin"
70            ]
71          }
72        }
73      ]
74    }
75  }
76  2015-08-07 19:46:14,099 INFO     pytan.handler: Deleted "User, name: 'API Test User', id: 15"
```

Create package

Create a package called package49

Example Python Code

```python
1   import os
2   import sys
3   sys.dont_write_bytecode = True
4
5   # Determine our script name, script dir
6   my_file = os.path.abspath(sys.argv[0])
7   my_dir = os.path.dirname(my_file)
8
9   # determine the pytan lib dir and add it to the path
10  parent_dir = os.path.dirname(my_dir)
11  pytan_root_dir = os.path.dirname(parent_dir)
12  lib_dir = os.path.join(pytan_root_dir, 'lib')
13  path_adds = [lib_dir]
14
15  for aa in path_adds:
16      if aa not in sys.path:
17          sys.path.append(aa)
18
19
20  # connection info for Tanium Server
21  USERNAME = "Tanium User"
22  PASSWORD = "T@n!um"
23  HOST = "172.16.31.128"
24  PORT = "443"
25
26  # Logging conrols
27  LOGLEVEL = 2
28  DEBUGFORMAT = False
29
30  import tempfile
31
32  import pytan
```

```
33   handler = pytan.Handler(
34       username=USERNAME,
35       password=PASSWORD,
36       host=HOST,
37       port=PORT,
38       loglevel=LOGLEVEL,
39       debugformat=DEBUGFORMAT,
40   )
41
42   print handler
43
44   # setup the arguments for the delete method (to remove the package in case it exists)
45   delete_kwargs = {}
46   delete_kwargs["objtype"] = 'package'
47   delete_kwargs["name"] = 'package49'
48
49
50   # setup the arguments for the handler method
51   kwargs = {}
52   kwargs["expire_seconds"] = 1500
53   kwargs["display_name"] = u'package49 API test'
54   kwargs["name"] = u'package49'
55   kwargs["parameters_json_file"] = u'../doc/example_of_all_package_parameters.json'
56   kwargs["verify_expire_seconds"] = 3600
57   kwargs["command"] = u'package49 $1 $2 $3 $4 $5 $6 $7 $8'
58   kwargs["file_urls"] = [u'3600::testing.vbs||https://content.tanium.com/files/initialcontent/bundles/
59   kwargs["verify_filter_options"] = [u'and']
60   kwargs["verify_filters"] = [u'Custom Tags, that contains:tag']
61   kwargs["command_timeout_seconds"] = 9999
62
63   # delete the object in case it already exists
64   try:
65       handler.delete(**delete_kwargs)
66   except Exception as e:
67       print e
68
69   # call the handler with the create_package method, passing in kwargs for arguments
70   response = handler.create_package(**kwargs)
71
72
73   print ""
74   print "Type of response: ", type(response)
75
76   print ""
77   print "print of response:"
78   print response
79
80   print ""
81   print "print the object returned in JSON format:"
82   print response.to_json(response)
83
84   # delete the object, we are done with it now
85   try:
86       handler.delete(**delete_kwargs)
87   except Exception as e:
88       print e
```

Output from Python Code

```
1   Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2   No results found searching for PackageSpec, name: 'package49'!!
3   2015-08-07 19:46:14,165 INFO     pytan.handler: New package 'package49' created with ID 82, command:
4
5   Type of response:  <class 'taniumpy.object_types.package_spec.PackageSpec'>
6
7   print of response:
8   PackageSpec, name: 'package49', id: 82
9
10  print the object returned in JSON format:
11  {
12    "_type": "package_spec",
13    "available_time": "2001-01-01T00:00:00",
14    "command": "package49 $1 $2 $3 $4 $5 $6 $7 $8",
15    "command_timeout": 9999,
16    "creation_time": "2001-01-01T00:00:00",
17    "deleted_flag": 0,
18    "display_name": "package49 API test",
19    "expire_seconds": 1500,
20    "files": {
21      "_type": "package_files",
22      "file": [
23        {
24          "_type": "file",
25          "bytes_downloaded": 0,
26          "bytes_total": 0,
27          "download_seconds": 3600,
28          "file_status": {
29            "_type": "file_status",
30            "status": [
31              {
32                "_type": "status",
33                "bytes_downloaded": 0,
34                "bytes_total": 0,
35                "cache_status": "Processing",
36                "server_id": 1,
37                "server_name": "JTANIUM1.localdomain:17472",
38                "status": 0
39              }
40            ]
41          },
42          "id": 184,
43          "name": "testing.vbs",
44          "size": 0,
45          "source": "https://content.tanium.com/files/initialcontent/bundles/2014-10-01_11-32-15-7844/
46          "status": 0
47        }
48      ]
49    },
50    "hidden_flag": 0,
51    "id": 82,
52    "last_update": "2001-01-01T00:00:00",
53    "modification_time": "2001-01-01T00:00:00",
54    "name": "package49",
55    "parameter_definition": "{\"parameterType\": \"com.tanium.components.parameters::ParametersArray\"
56    "skip_lock_flag": 0,
57    "source_id": 0,
```

```
58      "verify_expire_seconds": 3600,
59      "verify_group": {
60        "_type": "group",
61        "id": 211
62      },
63      "verify_group_id": 211
64    }
65    2015-08-07 19:46:14,174 INFO     pytan.handler: Deleted 'PackageSpec, id: 82'
```

Create group

Create a group called All Windows Computers API Test

Example Python Code

```python
1   import os
2   import sys
3   sys.dont_write_bytecode = True
4
5   # Determine our script name, script dir
6   my_file = os.path.abspath(sys.argv[0])
7   my_dir = os.path.dirname(my_file)
8
9   # determine the pytan lib dir and add it to the path
10  parent_dir = os.path.dirname(my_dir)
11  pytan_root_dir = os.path.dirname(parent_dir)
12  lib_dir = os.path.join(pytan_root_dir, 'lib')
13  path_adds = [lib_dir]
14
15  for aa in path_adds:
16      if aa not in sys.path:
17          sys.path.append(aa)
18
19
20  # connection info for Tanium Server
21  USERNAME = "Tanium User"
22  PASSWORD = "T@n!um"
23  HOST = "172.16.31.128"
24  PORT = "443"
25
26  # Logging conrols
27  LOGLEVEL = 2
28  DEBUGFORMAT = False
29
30  import tempfile
31
32  import pytan
33  handler = pytan.Handler(
34      username=USERNAME,
35      password=PASSWORD,
36      host=HOST,
37      port=PORT,
38      loglevel=LOGLEVEL,
39      debugformat=DEBUGFORMAT,
40  )
```

```
41
42   print handler
43
44   # setup the arguments for the delete method (to remove the package in case it exists)
45   delete_kwargs = {}
46   delete_kwargs["objtype"] = 'group'
47   delete_kwargs["name"] = 'All Windows Computers API Test'
48
49
50   # setup the arguments for the handler method
51   kwargs = {}
52   kwargs["groupname"] = u'All Windows Computers API Test'
53   kwargs["filters"] = [u'Operating System, that contains:Windows']
54   kwargs["filter_options"] = [u'and']
55
56   # delete the object in case it already exists
57   try:
58       handler.delete(**delete_kwargs)
59   except Exception as e:
60       print e
61
62   # call the handler with the create_group method, passing in kwargs for arguments
63   response = handler.create_group(**kwargs)
64
65
66   print ""
67   print "Type of response: ", type(response)
68
69   print ""
70   print "print of response:"
71   print response
72
73   print ""
74   print "print the object returned in JSON format:"
75   print response.to_json(response)
76
77   # delete the object, we are done with it now
78   try:
79       handler.delete(**delete_kwargs)
80   except Exception as e:
81       print e
```

Output from Python Code

```
1   Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2   No results found searching for Group, name: 'All Windows Computers API Test'!!
3   2015-08-07 19:46:14,211 INFO     pytan.handler: New group 'All Windows Computers API Test' created w
4
5   Type of response:  <class 'taniumpy.object_types.group.Group'>
6
7   print of response:
8   Group, name: 'All Windows Computers API Test', id: 212
9
10  print the object returned in JSON format:
11  {
12    "_type": "group",
13    "and_flag": 1,
```

```
14      "deleted_flag": 0,
15      "filters": {
16        "_type": "filters",
17        "filter": [
18          {
19            "_type": "filter",
20            "all_times_flag": 0,
21            "all_values_flag": 0,
22            "delimiter_index": 0,
23            "ignore_case_flag": 1,
24            "max_age_seconds": 0,
25            "not_flag": 0,
26            "operator": "RegexMatch",
27            "sensor": {
28              "_type": "sensor",
29              "hash": 45421433
30            },
31            "substring_flag": 0,
32            "substring_length": 0,
33            "substring_start": 0,
34            "utf8_flag": 0,
35            "value": ".*Windows.*",
36            "value_type": "String"
37          }
38        ]
39      },
40      "id": 212,
41      "name": "All Windows Computers API Test",
42      "not_flag": 0,
43      "sub_groups": {
44        "_type": "groups",
45        "group": []
46      },
47      "text": " Operating System containing \"Windows\"",
48      "type": 0
49    }
50    2015-08-07 19:46:14,219 INFO     pytan.handler: Deleted 'Group, id: 212'
```

Create whitelisted url

Create a whitelisted url

Example Python Code

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

```
12  lib_dir = os.path.join(pytan_root_dir, 'lib')
13  path_adds = [lib_dir]
14
15  for aa in path_adds:
16      if aa not in sys.path:
17          sys.path.append(aa)
18
19
20  # connection info for Tanium Server
21  USERNAME = "Tanium User"
22  PASSWORD = "T@n!um"
23  HOST = "172.16.31.128"
24  PORT = "443"
25
26  # Logging conrols
27  LOGLEVEL = 2
28  DEBUGFORMAT = False
29
30  import tempfile
31
32  import pytan
33  handler = pytan.Handler(
34      username=USERNAME,
35      password=PASSWORD,
36      host=HOST,
37      port=PORT,
38      loglevel=LOGLEVEL,
39      debugformat=DEBUGFORMAT,
40  )
41
42  print handler
43
44  # setup the arguments for the delete method (to remove the package in case it exists)
45  delete_kwargs = {}
46  delete_kwargs["objtype"] = 'whitelisted_url'
47  delete_kwargs["url_regex"] = 'regex:http://test.com/.*API_Test.*URL'
48
49
50  # setup the arguments for the handler method
51  kwargs = {}
52  kwargs["url"] = u'http://test.com/.*API_Test.*URL'
53  kwargs["regex"] = True
54  kwargs["properties"] = [[u'property1', u'value1']]
55  kwargs["download_seconds"] = 3600
56
57  # delete the object in case it already exists
58  try:
59      handler.delete(**delete_kwargs)
60  except Exception as e:
61      print e
62
63  # call the handler with the create_whitelisted_url method, passing in kwargs for arguments
64  response = handler.create_whitelisted_url(**kwargs)
65
66
67  print ""
68  print "Type of response: ", type(response)
69
```

```
70  print ""
71  print "print of response:"
72  print response
73
74  print ""
75  print "print the object returned in JSON format:"
76  print response.to_json(response)
77
78  # delete the object, we are done with it now
79  try:
80      handler.delete(**delete_kwargs)
81  except Exception as e:
82      print e
```

Output from Python Code

```
1   Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2   No results found searching for whitelisted_url with {'url_regex': 'regex:http://test.com/.*API_Test.
3   2015-08-07 19:46:14,252 INFO     pytan.handler: New Whitelisted URL 'regex:http://test.com/.*API_Tes
4
5   Type of response:  <class 'taniumpy.object_types.white_listed_url.WhiteListedUrl'>
6
7   print of response:
8   WhiteListedUrl, id: 52
9
10  print the object returned in JSON format:
11  {
12    "_type": "white_listed_url",
13    "download_seconds": 3600,
14    "id": 52,
15    "metadata": {
16      "_type": "metadata",
17      "item": [
18        {
19          "_type": "item",
20          "admin_flag": 0,
21          "name": "TConsole.WhitelistedURL.property1",
22          "value": "value1"
23        }
24      ]
25    },
26    "url_regex": "regex:http://test.com/.*API_Test.*URL"
27  }
28  2015-08-07 19:46:14,263 INFO     pytan.handler: Deleted 'WhiteListedUrl, id: 52'
```

PyTan API Invalid Create Object Examples

Invalid create sensor

Create a sensor (Unsupported!)

Example Python Code

```python
1  import os
2  import sys
3  sys.dont_write_bytecode = True
4
5  # Determine our script name, script dir
6  my_file = os.path.abspath(sys.argv[0])
7  my_dir = os.path.dirname(my_file)
8
9  # determine the pytan lib dir and add it to the path
10 parent_dir = os.path.dirname(my_dir)
11 pytan_root_dir = os.path.dirname(parent_dir)
12 lib_dir = os.path.join(pytan_root_dir, 'lib')
13 path_adds = [lib_dir]
14
15 for aa in path_adds:
16     if aa not in sys.path:
17         sys.path.append(aa)
18
19
20 # connection info for Tanium Server
21 USERNAME = "Tanium User"
22 PASSWORD = "T@n!um"
23 HOST = "172.16.31.128"
24 PORT = "443"
25
26 # Logging conrols
27 LOGLEVEL = 2
28 DEBUGFORMAT = False
29
30 import tempfile
31
32 import pytan
33 handler = pytan.Handler(
34     username=USERNAME,
35     password=PASSWORD,
36     host=HOST,
37     port=PORT,
38     loglevel=LOGLEVEL,
39     debugformat=DEBUGFORMAT,
40 )
41
42 print handler
43
44 # setup the arguments for the handler method
45 kwargs = {}
46
47
48 # call the handler with the create_sensor method, passing in kwargs for arguments
49 # this should throw an exception: pytan.exceptions.HandlerError
50 import traceback
51 try:
52     handler.create_sensor(**kwargs)
53 except Exception as e:
54     traceback.print_exc(file=sys.stdout)
```

Output from Python Code

```
1   Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2   Traceback (most recent call last):
3     File "<string>", line 53, in <module>
4     File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 708, in create_sensor
5       raise pytan.exceptions.HandlerError(m)
6   HandlerError: Sensor creation not supported via PyTan as of yet, too complex
7   Use create_sensor_from_json() instead!
```

PyTan API Valid Create Object From JSON Examples

Create package from json

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

Example Python Code

```python
1   import os
2   import sys
3   sys.dont_write_bytecode = True
4
5   # Determine our script name, script dir
6   my_file = os.path.abspath(sys.argv[0])
7   my_dir = os.path.dirname(my_file)
8
9   # determine the pytan lib dir and add it to the path
10  parent_dir = os.path.dirname(my_dir)
11  pytan_root_dir = os.path.dirname(parent_dir)
12  lib_dir = os.path.join(pytan_root_dir, 'lib')
13  path_adds = [lib_dir]
14
15  for aa in path_adds:
16      if aa not in sys.path:
17          sys.path.append(aa)
18
19
20  # connection info for Tanium Server
21  USERNAME = "Tanium User"
22  PASSWORD = "T@n!um"
23  HOST = "172.16.31.128"
24  PORT = "443"
25
26  # Logging conrols
27  LOGLEVEL = 2
28  DEBUGFORMAT = False
29
30  import tempfile
31
32  import pytan
33  handler = pytan.Handler(
34      username=USERNAME,
35      password=PASSWORD,
36      host=HOST,
```

```
37          port=PORT,
38          loglevel=LOGLEVEL,
39          debugformat=DEBUGFORMAT,
40      )
41
42      print handler
43
44      # set the attribute name and value we want to add to the original object (if any)
45      attr_name = "name"
46      attr_add = " API TEST"
47
48      # delete object before creating it?
49      delete = True
50
51      # setup the arguments for getting an object to export as json file
52      get_kwargs = {}
53      get_kwargs["objtype"] = u'package'
54      get_kwargs["id"] = 31
55
56
57      # get objects to use as an export to JSON file
58      orig_objs = handler.get(**get_kwargs)
59
60      # if attr_name and attr_add exists, modify the orig_objs to add attr_add to the attribute
61      # attr_name
62      if attr_name:
63          for x in orig_objs:
64              new_attr = getattr(x, attr_name)
65              new_attr += attr_add
66              setattr(x, attr_name, new_attr)
67              if delete:
68                  # delete the object in case it already exists
69                  del_kwargs = {}
70                  del_kwargs[attr_name] = new_attr
71                  del_kwargs['objtype'] = u'package'
72                  try:
73                      handler.delete(**del_kwargs)
74                  except Exception as e:
75                      print e
76
77      # export orig_objs to a json file
78      json_file, results = handler.export_to_report_file(
79          obj=orig_objs,
80          export_format='json',
81          report_dir=tempfile.gettempdir(),
82      )
83
84      # create the object from the exported JSON file
85      create_kwargs = {'objtype': u'package', 'json_file': json_file}
86      response = handler.create_from_json(**create_kwargs)
87
88
89      print ""
90      print "Type of response: ", type(response)
91
92      print ""
93      print "print of response:"
94      print response
```

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

Output from Python Code

```
1   Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2   2015-08-07 19:46:14,304 INFO     pytan.handler: Deleted 'PackageSpec, id: 76'
3   2015-08-07 19:46:14,305 INFO     pytan.handler: Report file '/var/folders/dk/vjr1r_c53yx6k6gzp2bbt_c
4   2015-08-07 19:46:14,337 INFO     pytan.handler: New PackageSpec, name: 'Disable Java Auto Update API
5
6   Type of response:  <class 'taniumpy.object_types.package_spec_list.PackageSpecList'>
7
8   print of response:
9   PackageSpecList, len: 1
10
11  print the object returned in JSON format:
12  {
13    "_type": "package_specs",
14    "package_spec": [
15      {
16        "_type": "package_spec",
17        "available_time": "2015-08-07T13:22:40",
18        "command": "cmd /c cscript //T:60 disable-java-auto-update.vbs",
19        "command_timeout": 60,
20        "creation_time": "2001-01-01T00:00:00",
21        "deleted_flag": 0,
22        "display_name": "Disable Java Auto Update",
23        "expire_seconds": 660,
24        "files": {
25          "_type": "package_files",
26          "file": [
27            {
28              "_type": "file",
29              "bytes_downloaded": 0,
30              "bytes_total": 0,
31              "cache_status": "CACHED",
32              "download_seconds": 0,
33              "file_status": {
34                "_type": "file_status",
35                "status": [
36                  {
37                    "_type": "status",
38                    "bytes_downloaded": 0,
39                    "bytes_total": 0,
40                    "cache_status": "Processing",
41                    "server_id": 1,
42                    "server_name": "JTANIUM1.localdomain:17472",
43                    "status": 0
44                  }
45                ]
46              },
47              "hash": "9e36208ce643c767ad76ef2ad6a69141fbb5a59a607b8eb8065db09e3a153c0d",
48              "id": 43,
49              "name": "disable-java-auto-update.vbs",
50              "size": 11377,
```

```
51              "source": "https://content.tanium.com/files/published/InitialContent/2015-06-04_18-59-45
52              "status": 0
53            }
54          ]
55        },
56        "hidden_flag": 0,
57        "id": 83,
58        "last_update": "2001-01-01T00:00:00",
59        "metadata": {
60          "_type": "metadata",
61          "item": [
62            {
63              "_type": "item",
64              "admin_flag": 0,
65              "name": "defined",
66              "value": "Tanium"
67            },
68            {
69              "_type": "item",
70              "admin_flag": 0,
71              "name": "category",
72              "value": "Tanium"
73            }
74          ]
75        },
76        "modification_time": "2001-01-01T00:00:00",
77        "name": "Disable Java Auto Update API TEST",
78        "skip_lock_flag": 0,
79        "source_id": 0,
80        "verify_expire_seconds": 600,
81        "verify_group": {
82          "_type": "group",
83          "id": 0
84        },
85        "verify_group_id": 0
86      }
87    ]
88 }
```

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
import os
import sys
sys.dont_write_bytecode = True

# Determine our script name, script dir
my_file = os.path.abspath(sys.argv[0])
my_dir = os.path.dirname(my_file)

# determine the pytan lib dir and add it to the path
```

```
10  parent_dir = os.path.dirname(my_dir)
11  pytan_root_dir = os.path.dirname(parent_dir)
12  lib_dir = os.path.join(pytan_root_dir, 'lib')
13  path_adds = [lib_dir]
14
15  for aa in path_adds:
16      if aa not in sys.path:
17          sys.path.append(aa)
18
19
20  # connection info for Tanium Server
21  USERNAME = "Tanium User"
22  PASSWORD = "T@n!um"
23  HOST = "172.16.31.128"
24  PORT = "443"
25
26  # Logging conrols
27  LOGLEVEL = 2
28  DEBUGFORMAT = False
29
30  import tempfile
31
32  import pytan
33  handler = pytan.Handler(
34      username=USERNAME,
35      password=PASSWORD,
36      host=HOST,
37      port=PORT,
38      loglevel=LOGLEVEL,
39      debugformat=DEBUGFORMAT,
40  )
41
42  print handler
43
44  # set the attribute name and value we want to add to the original object (if any)
45  attr_name = "name"
46  attr_add = " API TEST"
47
48  # delete object before creating it?
49  delete = True
50
51  # setup the arguments for getting an object to export as json file
52  get_kwargs = {}
53  get_kwargs["objtype"] = u'user'
54  get_kwargs["id"] = 1
55
56
57  # get objects to use as an export to JSON file
58  orig_objs = handler.get(**get_kwargs)
59
60  # if attr_name and attr_add exists, modify the orig_objs to add attr_add to the attribute
61  # attr_name
62  if attr_name:
63      for x in orig_objs:
64          new_attr = getattr(x, attr_name)
65          new_attr += attr_add
66          setattr(x, attr_name, new_attr)
67          if delete:
```

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

Output from Python Code

```
1   Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2   2015-08-07 19:46:14,358 INFO     pytan.handler: Deleted "User, name: 'Jim Olsen API TEST', id: 12"
3   2015-08-07 19:46:14,359 INFO     pytan.handler: Report file '/var/folders/dk/vjr1r_c53yx6k6gzp2bbt_c
4   2015-08-07 19:46:14,380 INFO     pytan.handler: New User, name: 'Jim Olsen API TEST', id: 16 (ID: 16
5
6   Type of response:  <class 'taniumpy.object_types.user_list.UserList'>
7
8   print of response:
9   UserList, len: 1
10
11  print the object returned in JSON format:
12  {
13    "_type": "users",
14    "user": [
15      {
16        "_type": "user",
17        "deleted_flag": 0,
18        "group_id": 0,
19        "id": 16,
20        "last_login": "2001-01-01T00:00:00",
21        "local_admin_flag": -1,
22        "name": "Jim Olsen API TEST",
23        "permissions": {
```

```
24            "_type": "permissions",
25            "permission": [
26              "admin",
27              "sensor_read",
28              "sensor_write",
29              "question_read",
30              "question_write",
31              "action_read",
32              "action_write",
33              "action_approval",
34              "notification_write",
35              "clients_read",
36              "question_log_read",
37              "content_admin"
38            ]
39          },
40          "roles": {
41            "_type": "roles",
42            "role": [
43              {
44                "_type": "role",
45                "description": "Administrators can perform all functions in the system, including creati
46                "id": 1,
47                "name": "Administrator",
48                "permissions": {
49                  "_type": "permissions",
50                  "permission": [
51                    "admin",
52                    "sensor_read",
53                    "sensor_write",
54                    "question_read",
55                    "question_write",
56                    "action_read",
57                    "action_write",
58                    "action_approval",
59                    "notification_write",
60                    "clients_read",
61                    "question_log_read",
62                    "content_admin"
63                  ]
64                }
65              }
66            ]
67          }
68        }
69      ]
70  }
```

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

```python
1    import os
2    import sys
3    sys.dont_write_bytecode = True
4
5    # Determine our script name, script dir
6    my_file = os.path.abspath(sys.argv[0])
7    my_dir = os.path.dirname(my_file)
8
9    # determine the pytan lib dir and add it to the path
10   parent_dir = os.path.dirname(my_dir)
11   pytan_root_dir = os.path.dirname(parent_dir)
12   lib_dir = os.path.join(pytan_root_dir, 'lib')
13   path_adds = [lib_dir]
14
15   for aa in path_adds:
16       if aa not in sys.path:
17           sys.path.append(aa)
18
19
20   # connection info for Tanium Server
21   USERNAME = "Tanium User"
22   PASSWORD = "T@n!um"
23   HOST = "172.16.31.128"
24   PORT = "443"
25
26   # Logging conrols
27   LOGLEVEL = 2
28   DEBUGFORMAT = False
29
30   import tempfile
31
32   import pytan
33   handler = pytan.Handler(
34       username=USERNAME,
35       password=PASSWORD,
36       host=HOST,
37       port=PORT,
38       loglevel=LOGLEVEL,
39       debugformat=DEBUGFORMAT,
40   )
41
42   print handler
43
44   # set the attribute name and value we want to add to the original object (if any)
45   attr_name = "name"
46   attr_add = " API TEST"
47
48   # delete object before creating it?
49   delete = True
50
51   # setup the arguments for getting an object to export as json file
52   get_kwargs = {}
53   get_kwargs["objtype"] = u'saved_question'
54   get_kwargs["id"] = 1
55
56
57   # get objects to use as an export to JSON file
58   orig_objs = handler.get(**get_kwargs)
```

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

Output from Python Code

```
1   Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2   2015-08-07 19:46:14,432 INFO     pytan.handler: Deleted 'SavedQuestion, id: 109'
3   2015-08-07 19:46:14,433 INFO     pytan.handler: Report file '/var/folders/dk/vjr1r_c53yx6k6gzp2bbt_c
4   2015-08-07 19:46:14,460 INFO     pytan.handler: New SavedQuestion, name: 'Has Tanium Standard Utilit
5
6   Type of response:  <class 'taniumpy.object_types.saved_question_list.SavedQuestionList'>
7
8   print of response:
9   SavedQuestionList, len: 1
10
11  print the object returned in JSON format:
12  {
13    "_type": "saved_questions",
14    "saved_question": [
```

```
15          {
16            "_type": "saved_question",
17            "action_tracking_flag": 0,
18            "archive_enabled_flag": 0,
19            "archive_owner": {
20              "_type": "user"
21            },
22            "expire_seconds": 600,
23            "hidden_flag": 0,
24            "id": 111,
25            "issue_seconds": 120,
26            "issue_seconds_never_flag": 0,
27            "keep_seconds": 0,
28            "mod_time": "2015-08-07T19:46:14",
29            "mod_user": {
30              "_type": "user",
31              "name": "Tanium User"
32            },
33            "most_recent_question_id": 1256,
34            "name": "Has Tanium Standard Utilities API TEST",
35            "packages": {
36              "_type": "package_specs",
37              "package_spec": [
38                {
39                  "_type": "package_spec",
40                  "id": 20,
41                  "name": "Distribute Tanium Standard Utilities"
42                }
43              ]
44            },
45            "public_flag": 1,
46            "query_text": "Get Has Tanium Standard Utilities from all machines",
47            "question": {
48              "_type": "question",
49              "action_tracking_flag": 0,
50              "expiration": "2015-08-07T19:32:37",
51              "expire_seconds": 0,
52              "force_computer_id_flag": 0,
53              "hidden_flag": 0,
54              "id": 1256,
55              "management_rights_group": {
56                "_type": "group",
57                "id": 0
58              },
59              "query_text": "Get Has Tanium Standard Utilities from all machines",
60              "saved_question": {
61                "_type": "saved_question",
62                "id": 110
63              },
64              "selects": {
65                "_type": "selects",
66                "select": [
67                  {
68                    "_type": "select",
69                    "filter": {
70                      "_type": "filter",
71                      "all_times_flag": 0,
72                      "all_values_flag": 0,
```

```
73                      "delimiter_index": 0,
74                      "end_time": "2001-01-01T00:00:00",
75                      "ignore_case_flag": 1,
76                      "max_age_seconds": 0,
77                      "not_flag": 0,
78                      "operator": "Less",
79                      "start_time": "2001-01-01T00:00:00",
80                      "substring_flag": 0,
81                      "substring_length": 0,
82                      "substring_start": 0,
83                      "utf8_flag": 0,
84                      "value_type": "String"
85                  },
86                  "sensor": {
87                      "_type": "sensor",
88                      "category": "Tanium",
89                      "creation_time": "2015-08-07T13:22:09",
90                      "delimiter": ",",
91                      "description": "Returns whether a machine has the Tanium Standard Utilities\nExample
92                      "exclude_from_parse_flag": 1,
93                      "hash": 1782389954,
94                      "hidden_flag": 0,
95                      "id": 194,
96                      "ignore_case_flag": 1,
97                      "last_modified_by": "Jim Olsen",
98                      "max_age_seconds": 900,
99                      "modification_time": "2015-08-07T13:22:09",
100                     "name": "Has Tanium Standard Utilities",
101                     "queries": {
102                         "_type": "queries",
103                         "query": [
104                             {
105                                 "_type": "query",
106                                 "platform": "Windows",
107                                 "script": "&#039;=======================================\n&#039; Has Tanium S
108                                 "script_type": "VBScript"
109                             },
110                             {
111                                 "_type": "query",
112                                 "platform": "Linux",
113                                 "script": "#!/bin/sh\n\n# THIS IS A STUB - NOT INTENDED AS FUNCTIONAL - NA\n#
114                                 "script_type": "UnixShell"
115                             },
116                             {
117                                 "_type": "query",
118                                 "platform": "Mac",
119                                 "script": "#!/bin/sh\n\n# THIS IS A STUB - NOT INTENDED AS FUNCTIONAL - NA\n#
120                                 "script_type": "UnixShell"
121                             },
122                             {
123                                 "_type": "query",
124                                 "platform": "Solaris",
125                                 "script": "#!/bin/sh\n\n# THIS IS A STUB - NOT INTENDED AS FUNCTIONAL - NA\n#
126                                 "script_type": "UnixShell"
127                             },
128                             {
129                                 "_type": "query",
130                                 "platform": "AIX",
```

```
131                       "script": "#!/bin/sh\n\n# THIS IS A STUB - NOT INTENDED AS FUNCTIONAL - NA\n#
132                       "script_type": "UnixShell"
133                     }
134                   ]
135                 },
136                 "source_id": 0,
137                 "string_count": 16,
138                 "value_type": "String"
139               }
140             }
141           ]
142         },
143         "skip_lock_flag": 0,
144         "user": {
145           "_type": "user",
146           "id": 1,
147           "name": "Jim Olsen"
148         }
149       },
150       "row_count_flag": 0,
151       "sort_column": 0,
152       "user": {
153         "_type": "user",
154         "id": 2,
155         "name": "Tanium User"
156       }
157     }
158   ]
159 }
```

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

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

```
19
20  # connection info for Tanium Server
21  USERNAME = "Tanium User"
22  PASSWORD = "T@n!um"
23  HOST = "172.16.31.128"
24  PORT = "443"
25
26  # Logging conrols
27  LOGLEVEL = 2
28  DEBUGFORMAT = False
29
30  import tempfile
31
32  import pytan
33  handler = pytan.Handler(
34      username=USERNAME,
35      password=PASSWORD,
36      host=HOST,
37      port=PORT,
38      loglevel=LOGLEVEL,
39      debugformat=DEBUGFORMAT,
40  )
41
42  print handler
43
44  # set the attribute name and value we want to add to the original object (if any)
45  attr_name = ""
46  attr_add = ""
47
48  # delete object before creating it?
49  delete = False
50
51  # setup the arguments for getting an object to export as json file
52  get_kwargs = {}
53  get_kwargs["objtype"] = u'action'
54  get_kwargs["id"] = 1
55
56
57  # get objects to use as an export to JSON file
58  orig_objs = handler.get(**get_kwargs)
59
60  # if attr_name and attr_add exists, modify the orig_objs to add attr_add to the attribute
61  # attr_name
62  if attr_name:
63      for x in orig_objs:
64          new_attr = getattr(x, attr_name)
65          new_attr += attr_add
66          setattr(x, attr_name, new_attr)
67          if delete:
68              # delete the object in case it already exists
69              del_kwargs = {}
70              del_kwargs[attr_name] = new_attr
71              del_kwargs['objtype'] = u'action'
72              try:
73                  handler.delete(**del_kwargs)
74              except Exception as e:
75                  print e
76
```

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

Output from Python Code

```
1    Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2    2015-08-07 19:46:14,474 INFO     pytan.handler: Report file '/var/folders/dk/vjr1r_c53yx6k6gzp2bbt_c
3    2015-08-07 19:46:14,496 INFO     pytan.handler: New Action, name: 'Distribute Tanium Standard Utilit
4
5    Type of response:  <class 'taniumpy.object_types.action_list.ActionList'>
6
7    print of response:
8    ActionList, len: 1
9
10   print the object returned in JSON format:
11   {
12     "_type": "actions",
13     "action": [
14       {
15         "_type": "action",
16         "action_group": {
17           "_type": "group",
18           "id": 0,
19           "name": "Default"
20         },
21         "approver": {
22           "_type": "user",
23           "id": 2,
24           "name": "Tanium User"
25         },
26         "comment": "Distribute Tanium Standard Utilities",
27         "creation_time": "2015-08-07T19:46:14",
28         "distribute_seconds": 3200,
29         "expiration_time": "2015-08-07T20:41:16",
30         "expire_seconds": 3300,
31         "history_saved_question": {
32           "_type": "saved_question",
```

```
33          "id": 102
34        },
35        "id": 60,
36        "name": "Distribute Tanium Standard Utilities",
37        "package_spec": {
38          "_type": "package_spec",
39          "command": "cmd /c cscript install-standard-utils.vbs \"Tools\\StdUtils\"",
40          "id": 20,
41          "name": "Distribute Tanium Standard Utilities"
42        },
43        "saved_action": {
44          "_type": "saved_action",
45          "id": 46
46        },
47        "skip_lock_flag": 0,
48        "start_time": "2015-08-07T19:46:16",
49        "status": "Open",
50        "stopped_flag": 0,
51        "target_group": {
52          "_type": "group",
53          "id": 37,
54          "name": "Default"
55        },
56        "user": {
57          "_type": "user",
58          "group_id": 0,
59          "id": 2,
60          "last_login": "2015-08-07T19:46:14",
61          "name": "Tanium User"
62        }
63      }
64    ]
65  }
```

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

```
14
15  for aa in path_adds:
16      if aa not in sys.path:
17          sys.path.append(aa)
18
19
20  # connection info for Tanium Server
21  USERNAME = "Tanium User"
22  PASSWORD = "T@n!um"
23  HOST = "172.16.31.128"
24  PORT = "443"
25
26  # Logging conrols
27  LOGLEVEL = 2
28  DEBUGFORMAT = False
29
30  import tempfile
31
32  import pytan
33  handler = pytan.Handler(
34      username=USERNAME,
35      password=PASSWORD,
36      host=HOST,
37      port=PORT,
38      loglevel=LOGLEVEL,
39      debugformat=DEBUGFORMAT,
40  )
41
42  print handler
43
44  # set the attribute name and value we want to add to the original object (if any)
45  attr_name = "name"
46  attr_add = " API TEST"
47
48  # delete object before creating it?
49  delete = True
50
51  # setup the arguments for getting an object to export as json file
52  get_kwargs = {}
53  get_kwargs["objtype"] = u'sensor'
54  get_kwargs["id"] = 381
55
56
57  # get objects to use as an export to JSON file
58  orig_objs = handler.get(**get_kwargs)
59
60  # if attr_name and attr_add exists, modify the orig_objs to add attr_add to the attribute
61  # attr_name
62  if attr_name:
63      for x in orig_objs:
64          new_attr = getattr(x, attr_name)
65          new_attr += attr_add
66          setattr(x, attr_name, new_attr)
67          if delete:
68              # delete the object in case it already exists
69              del_kwargs = {}
70              del_kwargs[attr_name] = new_attr
71              del_kwargs['objtype'] = u'sensor'
```

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

Output from Python Code

```
1    Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2    2015-08-07 19:46:14,551 INFO     pytan.handler: Deleted 'Sensor, id: 639'
3    2015-08-07 19:46:14,551 INFO     pytan.handler: Report file '/var/folders/dk/vjr1r_c53yx6k6gzp2bbt_c
4    2015-08-07 19:46:14,577 INFO     pytan.handler: New Sensor, name: 'Is Mac API TEST', id: 642 (ID: 64
5
6    Type of response:  <class 'taniumpy.object_types.sensor_list.SensorList'>
7
8    print of response:
9    SensorList, len: 1
10
11   print the object returned in JSON format:
12   {
13     "_type": "sensors",
14     "sensor": [
15       {
16         "_type": "sensor",
17         "category": "Operating System",
18         "creation_time": "2015-08-07T19:46:14",
19         "delimiter": ",",
20         "description": "Returns whether the machine is a Mac.  True if so, False if not.\nExample: Tru
21         "exclude_from_parse_flag": 0,
22         "hash": 2387245230,
23         "hidden_flag": 0,
24         "id": 642,
25         "ignore_case_flag": 1,
26         "last_modified_by": "Tanium User",
27         "max_age_seconds": 86400,
```

```
28          "modification_time": "2015-08-07T19:46:14",
29          "name": "Is Mac API TEST",
30          "queries": {
31            "_type": "queries",
32            "query": [
33              {
34                "_type": "query",
35                "platform": "Windows",
36                "script": "&#039;=======================================\n&#039; Is Mac\n&#039;========
37                "script_type": "VBScript"
38              },
39              {
40                "_type": "query",
41                "platform": "Linux",
42                "script": "#!/bin/bash\necho False\n",
43                "script_type": "UnixShell"
44              },
45              {
46                "_type": "query",
47                "platform": "Mac",
48                "script": "#!/bin/bash\necho True\n",
49                "script_type": "UnixShell"
50              },
51              {
52                "_type": "query",
53                "platform": "Solaris",
54                "script": "#!/bin/sh\n\n# THIS IS A STUB - NOT INTENDED AS FUNCTIONAL - NA\n# \n# \n\nec
55                "script_type": "UnixShell"
56              },
57              {
58                "_type": "query",
59                "platform": "AIX",
60                "script": "#!/bin/sh\n\n# THIS IS A STUB - NOT INTENDED AS FUNCTIONAL - NA\n# \n# \n\nec
61                "script_type": "UnixShell"
62              }
63            ]
64          },
65          "source_id": 0,
66          "string_count": 0,
67          "value_type": "String"
68        }
69      ]
70    }
```

Create question from json

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

Example Python Code

```python
import os
import sys
sys.dont_write_bytecode = True

```

```
5    # Determine our script name, script dir
6    my_file = os.path.abspath(sys.argv[0])
7    my_dir = os.path.dirname(my_file)
8
9    # determine the pytan lib dir and add it to the path
10   parent_dir = os.path.dirname(my_dir)
11   pytan_root_dir = os.path.dirname(parent_dir)
12   lib_dir = os.path.join(pytan_root_dir, 'lib')
13   path_adds = [lib_dir]
14
15   for aa in path_adds:
16       if aa not in sys.path:
17           sys.path.append(aa)
18
19
20   # connection info for Tanium Server
21   USERNAME = "Tanium User"
22   PASSWORD = "T@n!um"
23   HOST = "172.16.31.128"
24   PORT = "443"
25
26   # Logging conrols
27   LOGLEVEL = 2
28   DEBUGFORMAT = False
29
30   import tempfile
31
32   import pytan
33   handler = pytan.Handler(
34       username=USERNAME,
35       password=PASSWORD,
36       host=HOST,
37       port=PORT,
38       loglevel=LOGLEVEL,
39       debugformat=DEBUGFORMAT,
40   )
41
42   print handler
43
44   # set the attribute name and value we want to add to the original object (if any)
45   attr_name = ""
46   attr_add = ""
47
48   # delete object before creating it?
49   delete = False
50
51   # setup the arguments for getting an object to export as json file
52   get_kwargs = {}
53   get_kwargs["objtype"] = u'question'
54   get_kwargs["id"] = 1
55
56
57   # get objects to use as an export to JSON file
58   orig_objs = handler.get(**get_kwargs)
59
60   # if attr_name and attr_add exists, modify the orig_objs to add attr_add to the attribute
61   # attr_name
62   if attr_name:
```

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

Output from Python Code

```
1    Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2    2015-08-07 19:46:14,608 INFO     pytan.handler: Report file '/var/folders/dk/vjr1r_c53yx6k6gzp2bbt_c
3    2015-08-07 19:46:14,640 INFO     pytan.handler: New Question, id: 1305 (ID: 1305) created successful
4
5    Type of response:  <class 'taniumpy.object_types.question_list.QuestionList'>
6
7    print of response:
8    QuestionList, len: 1
9
10   print the object returned in JSON format:
11   {
12     "_type": "questions",
13     "question": [
14       {
15         "_type": "question",
16         "action_tracking_flag": 0,
17         "context_group": {
18           "_type": "group",
```

```
19              "id": 0
20            },
21            "expiration": "2015-08-07T19:56:14",
22            "expire_seconds": 0,
23            "force_computer_id_flag": 1,
24            "hidden_flag": 0,
25            "id": 1305,
26            "management_rights_group": {
27              "_type": "group",
28              "id": 0
29            },
30            "query_text": "Get Action Statuses matching \"Nil\" from all machines",
31            "saved_question": {
32              "_type": "saved_question",
33              "id": 0
34            },
35            "selects": {
36              "_type": "selects",
37              "select": [
38                {
39                  "_type": "select",
40                  "filter": {
41                    "_type": "filter",
42                    "all_times_flag": 0,
43                    "all_values_flag": 1,
44                    "delimiter_index": 0,
45                    "end_time": "2001-01-01T00:00:00",
46                    "ignore_case_flag": 1,
47                    "max_age_seconds": 0,
48                    "not_flag": 0,
49                    "operator": "RegexMatch",
50                    "start_time": "2001-01-01T00:00:00",
51                    "substring_flag": 0,
52                    "substring_length": 0,
53                    "substring_start": 0,
54                    "utf8_flag": 0,
55                    "value": "Nil",
56                    "value_type": "String"
57                  },
58                  "sensor": {
59                    "_type": "sensor",
60                    "category": "Reserved",
61                    "description": "The recorded state of each action a client has taken recently in the f
62                    "exclude_from_parse_flag": 1,
63                    "hash": 1792443391,
64                    "hidden_flag": 0,
65                    "id": 1,
66                    "ignore_case_flag": 1,
67                    "max_age_seconds": 3600,
68                    "name": "Action Statuses",
69                    "queries": {
70                      "_type": "queries",
71                      "query": [
72                        {
73                          "_type": "query",
74                          "platform": "Windows",
75                          "script": "Reserved",
76                          "script_type": "WMIQuery"
```

```
77                        }
78                      ]
79                    },
80                    "source_id": 0,
81                    "string_count": 238,
82                    "value_type": "String"
83                  }
84                }
85              ]
86            },
87            "skip_lock_flag": 0,
88            "user": {
89              "_type": "user",
90              "id": 2,
91              "name": "Tanium User"
92            }
93          }
94        ]
95    }
```

Create whitelisted url from json

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

Example Python Code

```python
import os
import sys
sys.dont_write_bytecode = True

# Determine our script name, script dir
my_file = os.path.abspath(sys.argv[0])
my_dir = os.path.dirname(my_file)

# determine the pytan lib dir and add it to the path
parent_dir = os.path.dirname(my_dir)
pytan_root_dir = os.path.dirname(parent_dir)
lib_dir = os.path.join(pytan_root_dir, 'lib')
path_adds = [lib_dir]

for aa in path_adds:
    if aa not in sys.path:
        sys.path.append(aa)


# connection info for Tanium Server
USERNAME = "Tanium User"
PASSWORD = "T@n!um"
HOST = "172.16.31.128"
PORT = "443"

# Logging conrols
LOGLEVEL = 2
```

```
28  DEBUGFORMAT = False
29
30  import tempfile
31
32  import pytan
33  handler = pytan.Handler(
34      username=USERNAME,
35      password=PASSWORD,
36      host=HOST,
37      port=PORT,
38      loglevel=LOGLEVEL,
39      debugformat=DEBUGFORMAT,
40  )
41
42  print handler
43
44  # set the attribute name and value we want to add to the original object (if any)
45  attr_name = "url_regex"
46  attr_add = " API TEST"
47
48  # delete object before creating it?
49  delete = True
50
51  # setup the arguments for getting an object to export as json file
52  get_kwargs = {}
53  get_kwargs["objtype"] = u'whitelisted_url'
54  get_kwargs["url_regex"] = u'test1'
55
56
57  # get objects to use as an export to JSON file
58  orig_objs = handler.get(**get_kwargs)
59
60  # if attr_name and attr_add exists, modify the orig_objs to add attr_add to the attribute
61  # attr_name
62  if attr_name:
63      for x in orig_objs:
64          new_attr = getattr(x, attr_name)
65          new_attr += attr_add
66          setattr(x, attr_name, new_attr)
67          if delete:
68              # delete the object in case it already exists
69              del_kwargs = {}
70              del_kwargs[attr_name] = new_attr
71              del_kwargs['objtype'] = u'whitelisted_url'
72              try:
73                  handler.delete(**del_kwargs)
74              except Exception as e:
75                  print e
76
77  # export orig_objs to a json file
78  json_file, results = handler.export_to_report_file(
79      obj=orig_objs,
80      export_format='json',
81      report_dir=tempfile.gettempdir(),
82  )
83
84  # create the object from the exported JSON file
85  create_kwargs = {'objtype': u'whitelisted_url', 'json_file': json_file}
```

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

Output from Python Code

```
1    Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2    2015-08-07 19:46:14,685 INFO     pytan.handler: Deleted 'WhiteListedUrl, id: 27'
3    2015-08-07 19:46:14,686 INFO     pytan.handler: Report file '/var/folders/dk/vjr1r_c53yx6k6gzp2bbt_c
4    2015-08-07 19:46:14,695 INFO     pytan.handler: New WhiteListedUrl, id: 53 (ID: 53) created successf
5
6    Type of response:  <class 'taniumpy.object_types.white_listed_url_list.WhiteListedUrlList'>
7
8    print of response:
9    WhiteListedUrlList, len: 1
10
11   print the object returned in JSON format:
12   {
13     "_type": "white_listed_urls",
14     "white_listed_url": [
15       {
16         "_type": "white_listed_url",
17         "download_seconds": 86400,
18         "id": 53,
19         "url_regex": "test1 API TEST"
20       }
21     ]
22   }
```

Create group from json

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

Example Python Code

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

```
8
9    # determine the pytan lib dir and add it to the path
10   parent_dir = os.path.dirname(my_dir)
11   pytan_root_dir = os.path.dirname(parent_dir)
12   lib_dir = os.path.join(pytan_root_dir, 'lib')
13   path_adds = [lib_dir]
14
15   for aa in path_adds:
16       if aa not in sys.path:
17           sys.path.append(aa)
18
19
20   # connection info for Tanium Server
21   USERNAME = "Tanium User"
22   PASSWORD = "T@n!um"
23   HOST = "172.16.31.128"
24   PORT = "443"
25
26   # Logging conrols
27   LOGLEVEL = 2
28   DEBUGFORMAT = False
29
30   import tempfile
31
32   import pytan
33   handler = pytan.Handler(
34       username=USERNAME,
35       password=PASSWORD,
36       host=HOST,
37       port=PORT,
38       loglevel=LOGLEVEL,
39       debugformat=DEBUGFORMAT,
40   )
41
42   print handler
43
44   # set the attribute name and value we want to add to the original object (if any)
45   attr_name = "name"
46   attr_add = " API TEST"
47
48   # delete object before creating it?
49   delete = True
50
51   # setup the arguments for getting an object to export as json file
52   get_kwargs = {}
53   get_kwargs["objtype"] = u'group'
54   get_kwargs["name"] = u'All Computers'
55
56
57   # get objects to use as an export to JSON file
58   orig_objs = handler.get(**get_kwargs)
59
60   # if attr_name and attr_add exists, modify the orig_objs to add attr_add to the attribute
61   # attr_name
62   if attr_name:
63       for x in orig_objs:
64           new_attr = getattr(x, attr_name)
65           new_attr += attr_add
```

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

Output from Python Code

```
1   Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2   2015-08-07 19:46:14,746 INFO     pytan.handler: Deleted 'Group, id: 157'
3   2015-08-07 19:46:14,747 INFO     pytan.handler: Report file '/var/folders/dk/vjr1r_c53yx6k6gzp2bbt_c
4   2015-08-07 19:46:14,769 INFO     pytan.handler: New Group, name: 'All Computers API TEST', id: 213 (
5
6   Type of response:  <class 'taniumpy.object_types.group_list.GroupList'>
7
8   print of response:
9   GroupList, len: 1
10
11  print the object returned in JSON format:
12  {
13    "_type": "groups",
14    "group": [
15      {
16        "_type": "group",
17        "and_flag": 0,
18        "deleted_flag": 0,
19        "filters": {
20          "_type": "filters",
21          "filter": []
```

```
22          },
23          "id": 213,
24          "name": "All Computers API TEST",
25          "not_flag": 0,
26          "sub_groups": {
27            "_type": "groups",
28            "group": []
29          },
30          "type": 0
31        }
32      ]
33    }
```

PyTan API Invalid Create Object From JSON Examples

Invalid create saved action from json

Create a saved action from json (not supported!)

Example Python Code

```python
1   import os
2   import sys
3   sys.dont_write_bytecode = True
4
5   # Determine our script name, script dir
6   my_file = os.path.abspath(sys.argv[0])
7   my_dir = os.path.dirname(my_file)
8
9   # determine the pytan lib dir and add it to the path
10  parent_dir = os.path.dirname(my_dir)
11  pytan_root_dir = os.path.dirname(parent_dir)
12  lib_dir = os.path.join(pytan_root_dir, 'lib')
13  path_adds = [lib_dir]
14
15  for aa in path_adds:
16      if aa not in sys.path:
17          sys.path.append(aa)
18
19
20  # connection info for Tanium Server
21  USERNAME = "Tanium User"
22  PASSWORD = "T@n!um"
23  HOST = "172.16.31.128"
24  PORT = "443"
25
26  # Logging conrols
27  LOGLEVEL = 2
28  DEBUGFORMAT = False
29
30  import tempfile
31
32  import pytan
33  handler = pytan.Handler(
34      username=USERNAME,
```

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

Output from Python Code

```
1    Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2    2015-08-07 19:46:14,794 INFO     pytan.handler: Report file '/var/folders/dk/vjr1r_c53yx6k6gzp2bbt_c
3    Traceback (most recent call last):
4      File "<string>", line 67, in <module>
5      File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 546, in create_from_json
6        raise pytan.exceptions.HandlerError(m(objtype, json_createable))
7    HandlerError: saved_action is not a json createable object! Supported objects: user, whitelisted_url
```

Invalid create client from json

Create a client from json (not supported!)

Example Python Code

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

```
4
5   # Determine our script name, script dir
6   my_file = os.path.abspath(sys.argv[0])
7   my_dir = os.path.dirname(my_file)
8
9   # determine the pytan lib dir and add it to the path
10  parent_dir = os.path.dirname(my_dir)
11  pytan_root_dir = os.path.dirname(parent_dir)
12  lib_dir = os.path.join(pytan_root_dir, 'lib')
13  path_adds = [lib_dir]
14
15  for aa in path_adds:
16      if aa not in sys.path:
17          sys.path.append(aa)
18
19
20  # connection info for Tanium Server
21  USERNAME = "Tanium User"
22  PASSWORD = "T@n!um"
23  HOST = "172.16.31.128"
24  PORT = "443"
25
26  # Logging conrols
27  LOGLEVEL = 2
28  DEBUGFORMAT = False
29
30  import tempfile
31
32  import pytan
33  handler = pytan.Handler(
34      username=USERNAME,
35      password=PASSWORD,
36      host=HOST,
37      port=PORT,
38      loglevel=LOGLEVEL,
39      debugformat=DEBUGFORMAT,
40  )
41
42  print handler
43
44  # setup the arguments for getting an object to export as json file
45  get_kwargs = {}
46  get_kwargs["objtype"] = u'client'
47  get_kwargs["status"] = u'Leader'
48
49  # get objects to use as an export to JSON file
50  orig_objs = handler.get(**get_kwargs)
51
52  # export orig_objs to a json file
53  json_file, results = handler.export_to_report_file(
54      obj=orig_objs,
55      export_format='json',
56      report_dir=tempfile.gettempdir(),
57  )
58
59  # call the handler with the create_from_json method, passing in kwargs for arguments
60  # this should throw an exception: pytan.exceptions.HandlerError
61  import traceback
```

```
62
63   # create the object from the exported JSON file
64   create_kwargs = {'objtype': u'client', 'json_file': json_file}
65   try:
66       response = handler.create_from_json(**create_kwargs)
67   except Exception as e:
68       traceback.print_exc(file=sys.stdout)
```

Output from Python Code

```
1   Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2   2015-08-07 19:46:14,805 INFO     pytan.handler: Report file '/var/folders/dk/vjr1r_c53yx6k6gzp2bbt_c
3   Traceback (most recent call last):
4     File "<string>", line 67, in <module>
5     File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 546, in create_from_json
6       raise pytan.exceptions.HandlerError(m(objtype, json_createable))
7   HandlerError: client is not a json createable object! Supported objects: user, whitelisted_url, save
```

Invalid create userrole from json

Create a user role from json (not supported!)

Example Python Code

```python
1    import os
2    import sys
3    sys.dont_write_bytecode = True
4
5    # Determine our script name, script dir
6    my_file = os.path.abspath(sys.argv[0])
7    my_dir = os.path.dirname(my_file)
8
9    # determine the pytan lib dir and add it to the path
10   parent_dir = os.path.dirname(my_dir)
11   pytan_root_dir = os.path.dirname(parent_dir)
12   lib_dir = os.path.join(pytan_root_dir, 'lib')
13   path_adds = [lib_dir]
14
15   for aa in path_adds:
16       if aa not in sys.path:
17           sys.path.append(aa)
18
19
20   # connection info for Tanium Server
21   USERNAME = "Tanium User"
22   PASSWORD = "T@n!um"
23   HOST = "172.16.31.128"
24   PORT = "443"
25
26   # Logging conrols
27   LOGLEVEL = 2
28   DEBUGFORMAT = False
29
30   import tempfile
```

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

Output from Python Code

```
1   Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2   2015-08-07 19:46:14,815 INFO     pytan.handler: Report file '/var/folders/dk/vjr1r_c53yx6k6gzp2bbt_c
3   Traceback (most recent call last):
4     File "<string>", line 67, in <module>
5     File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 546, in create_from_json
6       raise pytan.exceptions.HandlerError(m(objtype, json_createable))
7   HandlerError: userrole is not a json createable object! Supported objects: user, whitelisted_url, sa
```

Invalid create setting from json

Create a setting from json (not supported!)

Example Python Code

```
1   import os
2   import sys
3   sys.dont_write_bytecode = True
4
5   # Determine our script name, script dir
6   my_file = os.path.abspath(sys.argv[0])
7   my_dir = os.path.dirname(my_file)
8
9   # determine the pytan lib dir and add it to the path
10  parent_dir = os.path.dirname(my_dir)
11  pytan_root_dir = os.path.dirname(parent_dir)
12  lib_dir = os.path.join(pytan_root_dir, 'lib')
13  path_adds = [lib_dir]
14
15  for aa in path_adds:
16      if aa not in sys.path:
17          sys.path.append(aa)
18
19
20  # connection info for Tanium Server
21  USERNAME = "Tanium User"
22  PASSWORD = "T@n!um"
23  HOST = "172.16.31.128"
24  PORT = "443"
25
26  # Logging conrols
27  LOGLEVEL = 2
28  DEBUGFORMAT = False
29
30  import tempfile
31
32  import pytan
33  handler = pytan.Handler(
34      username=USERNAME,
35      password=PASSWORD,
36      host=HOST,
37      port=PORT,
38      loglevel=LOGLEVEL,
39      debugformat=DEBUGFORMAT,
40  )
41
42  print handler
43
44  # setup the arguments for getting an object to export as json file
45  get_kwargs = {}
46  get_kwargs["objtype"] = u'setting'
47  get_kwargs["id"] = 1
48
49  # get objects to use as an export to JSON file
50  orig_objs = handler.get(**get_kwargs)
51
52  # export orig_objs to a json file
53  json_file, results = handler.export_to_report_file(
54      obj=orig_objs,
55      export_format='json',
56      report_dir=tempfile.gettempdir(),
57  )
58
```

```
59    # call the handler with the create_from_json method, passing in kwargs for arguments
60    # this should throw an exception: pytan.exceptions.HandlerError
61    import traceback
62
63    # create the object from the exported JSON file
64    create_kwargs = {'objtype': u'setting', 'json_file': json_file}
65    try:
66        response = handler.create_from_json(**create_kwargs)
67    except Exception as e:
68        traceback.print_exc(file=sys.stdout)
```

Output from Python Code

```
1    Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2    2015-08-07 19:46:14,843 INFO     pytan.handler: Report file '/var/folders/dk/vjr1r_c53yx6k6gzp2bbt_c
3    Traceback (most recent call last):
4      File "<string>", line 67, in <module>
5      File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 546, in create_from_json
6        raise pytan.exceptions.HandlerError(m(objtype, json_createable))
7    HandlerError: setting is not a json createable object! Supported objects: user, whitelisted_url, sav
```

PyTan API Valid Export ResultSet Examples

Export resultset csv default options

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

Example Python Code

```
1    import os
2    import sys
3    sys.dont_write_bytecode = True
4
5    # Determine our script name, script dir
6    my_file = os.path.abspath(sys.argv[0])
7    my_dir = os.path.dirname(my_file)
8
9    # determine the pytan lib dir and add it to the path
10   parent_dir = os.path.dirname(my_dir)
11   pytan_root_dir = os.path.dirname(parent_dir)
12   lib_dir = os.path.join(pytan_root_dir, 'lib')
13   path_adds = [lib_dir]
14
15   for aa in path_adds:
16       if aa not in sys.path:
17           sys.path.append(aa)
18
19
20   # connection info for Tanium Server
21   USERNAME = "Tanium User"
22   PASSWORD = "T@n!um"
23   HOST = "172.16.31.128"
24   PORT = "443"
25
```

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

Output from Python Code

```
1   Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2   2015-08-07 19:46:14,930 DEBUG    pytan.handler.QuestionPoller: ID 1306: id resolved to 1306
3   2015-08-07 19:46:14,930 DEBUG    pytan.handler.QuestionPoller: ID 1306: expiration resolved to 2015-
4   2015-08-07 19:46:14,930 DEBUG    pytan.handler.QuestionPoller: ID 1306: query_text resolved to Get C
5   2015-08-07 19:46:14,930 DEBUG    pytan.handler.QuestionPoller: ID 1306: id resolved to 1306
6   2015-08-07 19:46:14,930 DEBUG    pytan.handler.QuestionPoller: ID 1306: Object Info resolved to Ques
7   2015-08-07 19:46:14,933 DEBUG    pytan.handler.QuestionPoller: ID 1306: Progress: Tested: 0, Passed:
8   2015-08-07 19:46:14,933 DEBUG    pytan.handler.QuestionPoller: ID 1306: Timing: Started: 2015-08-07
```

```
9    2015-08-07 19:46:14,933 INFO     pytan.handler.QuestionPoller: ID 1306: Progress Changed 0% (0 of 2)
10   2015-08-07 19:46:19,940 DEBUG    pytan.handler.QuestionPoller: ID 1306: Progress: Tested: 0, Passed:
11   2015-08-07 19:46:19,940 DEBUG    pytan.handler.QuestionPoller: ID 1306: Timing: Started: 2015-08-07
12   2015-08-07 19:46:24,944 DEBUG    pytan.handler.QuestionPoller: ID 1306: Progress: Tested: 0, Passed:
13   2015-08-07 19:46:24,944 DEBUG    pytan.handler.QuestionPoller: ID 1306: Timing: Started: 2015-08-07
14   2015-08-07 19:46:29,951 DEBUG    pytan.handler.QuestionPoller: ID 1306: Progress: Tested: 0, Passed:
15   2015-08-07 19:46:29,951 DEBUG    pytan.handler.QuestionPoller: ID 1306: Timing: Started: 2015-08-07
16   2015-08-07 19:46:34,955 DEBUG    pytan.handler.QuestionPoller: ID 1306: Progress: Tested: 0, Passed:
17   2015-08-07 19:46:34,955 DEBUG    pytan.handler.QuestionPoller: ID 1306: Timing: Started: 2015-08-07
18   2015-08-07 19:46:39,959 DEBUG    pytan.handler.QuestionPoller: ID 1306: Progress: Tested: 1, Passed:
19   2015-08-07 19:46:39,959 DEBUG    pytan.handler.QuestionPoller: ID 1306: Timing: Started: 2015-08-07
20   2015-08-07 19:46:39,959 INFO     pytan.handler.QuestionPoller: ID 1306: Progress Changed 50% (1 of 2
21   2015-08-07 19:46:44,964 DEBUG    pytan.handler.QuestionPoller: ID 1306: Progress: Tested: 1, Passed:
22   2015-08-07 19:46:44,965 DEBUG    pytan.handler.QuestionPoller: ID 1306: Timing: Started: 2015-08-07
23   2015-08-07 19:46:49,971 DEBUG    pytan.handler.QuestionPoller: ID 1306: Progress: Tested: 2, Passed:
24   2015-08-07 19:46:49,971 DEBUG    pytan.handler.QuestionPoller: ID 1306: Timing: Started: 2015-08-07
25   2015-08-07 19:46:49,971 INFO     pytan.handler.QuestionPoller: ID 1306: Progress Changed 100% (2 of
26   2015-08-07 19:46:49,971 INFO     pytan.handler.QuestionPoller: ID 1306: Reached Threshold of 99% (2

28   print the export_str returned from export_obj():
29   Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
30   2015-08-07 19:46:14,843 INFO     pytan.handler: Report file '/var/folders/dk/vjr1r_c53yx6k6gzp2bbt_c
31   Traceback (most recent call last):
32     File "<string>", line 67, in <module>
33     File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 546, in create_from_json
34       raise pytan.exceptions.HandlerError(m(objtype, json_createable))
35   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

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

```
22   PASSWORD = "T@n!um"
23   HOST = "172.16.31.128"
24   PORT = "443"
25
26   # Logging conrols
27   LOGLEVEL = 2
28   DEBUGFORMAT = False
29
30   import tempfile
31
32   import pytan
33   handler = pytan.Handler(
34       username=USERNAME,
35       password=PASSWORD,
36       host=HOST,
37       port=PORT,
38       loglevel=LOGLEVEL,
39       debugformat=DEBUGFORMAT,
40   )
41
42   print handler
43
44   # setup the export_obj kwargs for later
45   export_kwargs = {}
46   export_kwargs["export_format"] = u'csv'
47   export_kwargs["expand_grouped_columns"] = False
48
49   # ask the question that will provide the resultset that we want to use
50   ask_kwargs = {
51       'qtype': 'manual',
52       'sensors': [
53           "Computer Name", "IP Route Details", "IP Address",
54           'Folder Name Search with RegEx Match{dirname=Program Files,regex=.*Shared.*}',
55       ],
56   }
57   response = handler.ask(**ask_kwargs)
58
59   # export the object to a string
60   # (we could just as easily export to a file using export_to_report_file)
61   export_kwargs['obj'] = response['question_results']
62   export_str = handler.export_obj(**export_kwargs)
63
64
65   print ""
66   print "print the export_str returned from export_obj():"
67   if len(out.splitlines()) > 15:
68       out = out.splitlines()[0:15]
69       out.append('..trimmed for brevity..')
70       out = '\n'.join(out)
71
72   print out
```

Output from Python Code

```
1   Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2   2015-08-07 19:46:50,080 DEBUG    pytan.handler.QuestionPoller: ID 1307: id resolved to 1307
3   2015-08-07 19:46:50,080 DEBUG    pytan.handler.QuestionPoller: ID 1307: expiration resolved to 2015-
```

```
4   2015-08-07 19:46:50,080 DEBUG     pytan.handler.QuestionPoller: ID 1307: query_text resolved to Get C
5   2015-08-07 19:46:50,080 DEBUG     pytan.handler.QuestionPoller: ID 1307: id resolved to 1307
6   2015-08-07 19:46:50,080 DEBUG     pytan.handler.QuestionPoller: ID 1307: Object Info resolved to Ques
7   2015-08-07 19:46:50,083 DEBUG     pytan.handler.QuestionPoller: ID 1307: Progress: Tested: 0, Passed:
8   2015-08-07 19:46:50,083 DEBUG     pytan.handler.QuestionPoller: ID 1307: Timing: Started: 2015-08-07
9   2015-08-07 19:46:50,083 INFO      pytan.handler.QuestionPoller: ID 1307: Progress Changed 0% (0 of 2)
10  2015-08-07 19:46:55,088 DEBUG     pytan.handler.QuestionPoller: ID 1307: Progress: Tested: 1, Passed:
11  2015-08-07 19:46:55,088 DEBUG     pytan.handler.QuestionPoller: ID 1307: Timing: Started: 2015-08-07
12  2015-08-07 19:46:55,088 INFO      pytan.handler.QuestionPoller: ID 1307: Progress Changed 50% (1 of 2
13  2015-08-07 19:47:00,092 DEBUG     pytan.handler.QuestionPoller: ID 1307: Progress: Tested: 1, Passed:
14  2015-08-07 19:47:00,092 DEBUG     pytan.handler.QuestionPoller: ID 1307: Timing: Started: 2015-08-07
15  2015-08-07 19:47:05,096 DEBUG     pytan.handler.QuestionPoller: ID 1307: Progress: Tested: 1, Passed:
16  2015-08-07 19:47:05,096 DEBUG     pytan.handler.QuestionPoller: ID 1307: Timing: Started: 2015-08-07
17  2015-08-07 19:47:10,102 DEBUG     pytan.handler.QuestionPoller: ID 1307: Progress: Tested: 2, Passed:
18  2015-08-07 19:47:10,102 DEBUG     pytan.handler.QuestionPoller: ID 1307: Timing: Started: 2015-08-07
19  2015-08-07 19:47:10,102 INFO      pytan.handler.QuestionPoller: ID 1307: Progress Changed 100% (2 of
20  2015-08-07 19:47:10,102 INFO      pytan.handler.QuestionPoller: ID 1307: Reached Threshold of 99% (2
21
22  print the export_str returned from export_obj():
23  Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
24  2015-08-07 19:46:14,930 DEBUG     pytan.handler.QuestionPoller: ID 1306: id resolved to 1306
25  2015-08-07 19:46:14,930 DEBUG     pytan.handler.QuestionPoller: ID 1306: expiration resolved to 2015-
26  2015-08-07 19:46:14,930 DEBUG     pytan.handler.QuestionPoller: ID 1306: query_text resolved to Get C
27  2015-08-07 19:46:14,930 DEBUG     pytan.handler.QuestionPoller: ID 1306: id resolved to 1306
28  2015-08-07 19:46:14,930 DEBUG     pytan.handler.QuestionPoller: ID 1306: Object Info resolved to Ques
29  2015-08-07 19:46:14,933 DEBUG     pytan.handler.QuestionPoller: ID 1306: Progress: Tested: 0, Passed:
30  2015-08-07 19:46:14,933 DEBUG     pytan.handler.QuestionPoller: ID 1306: Timing: Started: 2015-08-07
31  2015-08-07 19:46:14,933 INFO      pytan.handler.QuestionPoller: ID 1306: Progress Changed 0% (0 of 2)
32  2015-08-07 19:46:19,940 DEBUG     pytan.handler.QuestionPoller: ID 1306: Progress: Tested: 0, Passed:
33  2015-08-07 19:46:19,940 DEBUG     pytan.handler.QuestionPoller: ID 1306: Timing: Started: 2015-08-07
34  2015-08-07 19:46:24,944 DEBUG     pytan.handler.QuestionPoller: ID 1306: Progress: Tested: 0, Passed:
35  2015-08-07 19:46:24,944 DEBUG     pytan.handler.QuestionPoller: ID 1306: Timing: Started: 2015-08-07
36  2015-08-07 19:46:29,951 DEBUG     pytan.handler.QuestionPoller: ID 1306: Progress: Tested: 0, Passed:
37  2015-08-07 19:46:29,951 DEBUG     pytan.handler.QuestionPoller: ID 1306: Timing: Started: 2015-08-07
38  ..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
1   import os
2   import sys
3   sys.dont_write_bytecode = True
4
5   # Determine our script name, script dir
6   my_file = os.path.abspath(sys.argv[0])
7   my_dir = os.path.dirname(my_file)
8
9   # determine the pytan lib dir and add it to the path
10  parent_dir = os.path.dirname(my_dir)
11  pytan_root_dir = os.path.dirname(parent_dir)
12  lib_dir = os.path.join(pytan_root_dir, 'lib')
13  path_adds = [lib_dir]
```

```
14
15  for aa in path_adds:
16      if aa not in sys.path:
17          sys.path.append(aa)
18
19
20  # connection info for Tanium Server
21  USERNAME = "Tanium User"
22  PASSWORD = "T@n!um"
23  HOST = "172.16.31.128"
24  PORT = "443"
25
26  # Logging conrols
27  LOGLEVEL = 2
28  DEBUGFORMAT = False
29
30  import tempfile
31
32  import pytan
33  handler = pytan.Handler(
34      username=USERNAME,
35      password=PASSWORD,
36      host=HOST,
37      port=PORT,
38      loglevel=LOGLEVEL,
39      debugformat=DEBUGFORMAT,
40  )
41
42  print handler
43
44  # setup the export_obj kwargs for later
45  export_kwargs = {}
46  export_kwargs["export_format"] = u'csv'
47  export_kwargs["expand_grouped_columns"] = True
48
49  # ask the question that will provide the resultset that we want to use
50  ask_kwargs = {
51      'qtype': 'manual',
52      'sensors': [
53          "Computer Name", "IP Route Details", "IP Address",
54          'Folder Name Search with RegEx Match{dirname=Program Files,regex=.*Shared.*}',
55      ],
56  }
57  response = handler.ask(**ask_kwargs)
58
59  # export the object to a string
60  # (we could just as easily export to a file using export_to_report_file)
61  export_kwargs['obj'] = response['question_results']
62  export_str = handler.export_obj(**export_kwargs)
63
64
65  print ""
66  print "print the export_str returned from export_obj():"
67  if len(out.splitlines()) > 15:
68      out = out.splitlines()[0:15]
69      out.append('..trimmed for brevity..')
70      out = '\n'.join(out)
71
```

```
72  print out
```

Output from Python Code

```
1   Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2   2015-08-07 19:47:10,227 DEBUG    pytan.handler.QuestionPoller: ID 1309: id resolved to 1309
3   2015-08-07 19:47:10,227 DEBUG    pytan.handler.QuestionPoller: ID 1309: expiration resolved to 2015-
4   2015-08-07 19:47:10,227 DEBUG    pytan.handler.QuestionPoller: ID 1309: query_text resolved to Get C
5   2015-08-07 19:47:10,227 DEBUG    pytan.handler.QuestionPoller: ID 1309: id resolved to 1309
6   2015-08-07 19:47:10,227 DEBUG    pytan.handler.QuestionPoller: ID 1309: Object Info resolved to Ques
7   2015-08-07 19:47:10,231 DEBUG    pytan.handler.QuestionPoller: ID 1309: Progress: Tested: 0, Passed:
8   2015-08-07 19:47:10,231 DEBUG    pytan.handler.QuestionPoller: ID 1309: Timing: Started: 2015-08-07
9   2015-08-07 19:47:10,231 INFO     pytan.handler.QuestionPoller: ID 1309: Progress Changed 0% (0 of 2)
10  2015-08-07 19:47:15,235 DEBUG    pytan.handler.QuestionPoller: ID 1309: Progress: Tested: 0, Passed:
11  2015-08-07 19:47:15,235 DEBUG    pytan.handler.QuestionPoller: ID 1309: Timing: Started: 2015-08-07
12  2015-08-07 19:47:20,239 DEBUG    pytan.handler.QuestionPoller: ID 1309: Progress: Tested: 1, Passed:
13  2015-08-07 19:47:20,239 DEBUG    pytan.handler.QuestionPoller: ID 1309: Timing: Started: 2015-08-07
14  2015-08-07 19:47:20,239 INFO     pytan.handler.QuestionPoller: ID 1309: Progress Changed 50% (1 of 2
15  2015-08-07 19:47:25,243 DEBUG    pytan.handler.QuestionPoller: ID 1309: Progress: Tested: 1, Passed:
16  2015-08-07 19:47:25,243 DEBUG    pytan.handler.QuestionPoller: ID 1309: Timing: Started: 2015-08-07
17  2015-08-07 19:47:30,250 DEBUG    pytan.handler.QuestionPoller: ID 1309: Progress: Tested: 1, Passed:
18  2015-08-07 19:47:30,250 DEBUG    pytan.handler.QuestionPoller: ID 1309: Timing: Started: 2015-08-07
19  2015-08-07 19:47:35,255 DEBUG    pytan.handler.QuestionPoller: ID 1309: Progress: Tested: 1, Passed:
20  2015-08-07 19:47:35,255 DEBUG    pytan.handler.QuestionPoller: ID 1309: Timing: Started: 2015-08-07
21  2015-08-07 19:47:40,259 DEBUG    pytan.handler.QuestionPoller: ID 1309: Progress: Tested: 2, Passed:
22  2015-08-07 19:47:40,259 DEBUG    pytan.handler.QuestionPoller: ID 1309: Timing: Started: 2015-08-07
23  2015-08-07 19:47:40,259 INFO     pytan.handler.QuestionPoller: ID 1309: Progress Changed 100% (2 of
24  2015-08-07 19:47:40,259 INFO     pytan.handler.QuestionPoller: ID 1309: Reached Threshold of 99% (2
25
26  print the export_str returned from export_obj():
27  Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
28  2015-08-07 19:46:50,080 DEBUG    pytan.handler.QuestionPoller: ID 1307: id resolved to 1307
29  2015-08-07 19:46:50,080 DEBUG    pytan.handler.QuestionPoller: ID 1307: expiration resolved to 2015-
30  2015-08-07 19:46:50,080 DEBUG    pytan.handler.QuestionPoller: ID 1307: query_text resolved to Get C
31  2015-08-07 19:46:50,080 DEBUG    pytan.handler.QuestionPoller: ID 1307: id resolved to 1307
32  2015-08-07 19:46:50,080 DEBUG    pytan.handler.QuestionPoller: ID 1307: Object Info resolved to Ques
33  2015-08-07 19:46:50,083 DEBUG    pytan.handler.QuestionPoller: ID 1307: Progress: Tested: 0, Passed:
34  2015-08-07 19:46:50,083 DEBUG    pytan.handler.QuestionPoller: ID 1307: Timing: Started: 2015-08-07
35  2015-08-07 19:46:50,083 INFO     pytan.handler.QuestionPoller: ID 1307: Progress Changed 0% (0 of 2)
36  2015-08-07 19:46:55,088 DEBUG    pytan.handler.QuestionPoller: ID 1307: Progress: Tested: 1, Passed:
37  2015-08-07 19:46:55,088 DEBUG    pytan.handler.QuestionPoller: ID 1307: Timing: Started: 2015-08-07
38  2015-08-07 19:46:55,088 INFO     pytan.handler.QuestionPoller: ID 1307: Progress Changed 50% (1 of 2
39  2015-08-07 19:47:00,092 DEBUG    pytan.handler.QuestionPoller: ID 1307: Progress: Tested: 1, Passed:
40  2015-08-07 19:47:00,092 DEBUG    pytan.handler.QuestionPoller: ID 1307: Timing: Started: 2015-08-07
41  2015-08-07 19:47:05,096 DEBUG    pytan.handler.QuestionPoller: ID 1307: Progress: Tested: 1, Passed:
42  ..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
1  import os
2  import sys
3  sys.dont_write_bytecode = True
4
5  # Determine our script name, script dir
6  my_file = os.path.abspath(sys.argv[0])
7  my_dir = os.path.dirname(my_file)
8
9  # determine the pytan lib dir and add it to the path
10 parent_dir = os.path.dirname(my_dir)
11 pytan_root_dir = os.path.dirname(parent_dir)
12 lib_dir = os.path.join(pytan_root_dir, 'lib')
13 path_adds = [lib_dir]
14
15 for aa in path_adds:
16     if aa not in sys.path:
17         sys.path.append(aa)
18
19
20 # connection info for Tanium Server
21 USERNAME = "Tanium User"
22 PASSWORD = "T@n!um"
23 HOST = "172.16.31.128"
24 PORT = "443"
25
26 # Logging conrols
27 LOGLEVEL = 2
28 DEBUGFORMAT = False
29
30 import tempfile
31
32 import pytan
33 handler = pytan.Handler(
34     username=USERNAME,
35     password=PASSWORD,
36     host=HOST,
37     port=PORT,
38     loglevel=LOGLEVEL,
39     debugformat=DEBUGFORMAT,
40 )
41
42 print handler
43
44 # setup the export_obj kwargs for later
45 export_kwargs = {}
46 export_kwargs["header_sort"] = True
47 export_kwargs["export_format"] = u'csv'
48 export_kwargs["header_add_type"] = True
49 export_kwargs["expand_grouped_columns"] = True
50 export_kwargs["header_add_sensor"] = True
51
52 # ask the question that will provide the resultset that we want to use
53 ask_kwargs = {
54     'qtype': 'manual',
55     'sensors': [
56         "Computer Name", "IP Route Details", "IP Address",
57         'Folder Name Search with RegEx Match{dirname=Program Files,regex=.*Shared.*}',
58     ],
```

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

Output from Python Code

```
1    Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2    2015-08-07 19:47:40,405 DEBUG     pytan.handler.QuestionPoller: ID 1310: id resolved to 1310
3    2015-08-07 19:47:40,405 DEBUG     pytan.handler.QuestionPoller: ID 1310: expiration resolved to 2015-
4    2015-08-07 19:47:40,405 DEBUG     pytan.handler.QuestionPoller: ID 1310: query_text resolved to Get C
5    2015-08-07 19:47:40,405 DEBUG     pytan.handler.QuestionPoller: ID 1310: id resolved to 1310
6    2015-08-07 19:47:40,405 DEBUG     pytan.handler.QuestionPoller: ID 1310: Object Info resolved to Ques
7    2015-08-07 19:47:40,408 DEBUG     pytan.handler.QuestionPoller: ID 1310: Progress: Tested: 0, Passed:
8    2015-08-07 19:47:40,408 DEBUG     pytan.handler.QuestionPoller: ID 1310: Timing: Started: 2015-08-07
9    2015-08-07 19:47:40,408 INFO      pytan.handler.QuestionPoller: ID 1310: Progress Changed 0% (0 of 2)
10   2015-08-07 19:47:45,417 DEBUG     pytan.handler.QuestionPoller: ID 1310: Progress: Tested: 0, Passed:
11   2015-08-07 19:47:45,417 DEBUG     pytan.handler.QuestionPoller: ID 1310: Timing: Started: 2015-08-07
12   2015-08-07 19:47:50,421 DEBUG     pytan.handler.QuestionPoller: ID 1310: Progress: Tested: 0, Passed:
13   2015-08-07 19:47:50,421 DEBUG     pytan.handler.QuestionPoller: ID 1310: Timing: Started: 2015-08-07
14   2015-08-07 19:47:55,425 DEBUG     pytan.handler.QuestionPoller: ID 1310: Progress: Tested: 0, Passed:
15   2015-08-07 19:47:55,425 DEBUG     pytan.handler.QuestionPoller: ID 1310: Timing: Started: 2015-08-07
16   2015-08-07 19:48:00,431 DEBUG     pytan.handler.QuestionPoller: ID 1310: Progress: Tested: 0, Passed:
17   2015-08-07 19:48:00,431 DEBUG     pytan.handler.QuestionPoller: ID 1310: Timing: Started: 2015-08-07
18   2015-08-07 19:48:05,435 DEBUG     pytan.handler.QuestionPoller: ID 1310: Progress: Tested: 0, Passed:
19   2015-08-07 19:48:05,435 DEBUG     pytan.handler.QuestionPoller: ID 1310: Timing: Started: 2015-08-07
20   2015-08-07 19:48:10,440 DEBUG     pytan.handler.QuestionPoller: ID 1310: Progress: Tested: 0, Passed:
21   2015-08-07 19:48:10,440 DEBUG     pytan.handler.QuestionPoller: ID 1310: Timing: Started: 2015-08-07
22   2015-08-07 19:48:15,444 DEBUG     pytan.handler.QuestionPoller: ID 1310: Progress: Tested: 0, Passed:
23   2015-08-07 19:48:15,444 DEBUG     pytan.handler.QuestionPoller: ID 1310: Timing: Started: 2015-08-07
24   2015-08-07 19:48:20,449 DEBUG     pytan.handler.QuestionPoller: ID 1310: Progress: Tested: 0, Passed:
25   2015-08-07 19:48:20,449 DEBUG     pytan.handler.QuestionPoller: ID 1310: Timing: Started: 2015-08-07
26   2015-08-07 19:48:25,453 DEBUG     pytan.handler.QuestionPoller: ID 1310: Progress: Tested: 0, Passed:
27   2015-08-07 19:48:25,453 DEBUG     pytan.handler.QuestionPoller: ID 1310: Timing: Started: 2015-08-07
28   2015-08-07 19:48:30,459 DEBUG     pytan.handler.QuestionPoller: ID 1310: Progress: Tested: 0, Passed:
29   2015-08-07 19:48:30,459 DEBUG     pytan.handler.QuestionPoller: ID 1310: Timing: Started: 2015-08-07
30   2015-08-07 19:48:35,467 DEBUG     pytan.handler.QuestionPoller: ID 1310: Progress: Tested: 0, Passed:
31   2015-08-07 19:48:35,467 DEBUG     pytan.handler.QuestionPoller: ID 1310: Timing: Started: 2015-08-07
32   2015-08-07 19:48:40,473 DEBUG     pytan.handler.QuestionPoller: ID 1310: Progress: Tested: 0, Passed:
33   2015-08-07 19:48:40,473 DEBUG     pytan.handler.QuestionPoller: ID 1310: Timing: Started: 2015-08-07
34   2015-08-07 19:48:45,481 DEBUG     pytan.handler.QuestionPoller: ID 1310: Progress: Tested: 0, Passed:
35   2015-08-07 19:48:45,481 DEBUG     pytan.handler.QuestionPoller: ID 1310: Timing: Started: 2015-08-07
36   2015-08-07 19:48:50,489 DEBUG     pytan.handler.QuestionPoller: ID 1310: Progress: Tested: 0, Passed:
37   2015-08-07 19:48:50,489 DEBUG     pytan.handler.QuestionPoller: ID 1310: Timing: Started: 2015-08-07
```

```
38   2015-08-07 19:48:55,493 DEBUG     pytan.handler.QuestionPoller: ID 1310: Progress: Tested: 1, Passed:
39   2015-08-07 19:48:55,493 DEBUG     pytan.handler.QuestionPoller: ID 1310: Timing: Started: 2015-08-07
40   2015-08-07 19:48:55,493 INFO      pytan.handler.QuestionPoller: ID 1310: Progress Changed 50% (1 of 2
41   2015-08-07 19:49:00,497 DEBUG     pytan.handler.QuestionPoller: ID 1310: Progress: Tested: 1, Passed:
42   2015-08-07 19:49:00,497 DEBUG     pytan.handler.QuestionPoller: ID 1310: Timing: Started: 2015-08-07
43   2015-08-07 19:49:05,502 DEBUG     pytan.handler.QuestionPoller: ID 1310: Progress: Tested: 1, Passed:
44   2015-08-07 19:49:05,502 DEBUG     pytan.handler.QuestionPoller: ID 1310: Timing: Started: 2015-08-07
45   2015-08-07 19:49:10,508 DEBUG     pytan.handler.QuestionPoller: ID 1310: Progress: Tested: 2, Passed:
46   2015-08-07 19:49:10,508 DEBUG     pytan.handler.QuestionPoller: ID 1310: Timing: Started: 2015-08-07
47   2015-08-07 19:49:10,508 INFO      pytan.handler.QuestionPoller: ID 1310: Progress Changed 100% (2 of
48   2015-08-07 19:49:10,508 INFO      pytan.handler.QuestionPoller: ID 1310: Reached Threshold of 99% (2
49
50   print the export_str returned from export_obj():
51   Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
52   2015-08-07 19:47:10,227 DEBUG     pytan.handler.QuestionPoller: ID 1309: id resolved to 1309
53   2015-08-07 19:47:10,227 DEBUG     pytan.handler.QuestionPoller: ID 1309: expiration resolved to 2015-
54   2015-08-07 19:47:10,227 DEBUG     pytan.handler.QuestionPoller: ID 1309: query_text resolved to Get C
55   2015-08-07 19:47:10,227 DEBUG     pytan.handler.QuestionPoller: ID 1309: id resolved to 1309
56   2015-08-07 19:47:10,227 DEBUG     pytan.handler.QuestionPoller: ID 1309: Object Info resolved to Ques
57   2015-08-07 19:47:10,231 DEBUG     pytan.handler.QuestionPoller: ID 1309: Progress: Tested: 0, Passed:
58   2015-08-07 19:47:10,231 DEBUG     pytan.handler.QuestionPoller: ID 1309: Timing: Started: 2015-08-07
59   2015-08-07 19:47:10,231 INFO      pytan.handler.QuestionPoller: ID 1309: Progress Changed 0% (0 of 2)
60   2015-08-07 19:47:15,235 DEBUG     pytan.handler.QuestionPoller: ID 1309: Progress: Tested: 0, Passed:
61   2015-08-07 19:47:15,235 DEBUG     pytan.handler.QuestionPoller: ID 1309: Timing: Started: 2015-08-07
62   2015-08-07 19:47:20,239 DEBUG     pytan.handler.QuestionPoller: ID 1309: Progress: Tested: 1, Passed:
63   2015-08-07 19:47:20,239 DEBUG     pytan.handler.QuestionPoller: ID 1309: Timing: Started: 2015-08-07
64   2015-08-07 19:47:20,239 INFO      pytan.handler.QuestionPoller: ID 1309: Progress Changed 50% (1 of 2
65   2015-08-07 19:47:25,243 DEBUG     pytan.handler.QuestionPoller: ID 1309: Progress: Tested: 1, Passed:
66   ..trimmed for brevity..
```

Export resultset json

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

Example Python Code

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

```
20   # connection info for Tanium Server
21   USERNAME = "Tanium User"
22   PASSWORD = "T@n!um"
23   HOST = "172.16.31.128"
24   PORT = "443"
25
26   # Logging conrols
27   LOGLEVEL = 2
28   DEBUGFORMAT = False
29
30   import tempfile
31
32   import pytan
33   handler = pytan.Handler(
34       username=USERNAME,
35       password=PASSWORD,
36       host=HOST,
37       port=PORT,
38       loglevel=LOGLEVEL,
39       debugformat=DEBUGFORMAT,
40   )
41
42   print handler
43
44   # setup the export_obj kwargs for later
45   export_kwargs = {}
46   export_kwargs["export_format"] = u'json'
47
48   # ask the question that will provide the resultset that we want to use
49   ask_kwargs = {
50       'qtype': 'manual',
51       'sensors': [
52           "Computer Name", "IP Route Details", "IP Address",
53           'Folder Name Search with RegEx Match{dirname=Program Files,regex=.*Shared.*}',
54       ],
55   }
56   response = handler.ask(**ask_kwargs)
57
58   # export the object to a string
59   # (we could just as easily export to a file using export_to_report_file)
60   export_kwargs['obj'] = response['question_results']
61   export_str = handler.export_obj(**export_kwargs)
62
63
64   print ""
65   print "print the export_str returned from export_obj():"
66   if len(out.splitlines()) > 15:
67       out = out.splitlines()[0:15]
68       out.append('..trimmed for brevity..')
69       out = '\n'.join(out)
70
71   print out
```

Output from Python Code

```
1   Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2   2015-08-07 19:49:10,709 DEBUG    pytan.handler.QuestionPoller: ID 1311: id resolved to 1311
```

```
3    2015-08-07 19:49:10,709 DEBUG     pytan.handler.QuestionPoller: ID 1311: expiration resolved to 2015-
4    2015-08-07 19:49:10,709 DEBUG     pytan.handler.QuestionPoller: ID 1311: query_text resolved to Get C
5    2015-08-07 19:49:10,709 DEBUG     pytan.handler.QuestionPoller: ID 1311: id resolved to 1311
6    2015-08-07 19:49:10,709 DEBUG     pytan.handler.QuestionPoller: ID 1311: Object Info resolved to Ques
7    2015-08-07 19:49:10,713 DEBUG     pytan.handler.QuestionPoller: ID 1311: Progress: Tested: 0, Passed:
8    2015-08-07 19:49:10,713 DEBUG     pytan.handler.QuestionPoller: ID 1311: Timing: Started: 2015-08-07
9    2015-08-07 19:49:10,713 INFO      pytan.handler.QuestionPoller: ID 1311: Progress Changed 0% (0 of 2)
10   2015-08-07 19:49:15,721 DEBUG     pytan.handler.QuestionPoller: ID 1311: Progress: Tested: 0, Passed:
11   2015-08-07 19:49:15,721 DEBUG     pytan.handler.QuestionPoller: ID 1311: Timing: Started: 2015-08-07
12   2015-08-07 19:49:20,725 DEBUG     pytan.handler.QuestionPoller: ID 1311: Progress: Tested: 0, Passed:
13   2015-08-07 19:49:20,726 DEBUG     pytan.handler.QuestionPoller: ID 1311: Timing: Started: 2015-08-07
14   2015-08-07 19:49:25,730 DEBUG     pytan.handler.QuestionPoller: ID 1311: Progress: Tested: 0, Passed:
15   2015-08-07 19:49:25,730 DEBUG     pytan.handler.QuestionPoller: ID 1311: Timing: Started: 2015-08-07
16   2015-08-07 19:49:30,739 DEBUG     pytan.handler.QuestionPoller: ID 1311: Progress: Tested: 1, Passed:
17   2015-08-07 19:49:30,739 DEBUG     pytan.handler.QuestionPoller: ID 1311: Timing: Started: 2015-08-07
18   2015-08-07 19:49:30,739 INFO      pytan.handler.QuestionPoller: ID 1311: Progress Changed 50% (1 of 2
19   2015-08-07 19:49:35,743 DEBUG     pytan.handler.QuestionPoller: ID 1311: Progress: Tested: 1, Passed:
20   2015-08-07 19:49:35,744 DEBUG     pytan.handler.QuestionPoller: ID 1311: Timing: Started: 2015-08-07
21   2015-08-07 19:49:40,751 DEBUG     pytan.handler.QuestionPoller: ID 1311: Progress: Tested: 2, Passed:
22   2015-08-07 19:49:40,751 DEBUG     pytan.handler.QuestionPoller: ID 1311: Timing: Started: 2015-08-07
23   2015-08-07 19:49:40,751 INFO      pytan.handler.QuestionPoller: ID 1311: Progress Changed 100% (2 of
24   2015-08-07 19:49:40,751 INFO      pytan.handler.QuestionPoller: ID 1311: Reached Threshold of 99% (2
25
26   print the export_str returned from export_obj():
27   Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
28   2015-08-07 19:47:40,405 DEBUG     pytan.handler.QuestionPoller: ID 1310: id resolved to 1310
29   2015-08-07 19:47:40,405 DEBUG     pytan.handler.QuestionPoller: ID 1310: expiration resolved to 2015-
30   2015-08-07 19:47:40,405 DEBUG     pytan.handler.QuestionPoller: ID 1310: query_text resolved to Get C
31   2015-08-07 19:47:40,405 DEBUG     pytan.handler.QuestionPoller: ID 1310: id resolved to 1310
32   2015-08-07 19:47:40,405 DEBUG     pytan.handler.QuestionPoller: ID 1310: Object Info resolved to Ques
33   2015-08-07 19:47:40,408 DEBUG     pytan.handler.QuestionPoller: ID 1310: Progress: Tested: 0, Passed:
34   2015-08-07 19:47:40,408 DEBUG     pytan.handler.QuestionPoller: ID 1310: Timing: Started: 2015-08-07
35   2015-08-07 19:47:40,408 INFO      pytan.handler.QuestionPoller: ID 1310: Progress Changed 0% (0 of 2)
36   2015-08-07 19:47:45,417 DEBUG     pytan.handler.QuestionPoller: ID 1310: Progress: Tested: 0, Passed:
37   2015-08-07 19:47:45,417 DEBUG     pytan.handler.QuestionPoller: ID 1310: Timing: Started: 2015-08-07
38   2015-08-07 19:47:50,421 DEBUG     pytan.handler.QuestionPoller: ID 1310: Progress: Tested: 0, Passed:
39   2015-08-07 19:47:50,421 DEBUG     pytan.handler.QuestionPoller: ID 1310: Timing: Started: 2015-08-07
40   2015-08-07 19:47:55,425 DEBUG     pytan.handler.QuestionPoller: ID 1310: Progress: Tested: 0, Passed:
41   2015-08-07 19:47:55,425 DEBUG     pytan.handler.QuestionPoller: ID 1310: Timing: Started: 2015-08-07
42   ..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

```python
import os
import sys
sys.dont_write_bytecode = True

# Determine our script name, script dir
my_file = os.path.abspath(sys.argv[0])
my_dir = os.path.dirname(my_file)

```

```
 9   # determine the pytan lib dir and add it to the path
10   parent_dir = os.path.dirname(my_dir)
11   pytan_root_dir = os.path.dirname(parent_dir)
12   lib_dir = os.path.join(pytan_root_dir, 'lib')
13   path_adds = [lib_dir]
14
15   for aa in path_adds:
16       if aa not in sys.path:
17           sys.path.append(aa)
18
19
20   # connection info for Tanium Server
21   USERNAME = "Tanium User"
22   PASSWORD = "T@n!um"
23   HOST = "172.16.31.128"
24   PORT = "443"
25
26   # Logging conrols
27   LOGLEVEL = 2
28   DEBUGFORMAT = False
29
30   import tempfile
31
32   import pytan
33   handler = pytan.Handler(
34       username=USERNAME,
35       password=PASSWORD,
36       host=HOST,
37       port=PORT,
38       loglevel=LOGLEVEL,
39       debugformat=DEBUGFORMAT,
40   )
41
42   print handler
43
44   # setup the export_obj kwargs for later
45   export_kwargs = {}
46   export_kwargs["export_format"] = u'csv'
47   export_kwargs["header_sort"] = []
48
49   # ask the question that will provide the resultset that we want to use
50   ask_kwargs = {
51       'qtype': 'manual',
52       'sensors': [
53           "Computer Name", "IP Route Details", "IP Address",
54           'Folder Name Search with RegEx Match{dirname=Program Files,regex=.*Shared.*}',
55       ],
56   }
57   response = handler.ask(**ask_kwargs)
58
59   # export the object to a string
60   # (we could just as easily export to a file using export_to_report_file)
61   export_kwargs['obj'] = response['question_results']
62   export_str = handler.export_obj(**export_kwargs)
63
64
65   print ""
66   print "print the export_str returned from export_obj():"
```

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

Output from Python Code

```
1    Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2    2015-08-07 19:49:40,850 DEBUG     pytan.handler.QuestionPoller: ID 1312: id resolved to 1312
3    2015-08-07 19:49:40,850 DEBUG     pytan.handler.QuestionPoller: ID 1312: expiration resolved to 2015-
4    2015-08-07 19:49:40,850 DEBUG     pytan.handler.QuestionPoller: ID 1312: query_text resolved to Get C
5    2015-08-07 19:49:40,850 DEBUG     pytan.handler.QuestionPoller: ID 1312: id resolved to 1312
6    2015-08-07 19:49:40,850 DEBUG     pytan.handler.QuestionPoller: ID 1312: Object Info resolved to Ques
7    2015-08-07 19:49:40,853 DEBUG     pytan.handler.QuestionPoller: ID 1312: Progress: Tested: 0, Passed:
8    2015-08-07 19:49:40,853 DEBUG     pytan.handler.QuestionPoller: ID 1312: Timing: Started: 2015-08-07
9    2015-08-07 19:49:40,854 INFO      pytan.handler.QuestionPoller: ID 1312: Progress Changed 0% (0 of 2)
10   2015-08-07 19:49:45,859 DEBUG     pytan.handler.QuestionPoller: ID 1312: Progress: Tested: 0, Passed:
11   2015-08-07 19:49:45,859 DEBUG     pytan.handler.QuestionPoller: ID 1312: Timing: Started: 2015-08-07
12   2015-08-07 19:49:50,863 DEBUG     pytan.handler.QuestionPoller: ID 1312: Progress: Tested: 0, Passed:
13   2015-08-07 19:49:50,863 DEBUG     pytan.handler.QuestionPoller: ID 1312: Timing: Started: 2015-08-07
14   2015-08-07 19:49:55,870 DEBUG     pytan.handler.QuestionPoller: ID 1312: Progress: Tested: 0, Passed:
15   2015-08-07 19:49:55,870 DEBUG     pytan.handler.QuestionPoller: ID 1312: Timing: Started: 2015-08-07
16   2015-08-07 19:50:00,877 DEBUG     pytan.handler.QuestionPoller: ID 1312: Progress: Tested: 0, Passed:
17   2015-08-07 19:50:00,877 DEBUG     pytan.handler.QuestionPoller: ID 1312: Timing: Started: 2015-08-07
18   2015-08-07 19:50:05,881 DEBUG     pytan.handler.QuestionPoller: ID 1312: Progress: Tested: 0, Passed:
19   2015-08-07 19:50:05,881 DEBUG     pytan.handler.QuestionPoller: ID 1312: Timing: Started: 2015-08-07
20   2015-08-07 19:50:10,886 DEBUG     pytan.handler.QuestionPoller: ID 1312: Progress: Tested: 0, Passed:
21   2015-08-07 19:50:10,886 DEBUG     pytan.handler.QuestionPoller: ID 1312: Timing: Started: 2015-08-07
22   2015-08-07 19:50:15,891 DEBUG     pytan.handler.QuestionPoller: ID 1312: Progress: Tested: 0, Passed:
23   2015-08-07 19:50:15,891 DEBUG     pytan.handler.QuestionPoller: ID 1312: Timing: Started: 2015-08-07
24   2015-08-07 19:50:20,896 DEBUG     pytan.handler.QuestionPoller: ID 1312: Progress: Tested: 0, Passed:
25   2015-08-07 19:50:20,896 DEBUG     pytan.handler.QuestionPoller: ID 1312: Timing: Started: 2015-08-07
26   2015-08-07 19:50:25,901 DEBUG     pytan.handler.QuestionPoller: ID 1312: Progress: Tested: 0, Passed:
27   2015-08-07 19:50:25,901 DEBUG     pytan.handler.QuestionPoller: ID 1312: Timing: Started: 2015-08-07
28   2015-08-07 19:50:30,906 DEBUG     pytan.handler.QuestionPoller: ID 1312: Progress: Tested: 0, Passed:
29   2015-08-07 19:50:30,906 DEBUG     pytan.handler.QuestionPoller: ID 1312: Timing: Started: 2015-08-07
30   2015-08-07 19:50:35,910 DEBUG     pytan.handler.QuestionPoller: ID 1312: Progress: Tested: 0, Passed:
31   2015-08-07 19:50:35,910 DEBUG     pytan.handler.QuestionPoller: ID 1312: Timing: Started: 2015-08-07
32   2015-08-07 19:50:40,915 DEBUG     pytan.handler.QuestionPoller: ID 1312: Progress: Tested: 0, Passed:
33   2015-08-07 19:50:40,915 DEBUG     pytan.handler.QuestionPoller: ID 1312: Timing: Started: 2015-08-07
34   2015-08-07 19:50:45,919 DEBUG     pytan.handler.QuestionPoller: ID 1312: Progress: Tested: 0, Passed:
35   2015-08-07 19:50:45,919 DEBUG     pytan.handler.QuestionPoller: ID 1312: Timing: Started: 2015-08-07
36   2015-08-07 19:50:50,923 DEBUG     pytan.handler.QuestionPoller: ID 1312: Progress: Tested: 0, Passed:
37   2015-08-07 19:50:50,923 DEBUG     pytan.handler.QuestionPoller: ID 1312: Timing: Started: 2015-08-07
38   2015-08-07 19:50:55,928 DEBUG     pytan.handler.QuestionPoller: ID 1312: Progress: Tested: 0, Passed:
39   2015-08-07 19:50:55,928 DEBUG     pytan.handler.QuestionPoller: ID 1312: Timing: Started: 2015-08-07
40   2015-08-07 19:51:00,934 DEBUG     pytan.handler.QuestionPoller: ID 1312: Progress: Tested: 0, Passed:
41   2015-08-07 19:51:00,935 DEBUG     pytan.handler.QuestionPoller: ID 1312: Timing: Started: 2015-08-07
42   2015-08-07 19:51:05,939 DEBUG     pytan.handler.QuestionPoller: ID 1312: Progress: Tested: 0, Passed:
43   2015-08-07 19:51:05,939 DEBUG     pytan.handler.QuestionPoller: ID 1312: Timing: Started: 2015-08-07
44   2015-08-07 19:51:10,947 DEBUG     pytan.handler.QuestionPoller: ID 1312: Progress: Tested: 2, Passed:
45   2015-08-07 19:51:10,947 DEBUG     pytan.handler.QuestionPoller: ID 1312: Timing: Started: 2015-08-07
46   2015-08-07 19:51:10,947 INFO      pytan.handler.QuestionPoller: ID 1312: Progress Changed 100% (2 of
47   2015-08-07 19:51:10,947 INFO      pytan.handler.QuestionPoller: ID 1312: Reached Threshold of 99% (2
48
```

```
49  print the export_str returned from export_obj():
50  Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
51  2015-08-07 19:49:10,709 DEBUG     pytan.handler.QuestionPoller: ID 1311: id resolved to 1311
52  2015-08-07 19:49:10,709 DEBUG     pytan.handler.QuestionPoller: ID 1311: expiration resolved to 2015-
53  2015-08-07 19:49:10,709 DEBUG     pytan.handler.QuestionPoller: ID 1311: query_text resolved to Get C
54  2015-08-07 19:49:10,709 DEBUG     pytan.handler.QuestionPoller: ID 1311: id resolved to 1311
55  2015-08-07 19:49:10,709 DEBUG     pytan.handler.QuestionPoller: ID 1311: Object Info resolved to Ques
56  2015-08-07 19:49:10,713 DEBUG     pytan.handler.QuestionPoller: ID 1311: Progress: Tested: 0, Passed:
57  2015-08-07 19:49:10,713 DEBUG     pytan.handler.QuestionPoller: ID 1311: Timing: Started: 2015-08-07
58  2015-08-07 19:49:10,713 INFO      pytan.handler.QuestionPoller: ID 1311: Progress Changed 0% (0 of 2)
59  2015-08-07 19:49:15,721 DEBUG     pytan.handler.QuestionPoller: ID 1311: Progress: Tested: 0, Passed:
60  2015-08-07 19:49:15,721 DEBUG     pytan.handler.QuestionPoller: ID 1311: Timing: Started: 2015-08-07
61  2015-08-07 19:49:20,725 DEBUG     pytan.handler.QuestionPoller: ID 1311: Progress: Tested: 0, Passed:
62  2015-08-07 19:49:20,726 DEBUG     pytan.handler.QuestionPoller: ID 1311: Timing: Started: 2015-08-07
63  2015-08-07 19:49:25,730 DEBUG     pytan.handler.QuestionPoller: ID 1311: Progress: Tested: 0, Passed:
64  2015-08-07 19:49:25,730 DEBUG     pytan.handler.QuestionPoller: ID 1311: Timing: Started: 2015-08-07
65  ..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
1   import os
2   import sys
3   sys.dont_write_bytecode = True
4
5   # Determine our script name, script dir
6   my_file = os.path.abspath(sys.argv[0])
7   my_dir = os.path.dirname(my_file)
8
9   # determine the pytan lib dir and add it to the path
10  parent_dir = os.path.dirname(my_dir)
11  pytan_root_dir = os.path.dirname(parent_dir)
12  lib_dir = os.path.join(pytan_root_dir, 'lib')
13  path_adds = [lib_dir]
14
15  for aa in path_adds:
16      if aa not in sys.path:
17          sys.path.append(aa)
18
19
20  # connection info for Tanium Server
21  USERNAME = "Tanium User"
22  PASSWORD = "T@n!um"
23  HOST = "172.16.31.128"
24  PORT = "443"
25
26  # Logging conrols
27  LOGLEVEL = 2
28  DEBUGFORMAT = False
29
30  import tempfile
31
```

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

Output from Python Code

```
1   Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2   2015-08-07 19:51:11,061 DEBUG    pytan.handler.QuestionPoller: ID 1313: id resolved to 1313
3   2015-08-07 19:51:11,061 DEBUG    pytan.handler.QuestionPoller: ID 1313: expiration resolved to 2015-
4   2015-08-07 19:51:11,061 DEBUG    pytan.handler.QuestionPoller: ID 1313: query_text resolved to Get C
5   2015-08-07 19:51:11,061 DEBUG    pytan.handler.QuestionPoller: ID 1313: id resolved to 1313
6   2015-08-07 19:51:11,061 DEBUG    pytan.handler.QuestionPoller: ID 1313: Object Info resolved to Ques
7   2015-08-07 19:51:11,066 DEBUG    pytan.handler.QuestionPoller: ID 1313: Progress: Tested: 0, Passed:
8   2015-08-07 19:51:11,066 DEBUG    pytan.handler.QuestionPoller: ID 1313: Timing: Started: 2015-08-07
9   2015-08-07 19:51:11,066 INFO     pytan.handler.QuestionPoller: ID 1313: Progress Changed 0% (0 of 2)
10  2015-08-07 19:51:16,074 DEBUG    pytan.handler.QuestionPoller: ID 1313: Progress: Tested: 0, Passed:
11  2015-08-07 19:51:16,074 DEBUG    pytan.handler.QuestionPoller: ID 1313: Timing: Started: 2015-08-07
12  2015-08-07 19:51:21,079 DEBUG    pytan.handler.QuestionPoller: ID 1313: Progress: Tested: 0, Passed:
13  2015-08-07 19:51:21,079 DEBUG    pytan.handler.QuestionPoller: ID 1313: Timing: Started: 2015-08-07
```

```
14   2015-08-07 19:51:26,083 DEBUG      pytan.handler.QuestionPoller: ID 1313: Progress: Tested: 0, Passed:
15   2015-08-07 19:51:26,083 DEBUG      pytan.handler.QuestionPoller: ID 1313: Timing: Started: 2015-08-07
16   2015-08-07 19:51:31,089 DEBUG      pytan.handler.QuestionPoller: ID 1313: Progress: Tested: 0, Passed:
17   2015-08-07 19:51:31,089 DEBUG      pytan.handler.QuestionPoller: ID 1313: Timing: Started: 2015-08-07
18   2015-08-07 19:51:36,093 DEBUG      pytan.handler.QuestionPoller: ID 1313: Progress: Tested: 0, Passed:
19   2015-08-07 19:51:36,093 DEBUG      pytan.handler.QuestionPoller: ID 1313: Timing: Started: 2015-08-07
20   2015-08-07 19:51:41,099 DEBUG      pytan.handler.QuestionPoller: ID 1313: Progress: Tested: 0, Passed:
21   2015-08-07 19:51:41,099 DEBUG      pytan.handler.QuestionPoller: ID 1313: Timing: Started: 2015-08-07
22   2015-08-07 19:51:46,107 DEBUG      pytan.handler.QuestionPoller: ID 1313: Progress: Tested: 0, Passed:
23   2015-08-07 19:51:46,107 DEBUG      pytan.handler.QuestionPoller: ID 1313: Timing: Started: 2015-08-07
24   2015-08-07 19:51:51,112 DEBUG      pytan.handler.QuestionPoller: ID 1313: Progress: Tested: 0, Passed:
25   2015-08-07 19:51:51,112 DEBUG      pytan.handler.QuestionPoller: ID 1313: Timing: Started: 2015-08-07
26   2015-08-07 19:51:56,117 DEBUG      pytan.handler.QuestionPoller: ID 1313: Progress: Tested: 0, Passed:
27   2015-08-07 19:51:56,117 DEBUG      pytan.handler.QuestionPoller: ID 1313: Timing: Started: 2015-08-07
28   2015-08-07 19:52:01,121 DEBUG      pytan.handler.QuestionPoller: ID 1313: Progress: Tested: 0, Passed:
29   2015-08-07 19:52:01,122 DEBUG      pytan.handler.QuestionPoller: ID 1313: Timing: Started: 2015-08-07
30   2015-08-07 19:52:06,125 DEBUG      pytan.handler.QuestionPoller: ID 1313: Progress: Tested: 0, Passed:
31   2015-08-07 19:52:06,126 DEBUG      pytan.handler.QuestionPoller: ID 1313: Timing: Started: 2015-08-07
32   2015-08-07 19:52:11,132 DEBUG      pytan.handler.QuestionPoller: ID 1313: Progress: Tested: 0, Passed:
33   2015-08-07 19:52:11,133 DEBUG      pytan.handler.QuestionPoller: ID 1313: Timing: Started: 2015-08-07
34   2015-08-07 19:52:16,136 DEBUG      pytan.handler.QuestionPoller: ID 1313: Progress: Tested: 0, Passed:
35   2015-08-07 19:52:16,136 DEBUG      pytan.handler.QuestionPoller: ID 1313: Timing: Started: 2015-08-07
36   2015-08-07 19:52:21,142 DEBUG      pytan.handler.QuestionPoller: ID 1313: Progress: Tested: 2, Passed:
37   2015-08-07 19:52:21,142 DEBUG      pytan.handler.QuestionPoller: ID 1313: Timing: Started: 2015-08-07
38   2015-08-07 19:52:21,142 INFO       pytan.handler.QuestionPoller: ID 1313: Progress Changed 100% (2 of
39   2015-08-07 19:52:21,142 INFO       pytan.handler.QuestionPoller: ID 1313: Reached Threshold of 99% (2
40
41   print the export_str returned from export_obj():
42   Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
43   2015-08-07 19:49:40,850 DEBUG      pytan.handler.QuestionPoller: ID 1312: id resolved to 1312
44   2015-08-07 19:49:40,850 DEBUG      pytan.handler.QuestionPoller: ID 1312: expiration resolved to 2015-
45   2015-08-07 19:49:40,850 DEBUG      pytan.handler.QuestionPoller: ID 1312: query_text resolved to Get C
46   2015-08-07 19:49:40,850 DEBUG      pytan.handler.QuestionPoller: ID 1312: id resolved to 1312
47   2015-08-07 19:49:40,850 DEBUG      pytan.handler.QuestionPoller: ID 1312: Object Info resolved to Ques
48   2015-08-07 19:49:40,853 DEBUG      pytan.handler.QuestionPoller: ID 1312: Progress: Tested: 0, Passed:
49   2015-08-07 19:49:40,853 DEBUG      pytan.handler.QuestionPoller: ID 1312: Timing: Started: 2015-08-07
50   2015-08-07 19:49:40,854 INFO       pytan.handler.QuestionPoller: ID 1312: Progress Changed 0% (0 of 2)
51   2015-08-07 19:49:45,859 DEBUG      pytan.handler.QuestionPoller: ID 1312: Progress: Tested: 0, Passed:
52   2015-08-07 19:49:45,859 DEBUG      pytan.handler.QuestionPoller: ID 1312: Timing: Started: 2015-08-07
53   2015-08-07 19:49:50,863 DEBUG      pytan.handler.QuestionPoller: ID 1312: Progress: Tested: 0, Passed:
54   2015-08-07 19:49:50,863 DEBUG      pytan.handler.QuestionPoller: ID 1312: Timing: Started: 2015-08-07
55   2015-08-07 19:49:55,870 DEBUG      pytan.handler.QuestionPoller: ID 1312: Progress: Tested: 0, Passed:
56   2015-08-07 19:49:55,870 DEBUG      pytan.handler.QuestionPoller: ID 1312: Timing: Started: 2015-08-07
57   ..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

```python
import os
import sys
sys.dont_write_bytecode = True

```

```python
5    # Determine our script name, script dir
6    my_file = os.path.abspath(sys.argv[0])
7    my_dir = os.path.dirname(my_file)
8
9    # determine the pytan lib dir and add it to the path
10   parent_dir = os.path.dirname(my_dir)
11   pytan_root_dir = os.path.dirname(parent_dir)
12   lib_dir = os.path.join(pytan_root_dir, 'lib')
13   path_adds = [lib_dir]
14
15   for aa in path_adds:
16       if aa not in sys.path:
17           sys.path.append(aa)
18
19
20   # connection info for Tanium Server
21   USERNAME = "Tanium User"
22   PASSWORD = "T@n!um"
23   HOST = "172.16.31.128"
24   PORT = "443"
25
26   # Logging conrols
27   LOGLEVEL = 2
28   DEBUGFORMAT = False
29
30   import tempfile
31
32   import pytan
33   handler = pytan.Handler(
34       username=USERNAME,
35       password=PASSWORD,
36       host=HOST,
37       port=PORT,
38       loglevel=LOGLEVEL,
39       debugformat=DEBUGFORMAT,
40   )
41
42   print handler
43
44   # setup the export_obj kwargs for later
45   export_kwargs = {}
46   export_kwargs["export_format"] = u'csv'
47   export_kwargs["header_sort"] = False
48
49   # ask the question that will provide the resultset that we want to use
50   ask_kwargs = {
51       'qtype': 'manual',
52       'sensors': [
53           "Computer Name", "IP Route Details", "IP Address",
54           'Folder Name Search with RegEx Match{dirname=Program Files,regex=.*Shared.*}',
55       ],
56   }
57   response = handler.ask(**ask_kwargs)
58
59   # export the object to a string
60   # (we could just as easily export to a file using export_to_report_file)
61   export_kwargs['obj'] = response['question_results']
62   export_str = handler.export_obj(**export_kwargs)
```

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

Output from Python Code

```
1    Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2    2015-08-07 19:52:21,243 DEBUG     pytan.handler.QuestionPoller: ID 1315: id resolved to 1315
3    2015-08-07 19:52:21,243 DEBUG     pytan.handler.QuestionPoller: ID 1315: expiration resolved to 2015-
4    2015-08-07 19:52:21,243 DEBUG     pytan.handler.QuestionPoller: ID 1315: query_text resolved to Get C
5    2015-08-07 19:52:21,243 DEBUG     pytan.handler.QuestionPoller: ID 1315: id resolved to 1315
6    2015-08-07 19:52:21,243 DEBUG     pytan.handler.QuestionPoller: ID 1315: Object Info resolved to Ques
7    2015-08-07 19:52:21,246 DEBUG     pytan.handler.QuestionPoller: ID 1315: Progress: Tested: 0, Passed:
8    2015-08-07 19:52:21,246 DEBUG     pytan.handler.QuestionPoller: ID 1315: Timing: Started: 2015-08-07
9    2015-08-07 19:52:21,246 INFO      pytan.handler.QuestionPoller: ID 1315: Progress Changed 0% (0 of 2)
10   2015-08-07 19:52:26,251 DEBUG     pytan.handler.QuestionPoller: ID 1315: Progress: Tested: 0, Passed:
11   2015-08-07 19:52:26,251 DEBUG     pytan.handler.QuestionPoller: ID 1315: Timing: Started: 2015-08-07
12   2015-08-07 19:52:31,255 DEBUG     pytan.handler.QuestionPoller: ID 1315: Progress: Tested: 0, Passed:
13   2015-08-07 19:52:31,255 DEBUG     pytan.handler.QuestionPoller: ID 1315: Timing: Started: 2015-08-07
14   2015-08-07 19:52:36,264 DEBUG     pytan.handler.QuestionPoller: ID 1315: Progress: Tested: 1, Passed:
15   2015-08-07 19:52:36,264 DEBUG     pytan.handler.QuestionPoller: ID 1315: Timing: Started: 2015-08-07
16   2015-08-07 19:52:36,264 INFO      pytan.handler.QuestionPoller: ID 1315: Progress Changed 50% (1 of 2
17   2015-08-07 19:52:41,271 DEBUG     pytan.handler.QuestionPoller: ID 1315: Progress: Tested: 1, Passed:
18   2015-08-07 19:52:41,271 DEBUG     pytan.handler.QuestionPoller: ID 1315: Timing: Started: 2015-08-07
19   2015-08-07 19:52:46,275 DEBUG     pytan.handler.QuestionPoller: ID 1315: Progress: Tested: 1, Passed:
20   2015-08-07 19:52:46,275 DEBUG     pytan.handler.QuestionPoller: ID 1315: Timing: Started: 2015-08-07
21   2015-08-07 19:52:51,282 DEBUG     pytan.handler.QuestionPoller: ID 1315: Progress: Tested: 2, Passed:
22   2015-08-07 19:52:51,283 DEBUG     pytan.handler.QuestionPoller: ID 1315: Timing: Started: 2015-08-07
23   2015-08-07 19:52:51,283 INFO      pytan.handler.QuestionPoller: ID 1315: Progress Changed 100% (2 of
24   2015-08-07 19:52:51,283 INFO      pytan.handler.QuestionPoller: ID 1315: Reached Threshold of 99% (2
25
26   print the export_str returned from export_obj():
27   Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
28   2015-08-07 19:51:11,061 DEBUG     pytan.handler.QuestionPoller: ID 1313: id resolved to 1313
29   2015-08-07 19:51:11,061 DEBUG     pytan.handler.QuestionPoller: ID 1313: expiration resolved to 2015-
30   2015-08-07 19:51:11,061 DEBUG     pytan.handler.QuestionPoller: ID 1313: query_text resolved to Get C
31   2015-08-07 19:51:11,061 DEBUG     pytan.handler.QuestionPoller: ID 1313: id resolved to 1313
32   2015-08-07 19:51:11,061 DEBUG     pytan.handler.QuestionPoller: ID 1313: Object Info resolved to Ques
33   2015-08-07 19:51:11,066 DEBUG     pytan.handler.QuestionPoller: ID 1313: Progress: Tested: 0, Passed:
34   2015-08-07 19:51:11,066 DEBUG     pytan.handler.QuestionPoller: ID 1313: Timing: Started: 2015-08-07
35   2015-08-07 19:51:11,066 INFO      pytan.handler.QuestionPoller: ID 1313: Progress Changed 0% (0 of 2)
36   2015-08-07 19:51:16,074 DEBUG     pytan.handler.QuestionPoller: ID 1313: Progress: Tested: 0, Passed:
37   2015-08-07 19:51:16,074 DEBUG     pytan.handler.QuestionPoller: ID 1313: Timing: Started: 2015-08-07
38   2015-08-07 19:51:21,079 DEBUG     pytan.handler.QuestionPoller: ID 1313: Progress: Tested: 0, Passed:
39   2015-08-07 19:51:21,079 DEBUG     pytan.handler.QuestionPoller: ID 1313: Timing: Started: 2015-08-07
40   2015-08-07 19:51:26,083 DEBUG     pytan.handler.QuestionPoller: ID 1313: Progress: Tested: 0, Passed:
41   2015-08-07 19:51:26,083 DEBUG     pytan.handler.QuestionPoller: ID 1313: Timing: Started: 2015-08-07
42   ..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
import os
import sys
sys.dont_write_bytecode = True

# Determine our script name, script dir
my_file = os.path.abspath(sys.argv[0])
my_dir = os.path.dirname(my_file)

# determine the pytan lib dir and add it to the path
parent_dir = os.path.dirname(my_dir)
pytan_root_dir = os.path.dirname(parent_dir)
lib_dir = os.path.join(pytan_root_dir, 'lib')
path_adds = [lib_dir]

for aa in path_adds:
    if aa not in sys.path:
        sys.path.append(aa)


# connection info for Tanium Server
USERNAME = "Tanium User"
PASSWORD = "T@n!um"
HOST = "172.16.31.128"
PORT = "443"

# Logging conrols
LOGLEVEL = 2
DEBUGFORMAT = False

import tempfile

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',
    'sensors': [
```

```
53            "Computer Name", "IP Route Details", "IP Address",
54            'Folder Name Search with RegEx Match{dirname=Program Files,regex=.*Shared.*}',
55        ],
56   }
57   response = handler.ask(**ask_kwargs)
58
59   # export the object to a string
60   # (we could just as easily export to a file using export_to_report_file)
61   export_kwargs['obj'] = response['question_results']
62   export_str = handler.export_obj(**export_kwargs)
63
64
65   print ""
66   print "print the export_str returned from export_obj():"
67   if len(out.splitlines()) > 15:
68       out = out.splitlines()[0:15]
69       out.append('..trimmed for brevity..')
70       out = '\n'.join(out)
71
72   print out
```

Output from Python Code

```
1    Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2    2015-08-07 19:52:51,388 DEBUG     pytan.handler.QuestionPoller: ID 1316: id resolved to 1316
3    2015-08-07 19:52:51,388 DEBUG     pytan.handler.QuestionPoller: ID 1316: expiration resolved to 2015-
4    2015-08-07 19:52:51,388 DEBUG     pytan.handler.QuestionPoller: ID 1316: query_text resolved to Get C
5    2015-08-07 19:52:51,388 DEBUG     pytan.handler.QuestionPoller: ID 1316: id resolved to 1316
6    2015-08-07 19:52:51,388 DEBUG     pytan.handler.QuestionPoller: ID 1316: Object Info resolved to Ques
7    2015-08-07 19:52:51,392 DEBUG     pytan.handler.QuestionPoller: ID 1316: Progress: Tested: 0, Passed:
8    2015-08-07 19:52:51,392 DEBUG     pytan.handler.QuestionPoller: ID 1316: Timing: Started: 2015-08-07
9    2015-08-07 19:52:51,392 INFO      pytan.handler.QuestionPoller: ID 1316: Progress Changed 0% (0 of 2)
10   2015-08-07 19:52:56,396 DEBUG     pytan.handler.QuestionPoller: ID 1316: Progress: Tested: 0, Passed:
11   2015-08-07 19:52:56,396 DEBUG     pytan.handler.QuestionPoller: ID 1316: Timing: Started: 2015-08-07
12   2015-08-07 19:53:01,404 DEBUG     pytan.handler.QuestionPoller: ID 1316: Progress: Tested: 0, Passed:
13   2015-08-07 19:53:01,404 DEBUG     pytan.handler.QuestionPoller: ID 1316: Timing: Started: 2015-08-07
14   2015-08-07 19:53:06,413 DEBUG     pytan.handler.QuestionPoller: ID 1316: Progress: Tested: 1, Passed:
15   2015-08-07 19:53:06,413 DEBUG     pytan.handler.QuestionPoller: ID 1316: Timing: Started: 2015-08-07
16   2015-08-07 19:53:06,413 INFO      pytan.handler.QuestionPoller: ID 1316: Progress Changed 50% (1 of 2
17   2015-08-07 19:53:11,418 DEBUG     pytan.handler.QuestionPoller: ID 1316: Progress: Tested: 2, Passed:
18   2015-08-07 19:53:11,419 DEBUG     pytan.handler.QuestionPoller: ID 1316: Timing: Started: 2015-08-07
19   2015-08-07 19:53:11,419 INFO      pytan.handler.QuestionPoller: ID 1316: Progress Changed 100% (2 of
20   2015-08-07 19:53:11,419 INFO      pytan.handler.QuestionPoller: ID 1316: Reached Threshold of 99% (2
21
22   print the export_str returned from export_obj():
23   Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
24   2015-08-07 19:52:21,243 DEBUG     pytan.handler.QuestionPoller: ID 1315: id resolved to 1315
25   2015-08-07 19:52:21,243 DEBUG     pytan.handler.QuestionPoller: ID 1315: expiration resolved to 2015-
26   2015-08-07 19:52:21,243 DEBUG     pytan.handler.QuestionPoller: ID 1315: query_text resolved to Get C
27   2015-08-07 19:52:21,243 DEBUG     pytan.handler.QuestionPoller: ID 1315: id resolved to 1315
28   2015-08-07 19:52:21,243 DEBUG     pytan.handler.QuestionPoller: ID 1315: Object Info resolved to Ques
29   2015-08-07 19:52:21,246 DEBUG     pytan.handler.QuestionPoller: ID 1315: Progress: Tested: 0, Passed:
30   2015-08-07 19:52:21,246 DEBUG     pytan.handler.QuestionPoller: ID 1315: Timing: Started: 2015-08-07
31   2015-08-07 19:52:21,246 INFO      pytan.handler.QuestionPoller: ID 1315: Progress Changed 0% (0 of 2)
32   2015-08-07 19:52:26,251 DEBUG     pytan.handler.QuestionPoller: ID 1315: Progress: Tested: 0, Passed:
33   2015-08-07 19:52:26,251 DEBUG     pytan.handler.QuestionPoller: ID 1315: Timing: Started: 2015-08-07
34   2015-08-07 19:52:31,255 DEBUG     pytan.handler.QuestionPoller: ID 1315: Progress: Tested: 0, Passed:
```

```
35  2015-08-07 19:52:31,255 DEBUG      pytan.handler.QuestionPoller: ID 1315: Timing: Started: 2015-08-07
36  2015-08-07 19:52:36,264 DEBUG      pytan.handler.QuestionPoller: ID 1315: Progress: Tested: 1, Passed:
37  2015-08-07 19:52:36,264 DEBUG      pytan.handler.QuestionPoller: ID 1315: Timing: Started: 2015-08-07
38  ..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
1   import os
2   import sys
3   sys.dont_write_bytecode = True
4
5   # Determine our script name, script dir
6   my_file = os.path.abspath(sys.argv[0])
7   my_dir = os.path.dirname(my_file)
8
9   # determine the pytan lib dir and add it to the path
10  parent_dir = os.path.dirname(my_dir)
11  pytan_root_dir = os.path.dirname(parent_dir)
12  lib_dir = os.path.join(pytan_root_dir, 'lib')
13  path_adds = [lib_dir]
14
15  for aa in path_adds:
16      if aa not in sys.path:
17          sys.path.append(aa)
18
19
20  # connection info for Tanium Server
21  USERNAME = "Tanium User"
22  PASSWORD = "T@n!um"
23  HOST = "172.16.31.128"
24  PORT = "443"
25
26  # Logging conrols
27  LOGLEVEL = 2
28  DEBUGFORMAT = False
29
30  import tempfile
31
32  import pytan
33  handler = pytan.Handler(
34      username=USERNAME,
35      password=PASSWORD,
36      host=HOST,
37      port=PORT,
38      loglevel=LOGLEVEL,
39      debugformat=DEBUGFORMAT,
40  )
41
42  print handler
43
44  # setup the export_obj kwargs for later
```

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

Output from Python Code

```
1   Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2   2015-08-07 19:53:11,519 DEBUG     pytan.handler.QuestionPoller: ID 1318: id resolved to 1318
3   2015-08-07 19:53:11,519 DEBUG     pytan.handler.QuestionPoller: ID 1318: expiration resolved to 2015-
4   2015-08-07 19:53:11,519 DEBUG     pytan.handler.QuestionPoller: ID 1318: query_text resolved to Get C
5   2015-08-07 19:53:11,519 DEBUG     pytan.handler.QuestionPoller: ID 1318: id resolved to 1318
6   2015-08-07 19:53:11,519 DEBUG     pytan.handler.QuestionPoller: ID 1318: Object Info resolved to Ques
7   2015-08-07 19:53:11,522 DEBUG     pytan.handler.QuestionPoller: ID 1318: Progress: Tested: 0, Passed:
8   2015-08-07 19:53:11,522 DEBUG     pytan.handler.QuestionPoller: ID 1318: Timing: Started: 2015-08-07
9   2015-08-07 19:53:11,522 INFO      pytan.handler.QuestionPoller: ID 1318: Progress Changed 0% (0 of 2)
10  2015-08-07 19:53:16,530 DEBUG     pytan.handler.QuestionPoller: ID 1318: Progress: Tested: 0, Passed:
11  2015-08-07 19:53:16,530 DEBUG     pytan.handler.QuestionPoller: ID 1318: Timing: Started: 2015-08-07
12  2015-08-07 19:53:21,538 DEBUG     pytan.handler.QuestionPoller: ID 1318: Progress: Tested: 0, Passed:
13  2015-08-07 19:53:21,538 DEBUG     pytan.handler.QuestionPoller: ID 1318: Timing: Started: 2015-08-07
14  2015-08-07 19:53:26,543 DEBUG     pytan.handler.QuestionPoller: ID 1318: Progress: Tested: 0, Passed:
15  2015-08-07 19:53:26,543 DEBUG     pytan.handler.QuestionPoller: ID 1318: Timing: Started: 2015-08-07
16  2015-08-07 19:53:31,548 DEBUG     pytan.handler.QuestionPoller: ID 1318: Progress: Tested: 0, Passed:
17  2015-08-07 19:53:31,548 DEBUG     pytan.handler.QuestionPoller: ID 1318: Timing: Started: 2015-08-07
18  2015-08-07 19:53:36,552 DEBUG     pytan.handler.QuestionPoller: ID 1318: Progress: Tested: 0, Passed:
19  2015-08-07 19:53:36,552 DEBUG     pytan.handler.QuestionPoller: ID 1318: Timing: Started: 2015-08-07
20  2015-08-07 19:53:41,559 DEBUG     pytan.handler.QuestionPoller: ID 1318: Progress: Tested: 0, Passed:
21  2015-08-07 19:53:41,559 DEBUG     pytan.handler.QuestionPoller: ID 1318: Timing: Started: 2015-08-07
22  2015-08-07 19:53:46,566 DEBUG     pytan.handler.QuestionPoller: ID 1318: Progress: Tested: 0, Passed:
23  2015-08-07 19:53:46,566 DEBUG     pytan.handler.QuestionPoller: ID 1318: Timing: Started: 2015-08-07
24  2015-08-07 19:53:51,570 DEBUG     pytan.handler.QuestionPoller: ID 1318: Progress: Tested: 0, Passed:
25  2015-08-07 19:53:51,570 DEBUG     pytan.handler.QuestionPoller: ID 1318: Timing: Started: 2015-08-07
26  2015-08-07 19:53:56,577 DEBUG     pytan.handler.QuestionPoller: ID 1318: Progress: Tested: 0, Passed:
```

```
27  2015-08-07 19:53:56,577 DEBUG    pytan.handler.QuestionPoller: ID 1318: Timing: Started: 2015-08-07
28  2015-08-07 19:54:01,583 DEBUG    pytan.handler.QuestionPoller: ID 1318: Progress: Tested: 0, Passed:
29  2015-08-07 19:54:01,583 DEBUG    pytan.handler.QuestionPoller: ID 1318: Timing: Started: 2015-08-07
30  2015-08-07 19:54:06,590 DEBUG    pytan.handler.QuestionPoller: ID 1318: Progress: Tested: 0, Passed:
31  2015-08-07 19:54:06,591 DEBUG    pytan.handler.QuestionPoller: ID 1318: Timing: Started: 2015-08-07
32  2015-08-07 19:54:11,600 DEBUG    pytan.handler.QuestionPoller: ID 1318: Progress: Tested: 0, Passed:
33  2015-08-07 19:54:11,600 DEBUG    pytan.handler.QuestionPoller: ID 1318: Timing: Started: 2015-08-07
34  2015-08-07 19:54:16,608 DEBUG    pytan.handler.QuestionPoller: ID 1318: Progress: Tested: 0, Passed:
35  2015-08-07 19:54:16,608 DEBUG    pytan.handler.QuestionPoller: ID 1318: Timing: Started: 2015-08-07
36  2015-08-07 19:54:21,612 DEBUG    pytan.handler.QuestionPoller: ID 1318: Progress: Tested: 0, Passed:
37  2015-08-07 19:54:21,612 DEBUG    pytan.handler.QuestionPoller: ID 1318: Timing: Started: 2015-08-07
38  2015-08-07 19:54:26,621 DEBUG    pytan.handler.QuestionPoller: ID 1318: Progress: Tested: 0, Passed:
39  2015-08-07 19:54:26,621 DEBUG    pytan.handler.QuestionPoller: ID 1318: Timing: Started: 2015-08-07
40  2015-08-07 19:54:31,629 DEBUG    pytan.handler.QuestionPoller: ID 1318: Progress: Tested: 0, Passed:
41  2015-08-07 19:54:31,629 DEBUG    pytan.handler.QuestionPoller: ID 1318: Timing: Started: 2015-08-07
42  2015-08-07 19:54:36,635 DEBUG    pytan.handler.QuestionPoller: ID 1318: Progress: Tested: 2, Passed:
43  2015-08-07 19:54:36,635 DEBUG    pytan.handler.QuestionPoller: ID 1318: Timing: Started: 2015-08-07
44  2015-08-07 19:54:36,635 INFO     pytan.handler.QuestionPoller: ID 1318: Progress Changed 100% (2 of
45  2015-08-07 19:54:36,635 INFO     pytan.handler.QuestionPoller: ID 1318: Reached Threshold of 99% (2
46
47  print the export_str returned from export_obj():
48  Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
49  2015-08-07 19:52:51,388 DEBUG    pytan.handler.QuestionPoller: ID 1316: id resolved to 1316
50  2015-08-07 19:52:51,388 DEBUG    pytan.handler.QuestionPoller: ID 1316: expiration resolved to 2015-
51  2015-08-07 19:52:51,388 DEBUG    pytan.handler.QuestionPoller: ID 1316: query_text resolved to Get C
52  2015-08-07 19:52:51,388 DEBUG    pytan.handler.QuestionPoller: ID 1316: id resolved to 1316
53  2015-08-07 19:52:51,388 DEBUG    pytan.handler.QuestionPoller: ID 1316: Object Info resolved to Ques
54  2015-08-07 19:52:51,392 DEBUG    pytan.handler.QuestionPoller: ID 1316: Progress: Tested: 0, Passed:
55  2015-08-07 19:52:51,392 DEBUG    pytan.handler.QuestionPoller: ID 1316: Timing: Started: 2015-08-07
56  2015-08-07 19:52:51,392 INFO     pytan.handler.QuestionPoller: ID 1316: Progress Changed 0% (0 of 2)
57  2015-08-07 19:52:56,396 DEBUG    pytan.handler.QuestionPoller: ID 1316: Progress: Tested: 0, Passed:
58  2015-08-07 19:52:56,396 DEBUG    pytan.handler.QuestionPoller: ID 1316: Timing: Started: 2015-08-07
59  2015-08-07 19:53:01,404 DEBUG    pytan.handler.QuestionPoller: ID 1316: Progress: Tested: 0, Passed:
60  2015-08-07 19:53:01,404 DEBUG    pytan.handler.QuestionPoller: ID 1316: Timing: Started: 2015-08-07
61  2015-08-07 19:53:06,413 DEBUG    pytan.handler.QuestionPoller: ID 1316: Progress: Tested: 1, Passed:
62  2015-08-07 19:53:06,413 DEBUG    pytan.handler.QuestionPoller: ID 1316: Timing: Started: 2015-08-07
63  ..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
import os
import sys
sys.dont_write_bytecode = True

# Determine our script name, script dir
my_file = os.path.abspath(sys.argv[0])
my_dir = os.path.dirname(my_file)

# determine the pytan lib dir and add it to the path
parent_dir = os.path.dirname(my_dir)
pytan_root_dir = os.path.dirname(parent_dir)
```

```
12  lib_dir = os.path.join(pytan_root_dir, 'lib')
13  path_adds = [lib_dir]
14
15  for aa in path_adds:
16      if aa not in sys.path:
17          sys.path.append(aa)
18
19
20  # connection info for Tanium Server
21  USERNAME = "Tanium User"
22  PASSWORD = "T@n!um"
23  HOST = "172.16.31.128"
24  PORT = "443"
25
26  # Logging conrols
27  LOGLEVEL = 2
28  DEBUGFORMAT = False
29
30  import tempfile
31
32  import pytan
33  handler = pytan.Handler(
34      username=USERNAME,
35      password=PASSWORD,
36      host=HOST,
37      port=PORT,
38      loglevel=LOGLEVEL,
39      debugformat=DEBUGFORMAT,
40  )
41
42  print handler
43
44  # setup the export_obj kwargs for later
45  export_kwargs = {}
46  export_kwargs["export_format"] = u'csv'
47  export_kwargs["header_add_type"] = True
48
49  # ask the question that will provide the resultset that we want to use
50  ask_kwargs = {
51      'qtype': 'manual',
52      'sensors': [
53          "Computer Name", "IP Route Details", "IP Address",
54          'Folder Name Search with RegEx Match{dirname=Program Files,regex=.*Shared.*}',
55      ],
56  }
57  response = handler.ask(**ask_kwargs)
58
59  # export the object to a string
60  # (we could just as easily export to a file using export_to_report_file)
61  export_kwargs['obj'] = response['question_results']
62  export_str = handler.export_obj(**export_kwargs)
63
64
65  print ""
66  print "print the export_str returned from export_obj():"
67  if len(out.splitlines()) > 15:
68      out = out.splitlines()[0:15]
69      out.append('..trimmed for brevity..')
```

```
70      out = '\n'.join(out)
71
72  print out
```

Output from Python Code

```
1   Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2   2015-08-07 19:54:36,751 DEBUG    pytan.handler.QuestionPoller: ID 1319: id resolved to 1319
3   2015-08-07 19:54:36,751 DEBUG    pytan.handler.QuestionPoller: ID 1319: expiration resolved to 2015-
4   2015-08-07 19:54:36,751 DEBUG    pytan.handler.QuestionPoller: ID 1319: query_text resolved to Get C
5   2015-08-07 19:54:36,751 DEBUG    pytan.handler.QuestionPoller: ID 1319: id resolved to 1319
6   2015-08-07 19:54:36,751 DEBUG    pytan.handler.QuestionPoller: ID 1319: Object Info resolved to Ques
7   2015-08-07 19:54:36,754 DEBUG    pytan.handler.QuestionPoller: ID 1319: Progress: Tested: 0, Passed:
8   2015-08-07 19:54:36,754 DEBUG    pytan.handler.QuestionPoller: ID 1319: Timing: Started: 2015-08-07
9   2015-08-07 19:54:36,754 INFO     pytan.handler.QuestionPoller: ID 1319: Progress Changed 0% (0 of 2)
10  2015-08-07 19:54:41,761 DEBUG    pytan.handler.QuestionPoller: ID 1319: Progress: Tested: 0, Passed:
11  2015-08-07 19:54:41,761 DEBUG    pytan.handler.QuestionPoller: ID 1319: Timing: Started: 2015-08-07
12  2015-08-07 19:54:46,767 DEBUG    pytan.handler.QuestionPoller: ID 1319: Progress: Tested: 0, Passed:
13  2015-08-07 19:54:46,767 DEBUG    pytan.handler.QuestionPoller: ID 1319: Timing: Started: 2015-08-07
14  2015-08-07 19:54:51,771 DEBUG    pytan.handler.QuestionPoller: ID 1319: Progress: Tested: 0, Passed:
15  2015-08-07 19:54:51,771 DEBUG    pytan.handler.QuestionPoller: ID 1319: Timing: Started: 2015-08-07
16  2015-08-07 19:54:56,775 DEBUG    pytan.handler.QuestionPoller: ID 1319: Progress: Tested: 2, Passed:
17  2015-08-07 19:54:56,775 DEBUG    pytan.handler.QuestionPoller: ID 1319: Timing: Started: 2015-08-07
18  2015-08-07 19:54:56,775 INFO     pytan.handler.QuestionPoller: ID 1319: Progress Changed 100% (2 of
19  2015-08-07 19:54:56,775 INFO     pytan.handler.QuestionPoller: ID 1319: Reached Threshold of 99% (2
20
21  print the export_str returned from export_obj():
22  Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
23  2015-08-07 19:53:11,519 DEBUG    pytan.handler.QuestionPoller: ID 1318: id resolved to 1318
24  2015-08-07 19:53:11,519 DEBUG    pytan.handler.QuestionPoller: ID 1318: expiration resolved to 2015-
25  2015-08-07 19:53:11,519 DEBUG    pytan.handler.QuestionPoller: ID 1318: query_text resolved to Get C
26  2015-08-07 19:53:11,519 DEBUG    pytan.handler.QuestionPoller: ID 1318: id resolved to 1318
27  2015-08-07 19:53:11,519 DEBUG    pytan.handler.QuestionPoller: ID 1318: Object Info resolved to Ques
28  2015-08-07 19:53:11,522 DEBUG    pytan.handler.QuestionPoller: ID 1318: Progress: Tested: 0, Passed:
29  2015-08-07 19:53:11,522 DEBUG    pytan.handler.QuestionPoller: ID 1318: Timing: Started: 2015-08-07
30  2015-08-07 19:53:11,522 INFO     pytan.handler.QuestionPoller: ID 1318: Progress Changed 0% (0 of 2)
31  2015-08-07 19:53:16,530 DEBUG    pytan.handler.QuestionPoller: ID 1318: Progress: Tested: 0, Passed:
32  2015-08-07 19:53:16,530 DEBUG    pytan.handler.QuestionPoller: ID 1318: Timing: Started: 2015-08-07
33  2015-08-07 19:53:21,538 DEBUG    pytan.handler.QuestionPoller: ID 1318: Progress: Tested: 0, Passed:
34  2015-08-07 19:53:21,538 DEBUG    pytan.handler.QuestionPoller: ID 1318: Timing: Started: 2015-08-07
35  2015-08-07 19:53:26,543 DEBUG    pytan.handler.QuestionPoller: ID 1318: Progress: Tested: 0, Passed:
36  2015-08-07 19:53:26,543 DEBUG    pytan.handler.QuestionPoller: ID 1318: Timing: Started: 2015-08-07
37  ..trimmed for brevity..
```

Export resultset csv sensor false

Export a ResultSet from asking a question as CSV with false for header_add_sensor

Example Python Code

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

```
5    # Determine our script name, script dir
6    my_file = os.path.abspath(sys.argv[0])
7    my_dir = os.path.dirname(my_file)
8
9    # determine the pytan lib dir and add it to the path
10   parent_dir = os.path.dirname(my_dir)
11   pytan_root_dir = os.path.dirname(parent_dir)
12   lib_dir = os.path.join(pytan_root_dir, 'lib')
13   path_adds = [lib_dir]
14
15   for aa in path_adds:
16       if aa not in sys.path:
17           sys.path.append(aa)
18
19
20   # connection info for Tanium Server
21   USERNAME = "Tanium User"
22   PASSWORD = "T@n!um"
23   HOST = "172.16.31.128"
24   PORT = "443"
25
26   # Logging conrols
27   LOGLEVEL = 2
28   DEBUGFORMAT = False
29
30   import tempfile
31
32   import pytan
33   handler = pytan.Handler(
34       username=USERNAME,
35       password=PASSWORD,
36       host=HOST,
37       port=PORT,
38       loglevel=LOGLEVEL,
39       debugformat=DEBUGFORMAT,
40   )
41
42   print handler
43
44   # setup the export_obj kwargs for later
45   export_kwargs = {}
46   export_kwargs["export_format"] = u'csv'
47   export_kwargs["header_add_sensor"] = False
48
49   # ask the question that will provide the resultset that we want to use
50   ask_kwargs = {
51       'qtype': 'manual',
52       'sensors': [
53           "Computer Name", "IP Route Details", "IP Address",
54           'Folder Name Search with RegEx Match{dirname=Program Files,regex=.*Shared.*}',
55       ],
56   }
57   response = handler.ask(**ask_kwargs)
58
59   # export the object to a string
60   # (we could just as easily export to a file using export_to_report_file)
61   export_kwargs['obj'] = response['question_results']
62   export_str = handler.export_obj(**export_kwargs)
```

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

Output from Python Code

```
1   Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2   2015-08-07 19:54:56,919 DEBUG     pytan.handler.QuestionPoller: ID 1320: id resolved to 1320
3   2015-08-07 19:54:56,919 DEBUG     pytan.handler.QuestionPoller: ID 1320: expiration resolved to 2015-
4   2015-08-07 19:54:56,919 DEBUG     pytan.handler.QuestionPoller: ID 1320: query_text resolved to Get C
5   2015-08-07 19:54:56,919 DEBUG     pytan.handler.QuestionPoller: ID 1320: id resolved to 1320
6   2015-08-07 19:54:56,919 DEBUG     pytan.handler.QuestionPoller: ID 1320: Object Info resolved to Ques
7   2015-08-07 19:54:56,923 DEBUG     pytan.handler.QuestionPoller: ID 1320: Progress: Tested: 0, Passed:
8   2015-08-07 19:54:56,923 DEBUG     pytan.handler.QuestionPoller: ID 1320: Timing: Started: 2015-08-07
9   2015-08-07 19:54:56,923 INFO      pytan.handler.QuestionPoller: ID 1320: Progress Changed 0% (0 of 2)
10  2015-08-07 19:55:01,930 DEBUG     pytan.handler.QuestionPoller: ID 1320: Progress: Tested: 0, Passed:
11  2015-08-07 19:55:01,930 DEBUG     pytan.handler.QuestionPoller: ID 1320: Timing: Started: 2015-08-07
12  2015-08-07 19:55:06,936 DEBUG     pytan.handler.QuestionPoller: ID 1320: Progress: Tested: 0, Passed:
13  2015-08-07 19:55:06,936 DEBUG     pytan.handler.QuestionPoller: ID 1320: Timing: Started: 2015-08-07
14  2015-08-07 19:55:11,944 DEBUG     pytan.handler.QuestionPoller: ID 1320: Progress: Tested: 0, Passed:
15  2015-08-07 19:55:11,944 DEBUG     pytan.handler.QuestionPoller: ID 1320: Timing: Started: 2015-08-07
16  2015-08-07 19:55:16,949 DEBUG     pytan.handler.QuestionPoller: ID 1320: Progress: Tested: 0, Passed:
17  2015-08-07 19:55:16,950 DEBUG     pytan.handler.QuestionPoller: ID 1320: Timing: Started: 2015-08-07
18  2015-08-07 19:55:21,957 DEBUG     pytan.handler.QuestionPoller: ID 1320: Progress: Tested: 1, Passed:
19  2015-08-07 19:55:21,957 DEBUG     pytan.handler.QuestionPoller: ID 1320: Timing: Started: 2015-08-07
20  2015-08-07 19:55:21,957 INFO      pytan.handler.QuestionPoller: ID 1320: Progress Changed 50% (1 of 2
21  2015-08-07 19:55:26,962 DEBUG     pytan.handler.QuestionPoller: ID 1320: Progress: Tested: 2, Passed:
22  2015-08-07 19:55:26,962 DEBUG     pytan.handler.QuestionPoller: ID 1320: Timing: Started: 2015-08-07
23  2015-08-07 19:55:26,962 INFO      pytan.handler.QuestionPoller: ID 1320: Progress Changed 100% (2 of
24  2015-08-07 19:55:26,962 INFO      pytan.handler.QuestionPoller: ID 1320: Reached Threshold of 99% (2
25
26  print the export_str returned from export_obj():
27  Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
28  2015-08-07 19:54:36,751 DEBUG     pytan.handler.QuestionPoller: ID 1319: id resolved to 1319
29  2015-08-07 19:54:36,751 DEBUG     pytan.handler.QuestionPoller: ID 1319: expiration resolved to 2015-
30  2015-08-07 19:54:36,751 DEBUG     pytan.handler.QuestionPoller: ID 1319: query_text resolved to Get C
31  2015-08-07 19:54:36,751 DEBUG     pytan.handler.QuestionPoller: ID 1319: id resolved to 1319
32  2015-08-07 19:54:36,751 DEBUG     pytan.handler.QuestionPoller: ID 1319: Object Info resolved to Ques
33  2015-08-07 19:54:36,754 DEBUG     pytan.handler.QuestionPoller: ID 1319: Progress: Tested: 0, Passed:
34  2015-08-07 19:54:36,754 DEBUG     pytan.handler.QuestionPoller: ID 1319: Timing: Started: 2015-08-07
35  2015-08-07 19:54:36,754 INFO      pytan.handler.QuestionPoller: ID 1319: Progress Changed 0% (0 of 2)
36  2015-08-07 19:54:41,761 DEBUG     pytan.handler.QuestionPoller: ID 1319: Progress: Tested: 0, Passed:
37  2015-08-07 19:54:41,761 DEBUG     pytan.handler.QuestionPoller: ID 1319: Timing: Started: 2015-08-07
38  2015-08-07 19:54:46,767 DEBUG     pytan.handler.QuestionPoller: ID 1319: Progress: Tested: 0, Passed:
39  2015-08-07 19:54:46,767 DEBUG     pytan.handler.QuestionPoller: ID 1319: Timing: Started: 2015-08-07
40  2015-08-07 19:54:51,771 DEBUG     pytan.handler.QuestionPoller: ID 1319: Progress: Tested: 0, Passed:
41  2015-08-07 19:54:51,771 DEBUG     pytan.handler.QuestionPoller: ID 1319: Timing: Started: 2015-08-07
42  ..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
import os
import sys
sys.dont_write_bytecode = True

# Determine our script name, script dir
my_file = os.path.abspath(sys.argv[0])
my_dir = os.path.dirname(my_file)

# determine the pytan lib dir and add it to the path
parent_dir = os.path.dirname(my_dir)
pytan_root_dir = os.path.dirname(parent_dir)
lib_dir = os.path.join(pytan_root_dir, 'lib')
path_adds = [lib_dir]

for aa in path_adds:
    if aa not in sys.path:
        sys.path.append(aa)


# connection info for Tanium Server
USERNAME = "Tanium User"
PASSWORD = "T@n!um"
HOST = "172.16.31.128"
PORT = "443"

# Logging conrols
LOGLEVEL = 2
DEBUGFORMAT = False

import tempfile

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',
    'sensors': [
```

```
53          "Computer Name", "IP Route Details", "IP Address",
54          'Folder Name Search with RegEx Match{dirname=Program Files,regex=.*Shared.*}',
55      ],
56  }
57  response = handler.ask(**ask_kwargs)
58
59  # export the object to a string
60  # (we could just as easily export to a file using export_to_report_file)
61  export_kwargs['obj'] = response['question_results']
62  export_str = handler.export_obj(**export_kwargs)
63
64
65  print ""
66  print "print the export_str returned from export_obj():"
67  if len(out.splitlines()) > 15:
68      out = out.splitlines()[0:15]
69      out.append('..trimmed for brevity..')
70      out = '\n'.join(out)
71
72  print out
```

Output from Python Code

```
1   Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2   2015-08-07 19:55:27,070 DEBUG     pytan.handler.QuestionPoller: ID 1321: id resolved to 1321
3   2015-08-07 19:55:27,070 DEBUG     pytan.handler.QuestionPoller: ID 1321: expiration resolved to 2015-
4   2015-08-07 19:55:27,070 DEBUG     pytan.handler.QuestionPoller: ID 1321: query_text resolved to Get C
5   2015-08-07 19:55:27,070 DEBUG     pytan.handler.QuestionPoller: ID 1321: id resolved to 1321
6   2015-08-07 19:55:27,070 DEBUG     pytan.handler.QuestionPoller: ID 1321: Object Info resolved to Ques
7   2015-08-07 19:55:27,074 DEBUG     pytan.handler.QuestionPoller: ID 1321: Progress: Tested: 0, Passed:
8   2015-08-07 19:55:27,074 DEBUG     pytan.handler.QuestionPoller: ID 1321: Timing: Started: 2015-08-07
9   2015-08-07 19:55:27,074 INFO      pytan.handler.QuestionPoller: ID 1321: Progress Changed 0% (0 of 2)
10  2015-08-07 19:55:32,077 DEBUG     pytan.handler.QuestionPoller: ID 1321: Progress: Tested: 0, Passed:
11  2015-08-07 19:55:32,077 DEBUG     pytan.handler.QuestionPoller: ID 1321: Timing: Started: 2015-08-07
12  2015-08-07 19:55:37,083 DEBUG     pytan.handler.QuestionPoller: ID 1321: Progress: Tested: 0, Passed:
13  2015-08-07 19:55:37,083 DEBUG     pytan.handler.QuestionPoller: ID 1321: Timing: Started: 2015-08-07
14  2015-08-07 19:55:42,089 DEBUG     pytan.handler.QuestionPoller: ID 1321: Progress: Tested: 0, Passed:
15  2015-08-07 19:55:42,089 DEBUG     pytan.handler.QuestionPoller: ID 1321: Timing: Started: 2015-08-07
16  2015-08-07 19:55:47,097 DEBUG     pytan.handler.QuestionPoller: ID 1321: Progress: Tested: 0, Passed:
17  2015-08-07 19:55:47,097 DEBUG     pytan.handler.QuestionPoller: ID 1321: Timing: Started: 2015-08-07
18  2015-08-07 19:55:52,105 DEBUG     pytan.handler.QuestionPoller: ID 1321: Progress: Tested: 0, Passed:
19  2015-08-07 19:55:52,105 DEBUG     pytan.handler.QuestionPoller: ID 1321: Timing: Started: 2015-08-07
20  2015-08-07 19:55:57,112 DEBUG     pytan.handler.QuestionPoller: ID 1321: Progress: Tested: 0, Passed:
21  2015-08-07 19:55:57,112 DEBUG     pytan.handler.QuestionPoller: ID 1321: Timing: Started: 2015-08-07
22  2015-08-07 19:56:02,119 DEBUG     pytan.handler.QuestionPoller: ID 1321: Progress: Tested: 0, Passed:
23  2015-08-07 19:56:02,119 DEBUG     pytan.handler.QuestionPoller: ID 1321: Timing: Started: 2015-08-07
24  2015-08-07 19:56:07,129 DEBUG     pytan.handler.QuestionPoller: ID 1321: Progress: Tested: 0, Passed:
25  2015-08-07 19:56:07,129 DEBUG     pytan.handler.QuestionPoller: ID 1321: Timing: Started: 2015-08-07
26  2015-08-07 19:56:12,138 DEBUG     pytan.handler.QuestionPoller: ID 1321: Progress: Tested: 0, Passed:
27  2015-08-07 19:56:12,138 DEBUG     pytan.handler.QuestionPoller: ID 1321: Timing: Started: 2015-08-07
28  2015-08-07 19:56:17,145 DEBUG     pytan.handler.QuestionPoller: ID 1321: Progress: Tested: 0, Passed:
29  2015-08-07 19:56:17,145 DEBUG     pytan.handler.QuestionPoller: ID 1321: Timing: Started: 2015-08-07
30  2015-08-07 19:56:22,152 DEBUG     pytan.handler.QuestionPoller: ID 1321: Progress: Tested: 0, Passed:
31  2015-08-07 19:56:22,152 DEBUG     pytan.handler.QuestionPoller: ID 1321: Timing: Started: 2015-08-07
32  2015-08-07 19:56:27,160 DEBUG     pytan.handler.QuestionPoller: ID 1321: Progress: Tested: 0, Passed:
33  2015-08-07 19:56:27,160 DEBUG     pytan.handler.QuestionPoller: ID 1321: Timing: Started: 2015-08-07
34  2015-08-07 19:56:32,167 DEBUG     pytan.handler.QuestionPoller: ID 1321: Progress: Tested: 2, Passed:
```

```
35   2015-08-07 19:56:32,167 DEBUG    pytan.handler.QuestionPoller: ID 1321: Timing: Started: 2015-08-07
36   2015-08-07 19:56:32,167 INFO     pytan.handler.QuestionPoller: ID 1321: Progress Changed 100% (2 of
37   2015-08-07 19:56:32,167 INFO     pytan.handler.QuestionPoller: ID 1321: Reached Threshold of 99% (2
38
39   print the export_str returned from export_obj():
40   Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
41   2015-08-07 19:54:56,919 DEBUG    pytan.handler.QuestionPoller: ID 1320: id resolved to 1320
42   2015-08-07 19:54:56,919 DEBUG    pytan.handler.QuestionPoller: ID 1320: expiration resolved to 2015-
43   2015-08-07 19:54:56,919 DEBUG    pytan.handler.QuestionPoller: ID 1320: query_text resolved to Get C
44   2015-08-07 19:54:56,919 DEBUG    pytan.handler.QuestionPoller: ID 1320: id resolved to 1320
45   2015-08-07 19:54:56,919 DEBUG    pytan.handler.QuestionPoller: ID 1320: Object Info resolved to Ques
46   2015-08-07 19:54:56,923 DEBUG    pytan.handler.QuestionPoller: ID 1320: Progress: Tested: 0, Passed:
47   2015-08-07 19:54:56,923 DEBUG    pytan.handler.QuestionPoller: ID 1320: Timing: Started: 2015-08-07
48   2015-08-07 19:54:56,923 INFO     pytan.handler.QuestionPoller: ID 1320: Progress Changed 0% (0 of 2)
49   2015-08-07 19:55:01,930 DEBUG    pytan.handler.QuestionPoller: ID 1320: Progress: Tested: 0, Passed:
50   2015-08-07 19:55:01,930 DEBUG    pytan.handler.QuestionPoller: ID 1320: Timing: Started: 2015-08-07
51   2015-08-07 19:55:06,936 DEBUG    pytan.handler.QuestionPoller: ID 1320: Progress: Tested: 0, Passed:
52   2015-08-07 19:55:06,936 DEBUG    pytan.handler.QuestionPoller: ID 1320: Timing: Started: 2015-08-07
53   2015-08-07 19:55:11,944 DEBUG    pytan.handler.QuestionPoller: ID 1320: Progress: Tested: 0, Passed:
54   2015-08-07 19:55:11,944 DEBUG    pytan.handler.QuestionPoller: ID 1320: Timing: Started: 2015-08-07
55   ..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
1    import os
2    import sys
3    sys.dont_write_bytecode = True
4
5    # Determine our script name, script dir
6    my_file = os.path.abspath(sys.argv[0])
7    my_dir = os.path.dirname(my_file)
8
9    # determine the pytan lib dir and add it to the path
10   parent_dir = os.path.dirname(my_dir)
11   pytan_root_dir = os.path.dirname(parent_dir)
12   lib_dir = os.path.join(pytan_root_dir, 'lib')
13   path_adds = [lib_dir]
14
15   for aa in path_adds:
16       if aa not in sys.path:
17           sys.path.append(aa)
18
19
20   # connection info for Tanium Server
21   USERNAME = "Tanium User"
22   PASSWORD = "T@n!um"
23   HOST = "172.16.31.128"
24   PORT = "443"
25
```

```
26   # Logging conrols
27   LOGLEVEL = 2
28   DEBUGFORMAT = False
29
30   import tempfile
31
32   import pytan
33   handler = pytan.Handler(
34       username=USERNAME,
35       password=PASSWORD,
36       host=HOST,
37       port=PORT,
38       loglevel=LOGLEVEL,
39       debugformat=DEBUGFORMAT,
40   )
41
42   print handler
43
44   # setup the export_obj kwargs for later
45   export_kwargs = {}
46   export_kwargs["export_format"] = u'csv'
47   export_kwargs["header_sort"] = [[]]
48
49   # ask the question that will provide the resultset that we want to use
50   ask_kwargs = {
51       'qtype': 'manual',
52       'sensors': [
53           "Computer Name"
54       ],
55   }
56   response = handler.ask(**ask_kwargs)
57   export_kwargs['obj'] = response['question_results']
58
59   # export the object to a string
60   # this should throw an exception: pytan.exceptions.HandlerError
61   import traceback
62
63   try:
64       handler.export_obj(**export_kwargs)
65   except Exception as e:
66       traceback.print_exc(file=sys.stdout)
```

Output from Python Code

```
1    Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2    2015-08-07 19:56:32,279 DEBUG     pytan.handler.QuestionPoller: ID 1323: id resolved to 1323
3    2015-08-07 19:56:32,279 DEBUG     pytan.handler.QuestionPoller: ID 1323: expiration resolved to 2015-
4    2015-08-07 19:56:32,279 DEBUG     pytan.handler.QuestionPoller: ID 1323: query_text resolved to Get C
5    2015-08-07 19:56:32,279 DEBUG     pytan.handler.QuestionPoller: ID 1323: id resolved to 1323
6    2015-08-07 19:56:32,279 DEBUG     pytan.handler.QuestionPoller: ID 1323: Object Info resolved to Ques
7    2015-08-07 19:56:32,282 DEBUG     pytan.handler.QuestionPoller: ID 1323: Progress: Tested: 0, Passed:
8    2015-08-07 19:56:32,282 DEBUG     pytan.handler.QuestionPoller: ID 1323: Timing: Started: 2015-08-07
9    2015-08-07 19:56:32,282 INFO      pytan.handler.QuestionPoller: ID 1323: Progress Changed 0% (0 of 2)
10   2015-08-07 19:56:37,290 DEBUG     pytan.handler.QuestionPoller: ID 1323: Progress: Tested: 2, Passed:
11   2015-08-07 19:56:37,290 DEBUG     pytan.handler.QuestionPoller: ID 1323: Timing: Started: 2015-08-07
12   2015-08-07 19:56:37,290 INFO      pytan.handler.QuestionPoller: ID 1323: Progress Changed 100% (2 of
13   2015-08-07 19:56:37,290 INFO      pytan.handler.QuestionPoller: ID 1323: Reached Threshold of 99% (2
```

```
14   Traceback (most recent call last):
15     File "<string>", line 65, in <module>
16     File "/Users/jolsen/gh/pytan/lib/pytan/utils.py", line 2699, in wrap
17       ret = f(*args, **kwargs)
18     File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 1084, in export_obj
19       pytan.utils.check_dictkey(**check_args)
20     File "/Users/jolsen/gh/pytan/lib/pytan/utils.py", line 2692, in check_dictkey
21       raise pytan.exceptions.HandlerError(err(key, valid_list_types, list_types))
22   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

```python
1   import os
2   import sys
3   sys.dont_write_bytecode = True
4
5   # Determine our script name, script dir
6   my_file = os.path.abspath(sys.argv[0])
7   my_dir = os.path.dirname(my_file)
8
9   # determine the pytan lib dir and add it to the path
10  parent_dir = os.path.dirname(my_dir)
11  pytan_root_dir = os.path.dirname(parent_dir)
12  lib_dir = os.path.join(pytan_root_dir, 'lib')
13  path_adds = [lib_dir]
14
15  for aa in path_adds:
16      if aa not in sys.path:
17          sys.path.append(aa)
18
19
20  # connection info for Tanium Server
21  USERNAME = "Tanium User"
22  PASSWORD = "T@n!um"
23  HOST = "172.16.31.128"
24  PORT = "443"
25
26  # Logging conrols
27  LOGLEVEL = 2
28  DEBUGFORMAT = False
29
30  import tempfile
31
32  import pytan
33  handler = pytan.Handler(
34      username=USERNAME,
35      password=PASSWORD,
36      host=HOST,
37      port=PORT,
38      loglevel=LOGLEVEL,
39      debugformat=DEBUGFORMAT,
```

```
40  )
41
42  print handler
43
44  # setup the export_obj kwargs for later
45  export_kwargs = {}
46  export_kwargs["export_format"] = u'csv'
47  export_kwargs["header_sort"] = u'bad'
48
49  # ask the question that will provide the resultset that we want to use
50  ask_kwargs = {
51      'qtype': 'manual',
52      'sensors': [
53          "Computer Name"
54      ],
55  }
56  response = handler.ask(**ask_kwargs)
57  export_kwargs['obj'] = response['question_results']
58
59  # export the object to a string
60  # this should throw an exception: pytan.exceptions.HandlerError
61  import traceback
62
63  try:
64      handler.export_obj(**export_kwargs)
65  except Exception as e:
66      traceback.print_exc(file=sys.stdout)
```

Output from Python Code

```
1   Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2   2015-08-07 19:56:37,342 DEBUG      pytan.handler.QuestionPoller: ID 1324: id resolved to 1324
3   2015-08-07 19:56:37,342 DEBUG      pytan.handler.QuestionPoller: ID 1324: expiration resolved to 2015-
4   2015-08-07 19:56:37,342 DEBUG      pytan.handler.QuestionPoller: ID 1324: query_text resolved to Get C
5   2015-08-07 19:56:37,342 DEBUG      pytan.handler.QuestionPoller: ID 1324: id resolved to 1324
6   2015-08-07 19:56:37,342 DEBUG      pytan.handler.QuestionPoller: ID 1324: Object Info resolved to Ques
7   2015-08-07 19:56:37,345 DEBUG      pytan.handler.QuestionPoller: ID 1324: Progress: Tested: 0, Passed:
8   2015-08-07 19:56:37,345 DEBUG      pytan.handler.QuestionPoller: ID 1324: Timing: Started: 2015-08-07
9   2015-08-07 19:56:37,345 INFO       pytan.handler.QuestionPoller: ID 1324: Progress Changed 0% (0 of 2)
10  2015-08-07 19:56:42,353 DEBUG      pytan.handler.QuestionPoller: ID 1324: Progress: Tested: 0, Passed:
11  2015-08-07 19:56:42,353 DEBUG      pytan.handler.QuestionPoller: ID 1324: Timing: Started: 2015-08-07
12  2015-08-07 19:56:47,361 DEBUG      pytan.handler.QuestionPoller: ID 1324: Progress: Tested: 1, Passed:
13  2015-08-07 19:56:47,361 DEBUG      pytan.handler.QuestionPoller: ID 1324: Timing: Started: 2015-08-07
14  2015-08-07 19:56:47,361 INFO       pytan.handler.QuestionPoller: ID 1324: Progress Changed 50% (1 of 2
15  2015-08-07 19:56:52,368 DEBUG      pytan.handler.QuestionPoller: ID 1324: Progress: Tested: 2, Passed:
16  2015-08-07 19:56:52,368 DEBUG      pytan.handler.QuestionPoller: ID 1324: Timing: Started: 2015-08-07
17  2015-08-07 19:56:52,368 INFO       pytan.handler.QuestionPoller: ID 1324: Progress Changed 100% (2 of
18  2015-08-07 19:56:52,368 INFO       pytan.handler.QuestionPoller: ID 1324: Reached Threshold of 99% (2
19  Traceback (most recent call last):
20    File "<string>", line 65, in <module>
21    File "/Users/jolsen/gh/pytan/lib/pytan/utils.py", line 2699, in wrap
22      ret = f(*args, **kwargs)
23    File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 1084, in export_obj
24      pytan.utils.check_dictkey(**check_args)
25    File "/Users/jolsen/gh/pytan/lib/pytan/utils.py", line 2685, in check_dictkey
26      raise pytan.exceptions.HandlerError(err(key, valid_types, k_type))
```

```
27   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
1    import os
2    import sys
3    sys.dont_write_bytecode = True
4
5    # Determine our script name, script dir
6    my_file = os.path.abspath(sys.argv[0])
7    my_dir = os.path.dirname(my_file)
8
9    # determine the pytan lib dir and add it to the path
10   parent_dir = os.path.dirname(my_dir)
11   pytan_root_dir = os.path.dirname(parent_dir)
12   lib_dir = os.path.join(pytan_root_dir, 'lib')
13   path_adds = [lib_dir]
14
15   for aa in path_adds:
16       if aa not in sys.path:
17           sys.path.append(aa)
18
19
20   # connection info for Tanium Server
21   USERNAME = "Tanium User"
22   PASSWORD = "T@n!um"
23   HOST = "172.16.31.128"
24   PORT = "443"
25
26   # Logging conrols
27   LOGLEVEL = 2
28   DEBUGFORMAT = False
29
30   import tempfile
31
32   import pytan
33   handler = pytan.Handler(
34       username=USERNAME,
35       password=PASSWORD,
36       host=HOST,
37       port=PORT,
38       loglevel=LOGLEVEL,
39       debugformat=DEBUGFORMAT,
40   )
41
42   print handler
43
44   # setup the export_obj kwargs for later
45   export_kwargs = {}
46   export_kwargs["export_format"] = u'csv'
47   export_kwargs["expand_grouped_columns"] = u'bad'
```

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

Output from Python Code

```
1   Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2   2015-08-07 19:56:52,416 DEBUG      pytan.handler.QuestionPoller: ID 1325: id resolved to 1325
3   2015-08-07 19:56:52,416 DEBUG      pytan.handler.QuestionPoller: ID 1325: expiration resolved to 2015-
4   2015-08-07 19:56:52,416 DEBUG      pytan.handler.QuestionPoller: ID 1325: query_text resolved to Get C
5   2015-08-07 19:56:52,416 DEBUG      pytan.handler.QuestionPoller: ID 1325: id resolved to 1325
6   2015-08-07 19:56:52,416 DEBUG      pytan.handler.QuestionPoller: ID 1325: Object Info resolved to Ques
7   2015-08-07 19:56:52,420 DEBUG      pytan.handler.QuestionPoller: ID 1325: Progress: Tested: 0, Passed:
8   2015-08-07 19:56:52,420 DEBUG      pytan.handler.QuestionPoller: ID 1325: Timing: Started: 2015-08-07
9   2015-08-07 19:56:52,420 INFO       pytan.handler.QuestionPoller: ID 1325: Progress Changed 0% (0 of 2)
10  2015-08-07 19:56:57,424 DEBUG      pytan.handler.QuestionPoller: ID 1325: Progress: Tested: 1, Passed:
11  2015-08-07 19:56:57,424 DEBUG      pytan.handler.QuestionPoller: ID 1325: Timing: Started: 2015-08-07
12  2015-08-07 19:56:57,424 INFO       pytan.handler.QuestionPoller: ID 1325: Progress Changed 50% (1 of 2
13  2015-08-07 19:57:02,429 DEBUG      pytan.handler.QuestionPoller: ID 1325: Progress: Tested: 2, Passed:
14  2015-08-07 19:57:02,429 DEBUG      pytan.handler.QuestionPoller: ID 1325: Timing: Started: 2015-08-07
15  2015-08-07 19:57:02,429 INFO       pytan.handler.QuestionPoller: ID 1325: Progress Changed 100% (2 of
16  2015-08-07 19:57:02,429 INFO       pytan.handler.QuestionPoller: ID 1325: Reached Threshold of 99% (2
17  Traceback (most recent call last):
18    File "<string>", line 65, in <module>
19    File "/Users/jolsen/gh/pytan/lib/pytan/utils.py", line 2699, in wrap
20      ret = f(*args, **kwargs)
21    File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 1084, in export_obj
22      pytan.utils.check_dictkey(**check_args)
23    File "/Users/jolsen/gh/pytan/lib/pytan/utils.py", line 2685, in check_dictkey
24      raise pytan.exceptions.HandlerError(err(key, valid_types, k_type))
25  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
1   import os
2   import sys
3   sys.dont_write_bytecode = True
4
5   # Determine our script name, script dir
6   my_file = os.path.abspath(sys.argv[0])
7   my_dir = os.path.dirname(my_file)
8
9   # determine the pytan lib dir and add it to the path
10  parent_dir = os.path.dirname(my_dir)
11  pytan_root_dir = os.path.dirname(parent_dir)
12  lib_dir = os.path.join(pytan_root_dir, 'lib')
13  path_adds = [lib_dir]
14
15  for aa in path_adds:
16      if aa not in sys.path:
17          sys.path.append(aa)
18
19
20  # connection info for Tanium Server
21  USERNAME = "Tanium User"
22  PASSWORD = "T@n!um"
23  HOST = "172.16.31.128"
24  PORT = "443"
25
26  # Logging conrols
27  LOGLEVEL = 2
28  DEBUGFORMAT = False
29
30  import tempfile
31
32  import pytan
33  handler = pytan.Handler(
34      username=USERNAME,
35      password=PASSWORD,
36      host=HOST,
37      port=PORT,
38      loglevel=LOGLEVEL,
39      debugformat=DEBUGFORMAT,
40  )
41
42  print handler
43
44  # setup the export_obj kwargs for later
45  export_kwargs = {}
46  export_kwargs["export_format"] = u'csv'
47  export_kwargs["sensors"] = [[]]
48  export_kwargs["header_add_sensor"] = True
49
50  # ask the question that will provide the resultset that we want to use
51  ask_kwargs = {
52      'qtype': 'manual',
53      'sensors': [
54          "Computer Name"
55      ],
56  }
57  response = handler.ask(**ask_kwargs)
58  export_kwargs['obj'] = response['question_results']
```

```
59
60   # export the object to a string
61   # this should throw an exception: pytan.exceptions.HandlerError
62   import traceback
63
64   try:
65       handler.export_obj(**export_kwargs)
66   except Exception as e:
67       traceback.print_exc(file=sys.stdout)
```

Output from Python Code

```
1    Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2    2015-08-07 19:57:02,479 DEBUG     pytan.handler.QuestionPoller: ID 1326: id resolved to 1326
3    2015-08-07 19:57:02,479 DEBUG     pytan.handler.QuestionPoller: ID 1326: expiration resolved to 2015-
4    2015-08-07 19:57:02,479 DEBUG     pytan.handler.QuestionPoller: ID 1326: query_text resolved to Get C
5    2015-08-07 19:57:02,479 DEBUG     pytan.handler.QuestionPoller: ID 1326: id resolved to 1326
6    2015-08-07 19:57:02,479 DEBUG     pytan.handler.QuestionPoller: ID 1326: Object Info resolved to Ques
7    2015-08-07 19:57:02,482 DEBUG     pytan.handler.QuestionPoller: ID 1326: Progress: Tested: 0, Passed:
8    2015-08-07 19:57:02,483 DEBUG     pytan.handler.QuestionPoller: ID 1326: Timing: Started: 2015-08-07
9    2015-08-07 19:57:02,483 INFO      pytan.handler.QuestionPoller: ID 1326: Progress Changed 0% (0 of 2)
10   2015-08-07 19:57:07,486 DEBUG     pytan.handler.QuestionPoller: ID 1326: Progress: Tested: 0, Passed:
11   2015-08-07 19:57:07,486 DEBUG     pytan.handler.QuestionPoller: ID 1326: Timing: Started: 2015-08-07
12   2015-08-07 19:57:12,490 DEBUG     pytan.handler.QuestionPoller: ID 1326: Progress: Tested: 2, Passed:
13   2015-08-07 19:57:12,490 DEBUG     pytan.handler.QuestionPoller: ID 1326: Timing: Started: 2015-08-07
14   2015-08-07 19:57:12,490 INFO      pytan.handler.QuestionPoller: ID 1326: Progress Changed 100% (2 of
15   2015-08-07 19:57:12,490 INFO      pytan.handler.QuestionPoller: ID 1326: Reached Threshold of 99% (2
16   Traceback (most recent call last):
17     File "<string>", line 66, in <module>
18     File "/Users/jolsen/gh/pytan/lib/pytan/utils.py", line 2699, in wrap
19       ret = f(*args, **kwargs)
20     File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 1084, in export_obj
21       pytan.utils.check_dictkey(**check_args)
22     File "/Users/jolsen/gh/pytan/lib/pytan/utils.py", line 2692, in check_dictkey
23       raise pytan.exceptions.HandlerError(err(key, valid_list_types, list_types))
24   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

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

```
12    lib_dir = os.path.join(pytan_root_dir, 'lib')
13    path_adds = [lib_dir]
14
15    for aa in path_adds:
16        if aa not in sys.path:
17            sys.path.append(aa)
18
19
20    # connection info for Tanium Server
21    USERNAME = "Tanium User"
22    PASSWORD = "T@n!um"
23    HOST = "172.16.31.128"
24    PORT = "443"
25
26    # Logging conrols
27    LOGLEVEL = 2
28    DEBUGFORMAT = False
29
30    import tempfile
31
32    import pytan
33    handler = pytan.Handler(
34        username=USERNAME,
35        password=PASSWORD,
36        host=HOST,
37        port=PORT,
38        loglevel=LOGLEVEL,
39        debugformat=DEBUGFORMAT,
40    )
41
42    print handler
43
44    # setup the export_obj kwargs for later
45    export_kwargs = {}
46    export_kwargs["export_format"] = u'bad'
47
48    # ask the question that will provide the resultset that we want to use
49    ask_kwargs = {
50        'qtype': 'manual',
51        'sensors': [
52            "Computer Name"
53        ],
54    }
55    response = handler.ask(**ask_kwargs)
56    export_kwargs['obj'] = response['question_results']
57
58    # export the object to a string
59    # this should throw an exception: pytan.exceptions.HandlerError
60    import traceback
61
62    try:
63        handler.export_obj(**export_kwargs)
64    except Exception as e:
65        traceback.print_exc(file=sys.stdout)
```

Output from Python Code

```
1   Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2   2015-08-07 19:57:12,541 DEBUG    pytan.handler.QuestionPoller: ID 1327: id resolved to 1327
3   2015-08-07 19:57:12,541 DEBUG    pytan.handler.QuestionPoller: ID 1327: expiration resolved to 2015-
4   2015-08-07 19:57:12,541 DEBUG    pytan.handler.QuestionPoller: ID 1327: query_text resolved to Get C
5   2015-08-07 19:57:12,541 DEBUG    pytan.handler.QuestionPoller: ID 1327: id resolved to 1327
6   2015-08-07 19:57:12,541 DEBUG    pytan.handler.QuestionPoller: ID 1327: Object Info resolved to Ques
7   2015-08-07 19:57:12,545 DEBUG    pytan.handler.QuestionPoller: ID 1327: Progress: Tested: 0, Passed:
8   2015-08-07 19:57:12,545 DEBUG    pytan.handler.QuestionPoller: ID 1327: Timing: Started: 2015-08-07
9   2015-08-07 19:57:12,545 INFO     pytan.handler.QuestionPoller: ID 1327: Progress Changed 0% (0 of 2)
10  2015-08-07 19:57:17,551 DEBUG    pytan.handler.QuestionPoller: ID 1327: Progress: Tested: 0, Passed:
11  2015-08-07 19:57:17,551 DEBUG    pytan.handler.QuestionPoller: ID 1327: Timing: Started: 2015-08-07
12  2015-08-07 19:57:22,556 DEBUG    pytan.handler.QuestionPoller: ID 1327: Progress: Tested: 2, Passed:
13  2015-08-07 19:57:22,556 DEBUG    pytan.handler.QuestionPoller: ID 1327: Timing: Started: 2015-08-07
14  2015-08-07 19:57:22,556 INFO     pytan.handler.QuestionPoller: ID 1327: Progress Changed 100% (2 of
15  2015-08-07 19:57:22,556 INFO     pytan.handler.QuestionPoller: ID 1327: Reached Threshold of 99% (2
16  Traceback (most recent call last):
17    File "<string>", line 64, in <module>
18    File "/Users/jolsen/gh/pytan/lib/pytan/utils.py", line 2699, in wrap
19      ret = f(*args, **kwargs)
20    File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 1078, in export_obj
21      raise pytan.exceptions.HandlerError(err)
22  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
1   import os
2   import sys
3   sys.dont_write_bytecode = True
4
5   # Determine our script name, script dir
6   my_file = os.path.abspath(sys.argv[0])
7   my_dir = os.path.dirname(my_file)
8
9   # determine the pytan lib dir and add it to the path
10  parent_dir = os.path.dirname(my_dir)
11  pytan_root_dir = os.path.dirname(parent_dir)
12  lib_dir = os.path.join(pytan_root_dir, 'lib')
13  path_adds = [lib_dir]
14
15  for aa in path_adds:
16      if aa not in sys.path:
17          sys.path.append(aa)
18
19
20  # connection info for Tanium Server
21  USERNAME = "Tanium User"
22  PASSWORD = "T@n!um"
23  HOST = "172.16.31.128"
24  PORT = "443"
```

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

Output from Python Code

```
1   Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2
3   print the export_str returned from export_obj():
4   category,creation_time,delimiter,description,exclude_from_parse_flag,hash,hidden_flag,id,ignore_case
5   Reserved,,,"The assigned name of the client machine.
```

```
6   Example: workstation-1.company.com",0,3409330187,0,3,1,,86400,,,,,Computer Name,,Windows,select CSNa
7   Network,2015-08-07T13:22:12,|,"Returns IPv4 network routes, filtered to exclude noise. With Flags, M
8   Example:  172.16.0.0|192.168.1.1|255.255.0.0|UG|100|eth0",1,435227963,0,552,1,Jim Olsen,60,0,defined
9   Set objWMIService = GetObject(&quot;winmgmts:&quot; _
10      &amp; &quot;{impersonationLevel=impersonate}!\\&quot; &amp; strComputer &amp; &quot;\root\cimv2&
11
12  Set collip = objWMIService.ExecQuery(&quot;select * from win32_networkadapterconfiguration where IPE
13  dim ipaddrs()
14  ipcount = 0
15  for each ipItem in collip
16      for each ipaddr in ipItem.IPAddress
17          ipcount = ipcount + 1
18      next
19  ..trimmed for brevity..
```

Export basetype json type false

Export a BaseType from getting objects as JSON with false for include_type

Example Python Code

```python
1   import os
2   import sys
3   sys.dont_write_bytecode = True
4
5   # Determine our script name, script dir
6   my_file = os.path.abspath(sys.argv[0])
7   my_dir = os.path.dirname(my_file)
8
9   # determine the pytan lib dir and add it to the path
10  parent_dir = os.path.dirname(my_dir)
11  pytan_root_dir = os.path.dirname(parent_dir)
12  lib_dir = os.path.join(pytan_root_dir, 'lib')
13  path_adds = [lib_dir]
14
15  for aa in path_adds:
16      if aa not in sys.path:
17          sys.path.append(aa)
18
19
20  # connection info for Tanium Server
21  USERNAME = "Tanium User"
22  PASSWORD = "T@n!um"
23  HOST = "172.16.31.128"
24  PORT = "443"
25
26  # Logging conrols
27  LOGLEVEL = 2
28  DEBUGFORMAT = False
29
30  import tempfile
31
32  import pytan
33  handler = pytan.Handler(
34      username=USERNAME,
```

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

Output from Python Code

```
1    Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2
3    print the export_str returned from export_obj():
4    {
5      "sensor": [
6        {
7          "category": "Reserved",
8          "description": "The assigned name of the client machine.\nExample: workstation-1.company.com",
9          "exclude_from_parse_flag": 0,
10         "hash": 3409330187,
11         "hidden_flag": 0,
12         "id": 3,
13         "ignore_case_flag": 1,
14         "max_age_seconds": 86400,
```

```
15        "name": "Computer Name",
16        "queries": {
17          "query": [
18            {
19    ..trimmed for brevity..
```

Export basetype json explode false

Export a BaseType from getting objects as JSON with false for explode_json_string_values

Example Python Code

```python
1   import os
2   import sys
3   sys.dont_write_bytecode = True
4
5   # Determine our script name, script dir
6   my_file = os.path.abspath(sys.argv[0])
7   my_dir = os.path.dirname(my_file)
8
9   # determine the pytan lib dir and add it to the path
10  parent_dir = os.path.dirname(my_dir)
11  pytan_root_dir = os.path.dirname(parent_dir)
12  lib_dir = os.path.join(pytan_root_dir, 'lib')
13  path_adds = [lib_dir]
14
15  for aa in path_adds:
16      if aa not in sys.path:
17          sys.path.append(aa)
18
19
20  # connection info for Tanium Server
21  USERNAME = "Tanium User"
22  PASSWORD = "T@n!um"
23  HOST = "172.16.31.128"
24  PORT = "443"
25
26  # Logging conrols
27  LOGLEVEL = 2
28  DEBUGFORMAT = False
29
30  import tempfile
31
32  import pytan
33  handler = pytan.Handler(
34      username=USERNAME,
35      password=PASSWORD,
36      host=HOST,
37      port=PORT,
38      loglevel=LOGLEVEL,
39      debugformat=DEBUGFORMAT,
40  )
41
42  print handler
43
```

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

Output from Python Code

```
1    Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2
3    print the export_str returned from export_obj():
4    {
5      "_type": "sensors",
6      "sensor": [
7        {
8          "_type": "sensor",
9          "category": "Reserved",
10         "description": "The assigned name of the client machine.\nExample: workstation-1.company.com",
11         "exclude_from_parse_flag": 0,
12         "hash": 3409330187,
13         "hidden_flag": 0,
14         "id": 3,
15         "ignore_case_flag": 1,
16         "max_age_seconds": 86400,
17         "name": "Computer Name",
18         "queries": {
19   ..trimmed for brevity..
```

Export basetype json explode true

Export a BaseType from getting objects as JSON with true for explode_json_string_values

Example Python Code

```python
import os
import sys
sys.dont_write_bytecode = True

# Determine our script name, script dir
my_file = os.path.abspath(sys.argv[0])
my_dir = os.path.dirname(my_file)

# determine the pytan lib dir and add it to the path
parent_dir = os.path.dirname(my_dir)
pytan_root_dir = os.path.dirname(parent_dir)
lib_dir = os.path.join(pytan_root_dir, 'lib')
path_adds = [lib_dir]

for aa in path_adds:
    if aa not in sys.path:
        sys.path.append(aa)


# connection info for Tanium Server
USERNAME = "Tanium User"
PASSWORD = "T@n!um"
HOST = "172.16.31.128"
PORT = "443"

# Logging conrols
LOGLEVEL = 2
DEBUGFORMAT = False

import tempfile

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",
```

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

Output from Python Code

```
1  Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2
3  print the export_str returned from export_obj():
4  {
5    "_type": "sensors",
6    "sensor": [
7      {
8        "_type": "sensor",
9        "category": "Reserved",
10       "description": "The assigned name of the client machine.\nExample: workstation-1.company.com",
11       "exclude_from_parse_flag": 0,
12       "hash": 3409330187,
13       "hidden_flag": 0,
14       "id": 3,
15       "ignore_case_flag": 1,
16       "max_age_seconds": 86400,
17       "name": "Computer Name",
18       "queries": {
19  ..trimmed for brevity..
```

Export basetype xml default options

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

Example Python Code

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

```
4
5    # Determine our script name, script dir
6    my_file = os.path.abspath(sys.argv[0])
7    my_dir = os.path.dirname(my_file)
8
9    # determine the pytan lib dir and add it to the path
10   parent_dir = os.path.dirname(my_dir)
11   pytan_root_dir = os.path.dirname(parent_dir)
12   lib_dir = os.path.join(pytan_root_dir, 'lib')
13   path_adds = [lib_dir]
14
15   for aa in path_adds:
16       if aa not in sys.path:
17           sys.path.append(aa)
18
19
20   # connection info for Tanium Server
21   USERNAME = "Tanium User"
22   PASSWORD = "T@n!um"
23   HOST = "172.16.31.128"
24   PORT = "443"
25
26   # Logging conrols
27   LOGLEVEL = 2
28   DEBUGFORMAT = False
29
30   import tempfile
31
32   import pytan
33   handler = pytan.Handler(
34       username=USERNAME,
35       password=PASSWORD,
36       host=HOST,
37       port=PORT,
38       loglevel=LOGLEVEL,
39       debugformat=DEBUGFORMAT,
40   )
41
42   print handler
43
44   # setup the export_obj kwargs for later
45   export_kwargs = {}
46   export_kwargs["export_format"] = u'xml'
47
48   # get the objects that will provide the basetype that we want to use
49   get_kwargs = {
50       'name': [
51           "Computer Name", "IP Route Details", "IP Address",
52           'Folder Name Search with RegEx Match',
53       ],
54       'objtype': 'sensor',
55   }
56   response = handler.get(**get_kwargs)
57
58   # export the object to a string
59   # (we could just as easily export to a file using export_to_report_file)
60   export_kwargs['obj'] = response
61   export_str = handler.export_obj(**export_kwargs)
```

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

Output from Python Code

```
1    Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2
3    print the export_str returned from export_obj():
4    <sensors><cache_info /><sensor><category>Reserved</category><preview_sensor_flag /><hash>3409330187<
5    Example: workstation-1.company.com</description><string_hints /><subcolumns /><metadata /><parameter
6    Example:  172.16.0.0|192.168.1.1|255.255.0.0|UG|100|eth0</description><string_hints /><subcolumns><s
7    Set objWMIService = GetObject(&amp;quot;winmgmts:&amp;quot; _
8        &amp;amp; &amp;quot;{impersonationLevel=impersonate}!\\&amp;quot; &amp;amp; strComputer &amp;amp
9
10   Set collip = objWMIService.ExecQuery(&amp;quot;select * from win32_networkadapterconfiguration where
11   dim ipaddrs()
12   ipcount = 0
13   for each ipItem in collip
14       for each ipaddr in ipItem.IPAddress
15           ipcount = ipcount + 1
16       next
17   next
18   redim ipaddrs(ipcount)
19   ..trimmed for brevity..
```

Export basetype xml minimal false

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

Example Python Code

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

```
14
15  for aa in path_adds:
16      if aa not in sys.path:
17          sys.path.append(aa)
18
19
20  # connection info for Tanium Server
21  USERNAME = "Tanium User"
22  PASSWORD = "T@n!um"
23  HOST = "172.16.31.128"
24  PORT = "443"
25
26  # Logging conrols
27  LOGLEVEL = 2
28  DEBUGFORMAT = False
29
30  import tempfile
31
32  import pytan
33  handler = pytan.Handler(
34      username=USERNAME,
35      password=PASSWORD,
36      host=HOST,
37      port=PORT,
38      loglevel=LOGLEVEL,
39      debugformat=DEBUGFORMAT,
40  )
41
42  print handler
43
44  # setup the export_obj kwargs for later
45  export_kwargs = {}
46  export_kwargs["export_format"] = u'xml'
47  export_kwargs["minimal"] = False
48
49  # get the objects that will provide the basetype that we want to use
50  get_kwargs = {
51      'name': [
52          "Computer Name", "IP Route Details", "IP Address",
53          'Folder Name Search with RegEx Match',
54      ],
55      'objtype': 'sensor',
56  }
57  response = handler.get(**get_kwargs)
58
59  # export the object to a string
60  # (we could just as easily export to a file using export_to_report_file)
61  export_kwargs['obj'] = response
62  export_str = handler.export_obj(**export_kwargs)
63
64
65  print ""
66  print "print the export_str returned from export_obj():"
67
68  out = export_str
69  if len(out.splitlines()) > 15:
70      out = out.splitlines()[0:15]
71      out.append('..trimmed for brevity..')
```

```
72        out = '\n'.join(out)
73
74   print out
```

Output from Python Code

```
1    Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2
3    print the export_str returned from export_obj():
4    <sensors><cache_info /><sensor><category>Reserved</category><preview_sensor_flag /><hash>3409330187<
5    Example: workstation-1.company.com</description><string_hints /><subcolumns /><metadata /><parameter
6    Example:  172.16.0.0|192.168.1.1|255.255.0.0|UG|100|eth0</description><string_hints /><subcolumns><s
7    Set objWMIService = GetObject(&amp;quot;winmgmts:&amp;quot; _
8        &amp;amp; &amp;quot;{impersonationLevel=impersonate}!\\&amp;quot; &amp;amp; strComputer &amp;amp
9
10   Set collip = objWMIService.ExecQuery(&amp;quot;select * from win32_networkadapterconfiguration where
11   dim ipaddrs()
12   ipcount = 0
13   for each ipItem in collip
14       for each ipaddr in ipItem.IPAddress
15           ipcount = ipcount + 1
16       next
17   next
18   redim ipaddrs(ipcount)
19   ..trimmed for brevity..
```

Export basetype xml minimal true

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

Example Python Code

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

```
23  HOST = "172.16.31.128"
24  PORT = "443"
25
26  # Logging conrols
27  LOGLEVEL = 2
28  DEBUGFORMAT = False
29
30  import tempfile
31
32  import pytan
33  handler = pytan.Handler(
34      username=USERNAME,
35      password=PASSWORD,
36      host=HOST,
37      port=PORT,
38      loglevel=LOGLEVEL,
39      debugformat=DEBUGFORMAT,
40  )
41
42  print handler
43
44  # setup the export_obj kwargs for later
45  export_kwargs = {}
46  export_kwargs["export_format"] = u'xml'
47  export_kwargs["minimal"] = True
48
49  # get the objects that will provide the basetype that we want to use
50  get_kwargs = {
51      'name': [
52          "Computer Name", "IP Route Details", "IP Address",
53          'Folder Name Search with RegEx Match',
54      ],
55      'objtype': 'sensor',
56  }
57  response = handler.get(**get_kwargs)
58
59  # export the object to a string
60  # (we could just as easily export to a file using export_to_report_file)
61  export_kwargs['obj'] = response
62  export_str = handler.export_obj(**export_kwargs)
63
64
65  print ""
66  print "print the export_str returned from export_obj():"
67
68  out = export_str
69  if len(out.splitlines()) > 15:
70      out = out.splitlines()[0:15]
71      out.append('..trimmed for brevity..')
72      out = '\n'.join(out)
73
74  print out
```

Output from Python Code

```
1  Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2
```

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

Export basetype csv with explode false

Export a BaseType from getting objects as CSV with false for explode_json_string_values

Example Python Code

```python
1   import os
2   import sys
3   sys.dont_write_bytecode = True
4
5   # Determine our script name, script dir
6   my_file = os.path.abspath(sys.argv[0])
7   my_dir = os.path.dirname(my_file)
8
9   # determine the pytan lib dir and add it to the path
10  parent_dir = os.path.dirname(my_dir)
11  pytan_root_dir = os.path.dirname(parent_dir)
12  lib_dir = os.path.join(pytan_root_dir, 'lib')
13  path_adds = [lib_dir]
14
15  for aa in path_adds:
16      if aa not in sys.path:
17          sys.path.append(aa)
18
19
20  # connection info for Tanium Server
21  USERNAME = "Tanium User"
22  PASSWORD = "T@n!um"
23  HOST = "172.16.31.128"
24  PORT = "443"
25
26  # Logging conrols
27  LOGLEVEL = 2
28  DEBUGFORMAT = False
29
30  import tempfile
31
```

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

Output from Python Code

```
1    Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2
3    print the export_str returned from export_obj():
4    category,creation_time,delimiter,description,exclude_from_parse_flag,hash,hidden_flag,id,ignore_case
5    Reserved,,,"The assigned name of the client machine.
6    Example: workstation-1.company.com",0,3409330187,0,3,1,,86400,,,,,Computer Name,,Windows,select CSNa
7    Network,2015-08-07T13:22:12,|,"Returns IPv4 network routes, filtered to exclude noise. With Flags, M
8    Example:  172.16.0.0|192.168.1.1|255.255.0.0|UG|100|eth0",1,435227963,0,552,1,Jim Olsen,60,0,defined
9    Set objWMIService = GetObject(&quot;winmgmts:&quot; _
10        &amp; &quot;{impersonationLevel=impersonate}!\\&quot; &amp; strComputer &amp; &quot;\root\cimv2&
11
```

```
12  Set collip = objWMIService.ExecQuery(&quot;select * from win32_networkadapterconfiguration where IPE
13  dim ipaddrs()
14  ipcount = 0
15  for each ipItem in collip
16      for each ipaddr in ipItem.IPAddress
17          ipcount = ipcount + 1
18      next
19  ..trimmed for brevity..
```

Export basetype csv with explode true

Export a BaseType from getting objects as CSV with true for explode_json_string_values

Example Python Code

```python
import os
import sys
sys.dont_write_bytecode = True

# Determine our script name, script dir
my_file = os.path.abspath(sys.argv[0])
my_dir = os.path.dirname(my_file)

# determine the pytan lib dir and add it to the path
parent_dir = os.path.dirname(my_dir)
pytan_root_dir = os.path.dirname(parent_dir)
lib_dir = os.path.join(pytan_root_dir, 'lib')
path_adds = [lib_dir]

for aa in path_adds:
    if aa not in sys.path:
        sys.path.append(aa)


# connection info for Tanium Server
USERNAME = "Tanium User"
PASSWORD = "T@n!um"
HOST = "172.16.31.128"
PORT = "443"

# Logging conrols
LOGLEVEL = 2
DEBUGFORMAT = False

import tempfile

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

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

Output from Python Code

```
1   Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2
3   print the export_str returned from export_obj():
4   category,creation_time,delimiter,description,exclude_from_parse_flag,hash,hidden_flag,id,ignore_case
5   Reserved,,,"The assigned name of the client machine.
6   Example: workstation-1.company.com",0,3409330187,0,3,1,,86400,,,,,Computer Name,,,,,,,,,,,,,,,,,,,,,
7   Network,2015-08-07T13:22:12,|,"Returns IPv4 network routes, filtered to exclude noise. With Flags, M
8   Example:  172.16.0.0|192.168.1.1|255.255.0.0|UG|100|eth0",1,435227963,0,552,1,Jim Olsen,60,0,defined
9   Set objWMIService = GetObject(&quot;winmgmts:&quot; _
10      &amp; &quot;{impersonationLevel=impersonate}!\\&quot; &amp; strComputer &amp; &quot;\root\cimv2&
11
12  Set collip = objWMIService.ExecQuery(&quot;select * from win32_networkadapterconfiguration where IPE
13  dim ipaddrs()
14  ipcount = 0
15  for each ipItem in collip
16      for each ipaddr in ipItem.IPAddress
17          ipcount = ipcount + 1
18      next
19  ..trimmed for brevity..
```

Export basetype csv with sort empty list

Export a BaseType from getting objects as CSV with an empty list for header_sort

Example Python Code

```python
import os
import sys
sys.dont_write_bytecode = True

# Determine our script name, script dir
my_file = os.path.abspath(sys.argv[0])
my_dir = os.path.dirname(my_file)

# determine the pytan lib dir and add it to the path
parent_dir = os.path.dirname(my_dir)
pytan_root_dir = os.path.dirname(parent_dir)
lib_dir = os.path.join(pytan_root_dir, 'lib')
path_adds = [lib_dir]

for aa in path_adds:
    if aa not in sys.path:
        sys.path.append(aa)


# connection info for Tanium Server
USERNAME = "Tanium User"
PASSWORD = "T@n!um"
HOST = "172.16.31.128"
PORT = "443"

# Logging conrols
LOGLEVEL = 2
DEBUGFORMAT = False

import tempfile

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",
```

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

Output from Python Code

```
1   Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2
3   print the export_str returned from export_obj():
4   category,creation_time,delimiter,description,exclude_from_parse_flag,hash,hidden_flag,id,ignore_case
5   Reserved,,,"The assigned name of the client machine.
6   Example: workstation-1.company.com",0,3409330187,0,3,1,,86400,,,,,Computer Name,,Windows,select CSNa
7   Network,2015-08-07T13:22:12,|,"Returns IPv4 network routes, filtered to exclude noise. With Flags, M
8   Example:  172.16.0.0|192.168.1.1|255.255.0.0|UG|100|eth0",1,435227963,0,552,1,Jim Olsen,60,0,defined
9   Set objWMIService = GetObject(&quot;winmgmts:&quot; _
10      &amp; &quot;{impersonationLevel=impersonate}!\\&quot; &amp; strComputer &amp; &quot;\root\cimv2&
11
12  Set collip = objWMIService.ExecQuery(&quot;select * from win32_networkadapterconfiguration where IPE
13  dim ipaddrs()
14  ipcount = 0
15  for each ipItem in collip
16      for each ipaddr in ipItem.IPAddress
17          ipcount = ipcount + 1
18      next
19  ..trimmed for brevity..
```

Export basetype csv with sort true

Export a BaseType from getting objects as CSV with true for header_sort

Example Python Code

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

```python
4
5    # Determine our script name, script dir
6    my_file = os.path.abspath(sys.argv[0])
7    my_dir = os.path.dirname(my_file)
8
9    # determine the pytan lib dir and add it to the path
10   parent_dir = os.path.dirname(my_dir)
11   pytan_root_dir = os.path.dirname(parent_dir)
12   lib_dir = os.path.join(pytan_root_dir, 'lib')
13   path_adds = [lib_dir]
14
15   for aa in path_adds:
16       if aa not in sys.path:
17           sys.path.append(aa)
18
19
20   # connection info for Tanium Server
21   USERNAME = "Tanium User"
22   PASSWORD = "T@n!um"
23   HOST = "172.16.31.128"
24   PORT = "443"
25
26   # Logging conrols
27   LOGLEVEL = 2
28   DEBUGFORMAT = False
29
30   import tempfile
31
32   import pytan
33   handler = pytan.Handler(
34       username=USERNAME,
35       password=PASSWORD,
36       host=HOST,
37       port=PORT,
38       loglevel=LOGLEVEL,
39       debugformat=DEBUGFORMAT,
40   )
41
42   print handler
43
44   # setup the export_obj kwargs for later
45   export_kwargs = {}
46   export_kwargs["export_format"] = u'csv'
47   export_kwargs["header_sort"] = True
48
49   # get the objects that will provide the basetype that we want to use
50   get_kwargs = {
51       'name': [
52           "Computer Name", "IP Route Details", "IP Address",
53           'Folder Name Search with RegEx Match',
54       ],
55       'objtype': 'sensor',
56   }
57   response = handler.get(**get_kwargs)
58
59   # export the object to a string
60   # (we could just as easily export to a file using export_to_report_file)
61   export_kwargs['obj'] = response
```

```
62   export_str = handler.export_obj(**export_kwargs)
63
64
65   print ""
66   print "print the export_str returned from export_obj():"
67
68   out = export_str
69   if len(out.splitlines()) > 15:
70       out = out.splitlines()[0:15]
71       out.append('..trimmed for brevity..')
72       out = '\n'.join(out)
73
74   print out
```

Output from Python Code

```
1    Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2
3    print the export_str returned from export_obj():
4    category,creation_time,delimiter,description,exclude_from_parse_flag,hash,hidden_flag,id,ignore_case
5    Reserved,,,"The assigned name of the client machine.
6    Example: workstation-1.company.com",0,3409330187,0,3,1,,86400,,,,,Computer Name,,Windows,select CSNa
7    Network,2015-08-07T13:22:12,|,"Returns IPv4 network routes, filtered to exclude noise. With Flags, M
8    Example:  172.16.0.0|192.168.1.1|255.255.0.0|UG|100|eth0",1,435227963,0,552,1,Jim Olsen,60,0,defined
9    Set objWMIService = GetObject(&quot;winmgmts:&quot; _
10       &amp; &quot;{impersonationLevel=impersonate}!\\&quot; &amp; strComputer &amp; &quot;\root\cimv2&
11
12   Set collip = objWMIService.ExecQuery(&quot;select * from win32_networkadapterconfiguration where IPE
13   dim ipaddrs()
14   ipcount = 0
15   for each ipItem in collip
16       for each ipaddr in ipItem.IPAddress
17           ipcount = ipcount + 1
18       next
19   ..trimmed for brevity..
```

Export basetype csv with sort list

Export a BaseType from getting objects as CSV with name and description for header_sort

Example Python Code

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

```
13    path_adds = [lib_dir]

14

15    for aa in path_adds:
16        if aa not in sys.path:
17            sys.path.append(aa)

18

19

20    # connection info for Tanium Server
21    USERNAME = "Tanium User"
22    PASSWORD = "T@n!um"
23    HOST = "172.16.31.128"
24    PORT = "443"

25

26    # Logging conrols
27    LOGLEVEL = 2
28    DEBUGFORMAT = False

29

30    import tempfile

31

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

41

42    print handler

43

44    # setup the export_obj kwargs for later
45    export_kwargs = {}
46    export_kwargs["export_format"] = u'csv'
47    export_kwargs["header_sort"] = [u'name', u'description']

48

49    # get the objects that will provide the basetype that we want to use
50    get_kwargs = {
51        'name': [
52            "Computer Name", "IP Route Details", "IP Address",
53            'Folder Name Search with RegEx Match',
54        ],
55        'objtype': 'sensor',
56    }
57    response = handler.get(**get_kwargs)

58

59    # export the object to a string
60    # (we could just as easily export to a file using export_to_report_file)
61    export_kwargs['obj'] = response
62    export_str = handler.export_obj(**export_kwargs)

63

64

65    print ""
66    print "print the export_str returned from export_obj():"

67

68    out = export_str
69    if len(out.splitlines()) > 15:
70        out = out.splitlines()[0:15]
```

```
71        out.append('..trimmed for brevity..')
72        out = '\n'.join(out)
73
74    print out
```

Output from Python Code

```
1    Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2
3    print the export_str returned from export_obj():
4    name,description,category,creation_time,delimiter,exclude_from_parse_flag,hash,hidden_flag,id,ignore
5    Computer Name,"The assigned name of the client machine.
6    Example: workstation-1.company.com",Reserved,,,0,3409330187,0,3,1,,86400,,,,,,Windows,select CSName
7    IP Route Details,"Returns IPv4 network routes, filtered to exclude noise. With Flags, Metric, Interf
8    Example:  172.16.0.0|192.168.1.1|255.255.0.0|UG|100|eth0",Network,2015-08-07T13:22:12,|,,1,435227963,
9    Set objWMIService = GetObject(&quot;winmgmts:&quot; _
10       &amp; &quot;{impersonationLevel=impersonate}!\\&quot; &amp; strComputer &amp; &quot;\root\cimv2&
11
12   Set collip = objWMIService.ExecQuery(&quot;select * from win32_networkadapterconfiguration where IPE
13   dim ipaddrs()
14   ipcount = 0
15   for each ipItem in collip
16       for each ipaddr in ipItem.IPAddress
17           ipcount = ipcount + 1
18       next
19   ..trimmed for brevity..
```

Export basetype json default options

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

Example Python Code

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

```
22   PASSWORD = "T@n!um"
23   HOST = "172.16.31.128"
24   PORT = "443"
25
26   # Logging conrols
27   LOGLEVEL = 2
28   DEBUGFORMAT = False
29
30   import tempfile
31
32   import pytan
33   handler = pytan.Handler(
34       username=USERNAME,
35       password=PASSWORD,
36       host=HOST,
37       port=PORT,
38       loglevel=LOGLEVEL,
39       debugformat=DEBUGFORMAT,
40   )
41
42   print handler
43
44   # setup the export_obj kwargs for later
45   export_kwargs = {}
46   export_kwargs["export_format"] = u'json'
47
48   # get the objects that will provide the basetype that we want to use
49   get_kwargs = {
50       'name': [
51           "Computer Name", "IP Route Details", "IP Address",
52           'Folder Name Search with RegEx Match',
53       ],
54       'objtype': 'sensor',
55   }
56   response = handler.get(**get_kwargs)
57
58   # export the object to a string
59   # (we could just as easily export to a file using export_to_report_file)
60   export_kwargs['obj'] = response
61   export_str = handler.export_obj(**export_kwargs)
62
63
64   print ""
65   print "print the export_str returned from export_obj():"
66
67   out = export_str
68   if len(out.splitlines()) > 15:
69       out = out.splitlines()[0:15]
70       out.append('..trimmed for brevity..')
71       out = '\n'.join(out)
72
73   print out
```

Output from Python Code

```
1   Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2
```

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

Export basetype json type true

Export a BaseType from getting objects as JSON with true for include_type

Example Python Code

```python
1    import os
2    import sys
3    sys.dont_write_bytecode = True
4
5    # Determine our script name, script dir
6    my_file = os.path.abspath(sys.argv[0])
7    my_dir = os.path.dirname(my_file)
8
9    # determine the pytan lib dir and add it to the path
10   parent_dir = os.path.dirname(my_dir)
11   pytan_root_dir = os.path.dirname(parent_dir)
12   lib_dir = os.path.join(pytan_root_dir, 'lib')
13   path_adds = [lib_dir]
14
15   for aa in path_adds:
16       if aa not in sys.path:
17           sys.path.append(aa)
18
19
20   # connection info for Tanium Server
21   USERNAME = "Tanium User"
22   PASSWORD = "T@n!um"
23   HOST = "172.16.31.128"
24   PORT = "443"
25
26   # Logging conrols
27   LOGLEVEL = 2
28   DEBUGFORMAT = False
29
30   import tempfile
31
```

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

Output from Python Code

```
1   Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2
3   print the export_str returned from export_obj():
4   {
5     "_type": "sensors",
6     "sensor": [
7       {
8         "_type": "sensor",
9         "category": "Reserved",
10        "description": "The assigned name of the client machine.\nExample: workstation-1.company.com",
11        "exclude_from_parse_flag": 0,
```

```
12        "hash": 3409330187,
13        "hidden_flag": 0,
14        "id": 3,
15        "ignore_case_flag": 1,
16        "max_age_seconds": 86400,
17        "name": "Computer Name",
18        "queries": {
19   ..trimmed for brevity..
```

PyTan API Invalid Export BaseType Examples

Invalid export basetype csv bad explode type

Export a BaseType from getting objects using a bad explode_json_string_values

Example Python Code

```python
1    import os
2    import sys
3    sys.dont_write_bytecode = True
4
5    # Determine our script name, script dir
6    my_file = os.path.abspath(sys.argv[0])
7    my_dir = os.path.dirname(my_file)
8
9    # determine the pytan lib dir and add it to the path
10   parent_dir = os.path.dirname(my_dir)
11   pytan_root_dir = os.path.dirname(parent_dir)
12   lib_dir = os.path.join(pytan_root_dir, 'lib')
13   path_adds = [lib_dir]
14
15   for aa in path_adds:
16       if aa not in sys.path:
17           sys.path.append(aa)
18
19
20   # connection info for Tanium Server
21   USERNAME = "Tanium User"
22   PASSWORD = "T@n!um"
23   HOST = "172.16.31.128"
24   PORT = "443"
25
26   # Logging conrols
27   LOGLEVEL = 2
28   DEBUGFORMAT = False
29
30   import tempfile
31
32   import pytan
33   handler = pytan.Handler(
34       username=USERNAME,
35       password=PASSWORD,
36       host=HOST,
37       port=PORT,
38       loglevel=LOGLEVEL,
```

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

Output from Python Code

```
1   Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2   Traceback (most recent call last):
3     File "<string>", line 66, in <module>
4     File "/Users/jolsen/gh/pytan/lib/pytan/utils.py", line 2699, in wrap
5       ret = f(*args, **kwargs)
6     File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 1084, in export_obj
7       pytan.utils.check_dictkey(**check_args)
8     File "/Users/jolsen/gh/pytan/lib/pytan/utils.py", line 2685, in check_dictkey
9       raise pytan.exceptions.HandlerError(err(key, valid_types, k_type))
10  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

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

```
6   my_file = os.path.abspath(sys.argv[0])
7   my_dir = os.path.dirname(my_file)
8
9   # determine the pytan lib dir and add it to the path
10  parent_dir = os.path.dirname(my_dir)
11  pytan_root_dir = os.path.dirname(parent_dir)
12  lib_dir = os.path.join(pytan_root_dir, 'lib')
13  path_adds = [lib_dir]
14
15  for aa in path_adds:
16      if aa not in sys.path:
17          sys.path.append(aa)
18
19
20  # connection info for Tanium Server
21  USERNAME = "Tanium User"
22  PASSWORD = "T@n!um"
23  HOST = "172.16.31.128"
24  PORT = "443"
25
26  # Logging conrols
27  LOGLEVEL = 2
28  DEBUGFORMAT = False
29
30  import tempfile
31
32  import pytan
33  handler = pytan.Handler(
34      username=USERNAME,
35      password=PASSWORD,
36      host=HOST,
37      port=PORT,
38      loglevel=LOGLEVEL,
39      debugformat=DEBUGFORMAT,
40  )
41
42  print handler
43
44  # setup the export_obj kwargs for later
45  export_kwargs = {}
46  export_kwargs["export_format"] = u'csv'
47  export_kwargs["header_sort"] = [[]]
48
49  # get the objects that will provide the basetype that we want to use
50  get_kwargs = {
51      'name': [
52          "Computer Name", "IP Route Details", "IP Address",
53          'Folder Name Search with RegEx Match',
54      ],
55      'objtype': 'sensor',
56  }
57  response = handler.get(**get_kwargs)
58  export_kwargs['obj'] = response
59
60  # export the object to a string
61  # this should throw an exception: pytan.exceptions.HandlerError
62  import traceback
63
```

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

Output from Python Code

```
1   Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2   Traceback (most recent call last):
3     File "<string>", line 66, in <module>
4     File "/Users/jolsen/gh/pytan/lib/pytan/utils.py", line 2699, in wrap
5       ret = f(*args, **kwargs)
6     File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 1084, in export_obj
7       pytan.utils.check_dictkey(**check_args)
8     File "/Users/jolsen/gh/pytan/lib/pytan/utils.py", line 2692, in check_dictkey
9       raise pytan.exceptions.HandlerError(err(key, valid_list_types, list_types))
10  HandlerError: 'header_sort' must be a list of [<type 'str'>, <type 'unicode'>], you supplied [<type
```

Invalid export basetype csv bad sort type

Export a BaseType from getting objects using a bad header_sort

Example Python Code

```
1   import os
2   import sys
3   sys.dont_write_bytecode = True
4
5   # Determine our script name, script dir
6   my_file = os.path.abspath(sys.argv[0])
7   my_dir = os.path.dirname(my_file)
8
9   # determine the pytan lib dir and add it to the path
10  parent_dir = os.path.dirname(my_dir)
11  pytan_root_dir = os.path.dirname(parent_dir)
12  lib_dir = os.path.join(pytan_root_dir, 'lib')
13  path_adds = [lib_dir]
14
15  for aa in path_adds:
16      if aa not in sys.path:
17          sys.path.append(aa)
18
19
20  # connection info for Tanium Server
21  USERNAME = "Tanium User"
22  PASSWORD = "T@n!um"
23  HOST = "172.16.31.128"
24  PORT = "443"
25
26  # Logging conrols
27  LOGLEVEL = 2
28  DEBUGFORMAT = False
29
30  import tempfile
```

```
31
32  import pytan
33  handler = pytan.Handler(
34      username=USERNAME,
35      password=PASSWORD,
36      host=HOST,
37      port=PORT,
38      loglevel=LOGLEVEL,
39      debugformat=DEBUGFORMAT,
40  )
41
42  print handler
43
44  # setup the export_obj kwargs for later
45  export_kwargs = {}
46  export_kwargs["export_format"] = u'csv'
47  export_kwargs["header_sort"] = u'bad'
48
49  # get the objects that will provide the basetype that we want to use
50  get_kwargs = {
51      'name': [
52          "Computer Name", "IP Route Details", "IP Address",
53          'Folder Name Search with RegEx Match',
54      ],
55      'objtype': 'sensor',
56  }
57  response = handler.get(**get_kwargs)
58  export_kwargs['obj'] = response
59
60  # export the object to a string
61  # this should throw an exception: pytan.exceptions.HandlerError
62  import traceback
63
64  try:
65      handler.export_obj(**export_kwargs)
66  except Exception as e:
67      traceback.print_exc(file=sys.stdout)
```

Output from Python Code

```
1   Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2   Traceback (most recent call last):
3     File "<string>", line 66, in <module>
4     File "/Users/jolsen/gh/pytan/lib/pytan/utils.py", line 2699, in wrap
5       ret = f(*args, **kwargs)
6     File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 1084, in export_obj
7       pytan.utils.check_dictkey(**check_args)
8     File "/Users/jolsen/gh/pytan/lib/pytan/utils.py", line 2685, in check_dictkey
9       raise pytan.exceptions.HandlerError(err(key, valid_types, k_type))
10  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

```python
import os
import sys
sys.dont_write_bytecode = True

# Determine our script name, script dir
my_file = os.path.abspath(sys.argv[0])
my_dir = os.path.dirname(my_file)

# determine the pytan lib dir and add it to the path
parent_dir = os.path.dirname(my_dir)
pytan_root_dir = os.path.dirname(parent_dir)
lib_dir = os.path.join(pytan_root_dir, 'lib')
path_adds = [lib_dir]

for aa in path_adds:
    if aa not in sys.path:
        sys.path.append(aa)


# connection info for Tanium Server
USERNAME = "Tanium User"
PASSWORD = "T@n!um"
HOST = "172.16.31.128"
PORT = "443"

# Logging conrols
LOGLEVEL = 2
DEBUGFORMAT = False

import tempfile

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)
```

```
58   export_kwargs['obj'] = response
59
60   # export the object to a string
61   # this should throw an exception: pytan.exceptions.HandlerError
62   import traceback
63
64   try:
65       handler.export_obj(**export_kwargs)
66   except Exception as e:
67       traceback.print_exc(file=sys.stdout)
```

Output from Python Code

```
1    Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2    Traceback (most recent call last):
3      File "<string>", line 66, in <module>
4      File "/Users/jolsen/gh/pytan/lib/pytan/utils.py", line 2699, in wrap
5        ret = f(*args, **kwargs)
6      File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 1084, in export_obj
7        pytan.utils.check_dictkey(**check_args)
8      File "/Users/jolsen/gh/pytan/lib/pytan/utils.py", line 2685, in check_dictkey
9        raise pytan.exceptions.HandlerError(err(key, valid_types, k_type))
10   HandlerError: 'minimal' must be one of [<type 'bool'>], you supplied <type 'unicode'>!
```

Invalid export basetype json bad include type

Export a BaseType from getting objects using a bad include_type

Example Python Code

```
1    import os
2    import sys
3    sys.dont_write_bytecode = True
4
5    # Determine our script name, script dir
6    my_file = os.path.abspath(sys.argv[0])
7    my_dir = os.path.dirname(my_file)
8
9    # determine the pytan lib dir and add it to the path
10   parent_dir = os.path.dirname(my_dir)
11   pytan_root_dir = os.path.dirname(parent_dir)
12   lib_dir = os.path.join(pytan_root_dir, 'lib')
13   path_adds = [lib_dir]
14
15   for aa in path_adds:
16       if aa not in sys.path:
17           sys.path.append(aa)
18
19
20   # connection info for Tanium Server
21   USERNAME = "Tanium User"
22   PASSWORD = "T@n!um"
23   HOST = "172.16.31.128"
24   PORT = "443"
```

```
25
26   # Logging conrols
27   LOGLEVEL = 2
28   DEBUGFORMAT = False
29
30   import tempfile
31
32   import pytan
33   handler = pytan.Handler(
34       username=USERNAME,
35       password=PASSWORD,
36       host=HOST,
37       port=PORT,
38       loglevel=LOGLEVEL,
39       debugformat=DEBUGFORMAT,
40   )
41
42   print handler
43
44   # setup the export_obj kwargs for later
45   export_kwargs = {}
46   export_kwargs["export_format"] = u'json'
47   export_kwargs["include_type"] = u'bad'
48
49   # get the objects that will provide the basetype that we want to use
50   get_kwargs = {
51       'name': [
52           "Computer Name", "IP Route Details", "IP Address",
53           'Folder Name Search with RegEx Match',
54       ],
55       'objtype': 'sensor',
56   }
57   response = handler.get(**get_kwargs)
58   export_kwargs['obj'] = response
59
60   # export the object to a string
61   # this should throw an exception: pytan.exceptions.HandlerError
62   import traceback
63
64   try:
65       handler.export_obj(**export_kwargs)
66   except Exception as e:
67       traceback.print_exc(file=sys.stdout)
```

Output from Python Code

```
1   Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2   Traceback (most recent call last):
3     File "<string>", line 66, in <module>
4     File "/Users/jolsen/gh/pytan/lib/pytan/utils.py", line 2699, in wrap
5       ret = f(*args, **kwargs)
6     File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 1084, in export_obj
7       pytan.utils.check_dictkey(**check_args)
8     File "/Users/jolsen/gh/pytan/lib/pytan/utils.py", line 2685, in check_dictkey
9       raise pytan.exceptions.HandlerError(err(key, valid_types, k_type))
10  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

```python
import os
import sys
sys.dont_write_bytecode = True

# Determine our script name, script dir
my_file = os.path.abspath(sys.argv[0])
my_dir = os.path.dirname(my_file)

# determine the pytan lib dir and add it to the path
parent_dir = os.path.dirname(my_dir)
pytan_root_dir = os.path.dirname(parent_dir)
lib_dir = os.path.join(pytan_root_dir, 'lib')
path_adds = [lib_dir]

for aa in path_adds:
    if aa not in sys.path:
        sys.path.append(aa)


# connection info for Tanium Server
USERNAME = "Tanium User"
PASSWORD = "T@n!um"
HOST = "172.16.31.128"
PORT = "443"

# Logging conrols
LOGLEVEL = 2
DEBUGFORMAT = False

import tempfile

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",
```

```
53          'Folder Name Search with RegEx Match',
54      ],
55      'objtype': 'sensor',
56  }
57  response = handler.get(**get_kwargs)
58  export_kwargs['obj'] = response
59
60  # export the object to a string
61  # this should throw an exception: pytan.exceptions.HandlerError
62  import traceback
63
64  try:
65      handler.export_obj(**export_kwargs)
66  except Exception as e:
67      traceback.print_exc(file=sys.stdout)
```

Output from Python Code

```
1   Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2   Traceback (most recent call last):
3     File "<string>", line 66, in <module>
4     File "/Users/jolsen/gh/pytan/lib/pytan/utils.py", line 2699, in wrap
5       ret = f(*args, **kwargs)
6     File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 1084, in export_obj
7       pytan.utils.check_dictkey(**check_args)
8     File "/Users/jolsen/gh/pytan/lib/pytan/utils.py", line 2685, in check_dictkey
9       raise pytan.exceptions.HandlerError(err(key, valid_types, k_type))
10  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

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

```python
20  # connection info for Tanium Server
21  USERNAME = "Tanium User"
22  PASSWORD = "T@n!um"
23  HOST = "172.16.31.128"
24  PORT = "443"
25
26  # Logging conrols
27  LOGLEVEL = 2
28  DEBUGFORMAT = False
29
30  import tempfile
31
32  import pytan
33  handler = pytan.Handler(
34      username=USERNAME,
35      password=PASSWORD,
36      host=HOST,
37      port=PORT,
38      loglevel=LOGLEVEL,
39      debugformat=DEBUGFORMAT,
40  )
41
42  print handler
43
44  # setup the export_obj kwargs for later
45  export_kwargs = {}
46  export_kwargs["export_format"] = u'bad'
47
48  # get the objects that will provide the basetype that we want to use
49  get_kwargs = {
50      'name': [
51          "Computer Name", "IP Route Details", "IP Address",
52          'Folder Name Search with RegEx Match',
53      ],
54      'objtype': 'sensor',
55  }
56  response = handler.get(**get_kwargs)
57  export_kwargs['obj'] = response
58
59  # export the object to a string
60  # this should throw an exception: pytan.exceptions.HandlerError
61  import traceback
62
63  try:
64      handler.export_obj(**export_kwargs)
65  except Exception as e:
66      traceback.print_exc(file=sys.stdout)
```

Output from Python Code

```
1  Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2  Traceback (most recent call last):
3    File "<string>", line 65, in <module>
4    File "/Users/jolsen/gh/pytan/lib/pytan/utils.py", line 2699, in wrap
5      ret = f(*args, **kwargs)
6    File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 1078, in export_obj
7      raise pytan.exceptions.HandlerError(err)
```

```
8   HandlerError: u'bad' not a supported export format for SensorList, must be one of: xml, json, csv
```

1.3 taniumpy package

A python package that handles the serialization/deserialization of XML SOAP requests/responses from Tanium to/from python objects.

1.3.1 taniumpy.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 Base-Type.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.permission_list module

class taniumpy.object_types.permission_list.**PermissionList**
    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 ResultSet from a result_set SOAPElement
>
> **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.string_hint_list module

class taniumpy.object_types.string_hint_list.**StringHintList**
> 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_setting_list module

class taniumpy.object_types.system_setting_list.**SystemSettingList**
> 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_role module

class taniumpy.object_types.user_role.**UserRole**
    Bases: taniumpy.object_types.base.BaseType

taniumpy.object_types.user_role_list module

class taniumpy.object_types.user_role_list.**UserRoleList**
    Bases: taniumpy.object_types.base.BaseType

taniumpy.object_types.version_aggregate module

class taniumpy.object_types.version_aggregate.**VersionAggregate**
    Bases: taniumpy.object_types.base.BaseType

taniumpy.object_types.version_aggregate_list module

class taniumpy.object_types.version_aggregate_list.**VersionAggregateList**
    Bases: taniumpy.object_types.base.BaseType

taniumpy.object_types.white_listed_url module

class taniumpy.object_types.white_listed_url.**WhiteListedUrl**
    Bases: taniumpy.object_types.base.BaseType

taniumpy.object_types.white_listed_url_list module

class taniumpy.object_types.white_listed_url_list.**WhiteListedUrlList**
    Bases: `taniumpy.object_types.base.BaseType`

taniumpy.object_types.xml_error module

class taniumpy.object_types.xml_error.**XmlError**
    Bases: `taniumpy.object_types.base.BaseType`

1.4 xmltodict module

Makes working with XML feel like you are working with JSON

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.

1.5 ddt module

ddt.**data**(**values*)
Method decorator to add to your test methods.

Should be added to methods of instances of unittest.TestCase.

ddt.**ddt**(*cls*)
Class decorator for subclasses of unittest.TestCase.

Apply this decorator to the test case class, and then decorate test methods with @data.

For each method decorated with @data, this will effectively create as many methods as data items are passed as parameters to @data.

The names of the test methods follow the pattern original_test_name_{ordinal}_{data}. ordinal is the position of the data argument, starting with 1.

For data we use a string representation of the data value converted into a valid python identifier. If data.__name__ exists, we use that instead.

For each method decorated with @file_data('test_data.json'), the decorator will try to load the test_data.json file located relative to the python file containing the method that is decorated. It will, for each test_name key create as many methods in the list of values from the data key.

---

ddt.**file_data**(*value*)
> Method decorator to add to your test methods.
>
> Should be added to methods of instances of `unittest.TestCase`.
>
> `value` should be a path relative to the directory of the file containing the decorated `unittest.TestCase`. The file should contain JSON encoded data, that can either be a list or a dict.
>
> In case of a list, each value in the list will correspond to one test case, and the value will be concatenated to the test method name.
>
> In case of a dict, keys will be used as suffixes to the name of the test case, and values will be fed as test data.

ddt.**is_hash_randomized**()

ddt.**mk_test_name**(*name*, *value*, *index=0*)
> Generate a new name for a test case.
>
> It will take the original test name and append an ordinal index and a string representation of the value, and convert the result into a valid python identifier by replacing extraneous characters with _.
>
> If hash randomization is enabled (a feature available since 2.7.3/3.2.3 and enabled by default since 3.3) and a "non-trivial" value is passed this will omit the name argument by default. Set *PYTHONHASHSEED* to a fixed value before running tests in these cases to get the names back consistently or use the *__name__* attribute on data values.
>
> A "trivial" value is a plain scalar, or a tuple or list consisting only of trivial values.

ddt.**unpack**(*func*)
> Method decorator to add unpack feature.

1.6 threaded_http module

Simple HTTP server for testing purposes

class threaded_http.**CustomHTTPHandler**(*request*, *client_address*, *server*)
> Bases: `BaseHTTPServer.BaseHTTPRequestHandler`
>
> **ENABLE_LOGGING = True**
>
> **__module__ = 'threaded_http'**
>
> **do_GET**()
>
> **do_POST**()
>
> **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

1.7 requests package

1.7.1 requests HTTP library

Requests is an HTTP library, written in Python, for human beings. Basic GET usage:

```
>>> import requests
>>> r = requests.get('https://www.python.org')
>>> r.status_code
200
>>> 'Python is a programming language' in r.content
True
```

... or POST:

```
>>> payload = dict(key1='value1', key2='value2')
>>> r = requests.post('http://httpbin.org/post', data=payload)
>>> print(r.text)
{
  ...
  "form": {
    "key2": "value2",
    "key1": "value1"
  },
  ...
}
```

The other HTTP methods are supported - see *requests.api*. Full documentation is at <http://python-requests.org>.

> **copyright**
>
>> 3. 2015 by Kenneth Reitz.
>
> **license** Apache 2.0, see LICENSE for more details.

INDICES AND TABLES

- *genindex*
- *modindex*
- *search*

Symbols

G

H

I