PyTan Documentation

Release 2.1.0

Jim Olsen

August 31, 2015

CHAPTER

ONE

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

1.1.2 Why it was created

This was created to solve for the following needs:

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

1.1.3 Requirements

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

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

Parametersusername : 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 Parametershttp_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_MAPSmaps a given *loglevel* to respective logger names and their logger levels

pytan.constants.INFO_FORMATdebugformat=False

pytan.constants.DEBUG_FORMATdebugformat=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

Examples

Setup a Handler() object:

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

_add(*obj*, ***kwargs*)

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

Parametersobj : taniumpy.object_types.base.BaseType

•object to add

Returnsadded_obj : taniumpy.object_types.base.BaseType

•full object that was added

_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.

Parameterssensor_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*

sse : bool, optional

•default: True

•True: perform a server side export when getting result data

•False: perform a normal get result data (default for 6.2)

sse_format : str, optional

•default: 'xml_obj'

•format to have server side export report in, one of: {'csv', 'xml', 'xml_obj', 'cef', 0, 1, 2}

leading : str, optional

•default: ''

•used for sse_format 'cef' only, the string to prepend to each row

trailing : str, optional

•default: ''

•used for sse_format 'cef' only, the string to append to each row

polling_secs : int, optional

•default: 5

•Number of seconds to wait in between GetResultInfo loops

•This is passed through to pytan.pollers.QuestionPoller

complete_pct : int/float, optional

•default: 99

•Percentage of mr_tested out of estimated_total to consider the question "done"

•This is passed through to pytan.pollers.QuestionPoller

override_timeout_secs : int, optional

•default: 0

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

•This is passed through to pytan.pollers.QuestionPoller

callbacks : dict, optional

•default: {}

•can be a dict of functions to be run with the key names being the various state changes: 'ProgressChanged', 'AnswersChanged', 'AnswersComplete'

•This is passed through to pytan.pollers.QuestionPoller.run()

Returnsret : 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': '.*'
... }
```

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

Parametersobj : taniumpy.object_types.base.BaseType

•object to pass to self._check_sse_empty_rs

_check_sse_empty_rs(*obj*, *ok_version*, ***kwargs*)
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

Parametersobj : taniumpy.object_types.base.BaseType

•object to get result info for to ensure non-empty answers

ok_version : bool

•if the version currently running is an "ok" version

_check_sse_format_support(*sse_format*, *sse_format_int*, ***kwargs*)
Determines if the export format integer is supported in the server version

Parameterssse_format : str or int

•user supplied export format

sse_format_int : int

•*sse_format* parsed into an int

_check_sse_timing(*ok_version*, ***kwargs*)
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

Parametersok_version : bool

•if the version currently running is an "ok" version

_check_sse_version(***kwargs*)
Validates that the server version supports server side export

_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.

Parameterspackage_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

Returnsret : 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

Notes

- **For 6.2:**
  - We need to add an Action object
  - The Action object should not be in an ActionList
  - Action.start_time must be specified, if it is not specified the action shows up as expired immediately. We default to 1 second from current time if start_seconds_from_now is not passed in
- **For 6.5 / 6.6:**
  - We need to add a SavedAction object, the server creates the Action object for us
  - To emulate what the console does, the SavedAction should be in a SavedActionList
  - Action.start_time does not need to be specified

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}
... }
```

_export_class_BaseType(*obj*, *export_format*, ***kwargs*)

Handles exporting taniumpy.object_types.base.BaseType

Parametersobj : taniumpy.object_types.base.BaseType

•taniumpy object to export

export_format : str

•str of format to perform export in

Returnsresult : str

•results of exporting *obj* into format *export_format*

_export_class_ResultSet(*obj*, *export_format*, ***kwargs*)
Handles exporting taniumpy.object_types.result_set.ResultSet

Parametersobj : taniumpy.object_types.result_set.ResultSet

•taniumpy object to export

export_format : str

•str of format to perform export in

Returnsresult : str

•results of exporting *obj* into format *export_format*

_export_format_csv(*obj*, ***kwargs*)
Handles exporting format: CSV

Parametersobj : taniumpy.object_types.result_set.ResultSet or taniumpy.object_types.base.BaseType

•taniumpy object to export

Returnsresult : str

•results of exporting *obj* into csv format

_export_format_json(*obj*, ***kwargs*)
Handles exporting format: JSON

Parametersobj : taniumpy.object_types.result_set.ResultSet or taniumpy.object_types.base.BaseType

•taniumpy object to export

Returnsresult : str

•results of exporting *obj* into json format

_export_format_xml(*obj*, ***kwargs*)
Handles exporting format: XML

Parametersobj : taniumpy.object_types.result_set.ResultSet or taniumpy.object_types.base.BaseType

•taniumpy object to export

Returnsresult : str

•results of exporting *obj* into XML format

_find(*obj*, ***kwargs*)
Wrapper for interfacing with `taniumpy.session.Session.find()`

Parametersobj : taniumpy.object_types.base.BaseType

•object to find

Returnsfound : taniumpy.object_types.base.BaseType

•full object that was found

_get_multi(*obj_map*, ***kwargs*)
Find multiple item wrapper using `_find()`

Parametersobj_map : dict

•dict containing the map for a given object type

Returnsfound : `taniumpy.object_types.base.BaseType`

•full object that was found

_get_package_def(*d*, ***kwargs*)
Uses `get()` to update a definition with a package object

Parametersd : dict

•dict containing package definition

Returnsd : dict

•dict containing package definitions with package object in 'package_obj'

_get_sensor_defs(*defs*, ***kwargs*)
Uses `get()` to update a definition with a sensor object

Parametersdefs : list of dict

•list of dicts containing sensor definitions

Returnsdefs : list of dict

•list of dicts containing sensor definitions with sensor object in 'sensor_obj'

_get_single(*obj_map*, ***kwargs*)
Find single item wrapper using `_find()`

Parametersobj_map : dict

•dict containing the map for a given object type

Returnsfound : `taniumpy.object_types.base.BaseType`

•full object that was found

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

Returnsdict, 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"

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

Returnsbool

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

•False otherwise

_resolve_sse_format(*sse_format*, ***kwargs*)
Resolves the server side export format the user supplied to an integer for the API

Parametersssse_format : str or int

•user supplied export format

Returnssse_format_int : int

•*sse_format* parsed into an int

_single_find(*obj_map*, *k*, *v*, ***kwargs*)
Wrapper for single item searches interfacing with `taniumpy.session.Session.find()`

Parametersobj_map : dict

•dict containing the map for a given object type

k : str

•attribute name to set to *v*

v : str

•attribute value to set on *k*

Returnsfound : `taniumpy.object_types.base.BaseType`

•full object that was found

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

Parametersv_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

Returnsbool

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

•False otherwise

approve_saved_action(*id*, ***kwargs*)
Approve a saved action

Parametersid : int

•id of saved action to approve

Returnssaved_action_approve_obj : `taniumpy.object_types.saved_action_approval.SavedActio`

•The object containing the return from SavedActionApproval

ask(***kwargs*)
Ask a type of question and get the results back

Parametersqtype : str, optional

•default: 'manual'

•type of question to ask: {'saved', 'manual', '_manual'}

Returnsresult : 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_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()

Parameterssensors : 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

polling_secs : int, optional

•default: 5

•Number of seconds to wait in between GetResultInfo loops

•This is passed through to `pytan.pollers.QuestionPoller`

complete_pct : int/float, optional

•default: 99

•Percentage of mr_tested out of estimated_total to consider the question "done"

•This is passed through to `pytan.pollers.QuestionPoller`

override_timeout_secs : int, optional

•default: 0

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

•This is passed through to `pytan.pollers.QuestionPoller`

callbacks : dict, optional

•default: {}

•can be a dict of functions to be run with the key names being the various state changes: 'ProgressChanged', 'AnswersChanged', 'AnswersComplete'

•This is passed through to `pytan.pollers.QuestionPoller.run()`

Returnsresult : 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=
```

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 De
```

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']
```

ask_parsed(*question_text*, *picker=None*, *get_results=True*, ***kwargs*)

Ask a parsed question as *question_text* and use the index of the parsed results from *picker*

Parametersquestion_text : str

- The question text you want the server to parse into a list of parsed results

picker : int

- default: None
- The index number of the parsed results that correlates to the actual question you wish to run

get_results : bool, optional

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

sse : bool, optional

- default: True
- True: perform a server side export when getting result data
- False: perform a normal get result data (default for 6.2)

sse_format : str, optional

- default: 'xml_obj'
- format to have server side export report in, one of: {'csv', 'xml', 'xml_obj', 'cef', 0, 1, 2}

leading : str, optional

- default: ''
- used for sse_format 'cef' only, the string to prepend to each row

trailing : str, optional

- default: ''
- used for sse_format 'cef' only, the string to append to each row

polling_secs : int, optional

- default: 5

•Number of seconds to wait in between GetResultInfo loops

•This is passed through to `pytan.pollers.QuestionPoller`

complete_pct : int/float, optional

•default: 99

•Percentage of mr_tested out of estimated_total to consider the question "done"

•This is passed through to `pytan.pollers.QuestionPoller`

override_timeout_secs : int, optional

•default: 0

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

•This is passed through to `pytan.pollers.QuestionPoller`

callbacks : dict, optional

•default: {}

•can be a dict of functions to be run with the key names being the various state changes: 'ProgressChanged', 'AnswersChanged', 'AnswersComplete'

•This is passed through to `pytan.pollers.QuestionPoller.run()`

Returnsret : dict, containing:

•*question_object* : `taniumpy.object_types.question.Question`, the actual question 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

Examples

Ask the server to parse 'computer name', but don't pick a choice (will print out a list of choices at critical logging lev

```
>>> v = handler.ask_parsed('computer name')
```

Ask the server to parse 'computer name' and pick index 1 as the question you want to run:

```
>>> v = handler.ask_parsed('computer name', picker=1)
```

ask_saved(*refresh_data=False*, ***kwargs*)

Ask a saved question and get the results back

Parametersid : 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

sse : bool, optional

•default: True

•True: perform a server side export when getting result data

•False: perform a normal get result data (default for 6.2)

sse_format : str, optional

•default: 'xml_obj'

•format to have server side export report in, one of: {'csv', 'xml', 'xml_obj', 'cef', 0, 1, 2}

leading : str, optional

•default: ''

•used for sse_format 'cef' only, the string to prepend to each row

trailing : str, optional

•default: ''

•used for sse_format 'cef' only, the string to append to each row

Returnsret : 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

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

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

Parametersname : 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

Returnsplugin_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

create_from_json(*objtype*, *json_file*, ***kwargs*)

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

Parametersobjtype : str

•Type of object described in *json_file*

json_file : str

•path to JSON file that describes an API object

Returnsret : `taniumpy.object_types.base.BaseType`

•TaniumPy object added to Tanium SOAP Server

See also:

pytan.constants.GET_OBJ_MAPmaps objtype to supported ‘create_json’ types

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

Create a group object

Parametersgroupname : 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

Returnsgroup_obj : `taniumpy.object_types.group.Group`

•TaniumPy object added to Tanium SOAP Server

See also:

pytan.constants.FILTER_MAPSvalid filters for filters

pytan.constants.OPTION_MAPSvalid options for filter_options

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

Parametersname : 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

Returnspackage_obj : taniumpy.object_types.package_spec.PackageSpec

•TaniumPy object added to Tanium SOAP Server

See also:

pytan.constants.FILTER_MAPSvalid filters for verify_filters

pytan.constants.OPTION_MAPSvalid options for verify_filter_options

create_report_file(*contents*, *report_file=None*, ***kwargs*)

Exports a python API object to a file

Parameterscontents : str

•contents to write to *report_file*

report_file : str, optional

•filename to save report as

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*

Returnsreport_path : str

•the full path to the file created with *contents*

create_sensor(***kwargs*)

Create a sensor object

Raisespytan.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_user(*username*, *rolename=[]*, *roleid=[]*, *properties=[]*, ***kwargs*)

Create a user object

Parametersusername : 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

Returnsuser_obj : `taniumpy.object_types.user.User`

•TaniumPy object added to Tanium SOAP Server

create_whitelisted_url(*url*, *regex=False*, *download_seconds=86400*, *properties=[]*, ***kwargs*)

Create a whitelisted url object

Parametersurl : 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

Returnsurl_obj : taniumpy.object_types.white_listed_url.WhiteListedUrl

•TaniumPy object added to Tanium SOAP Server

delete(*objtype*, ***kwargs*)

Delete an object type

Parametersobjtype : 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

Returnsret : dict

•dict containing deploy action object and results from deploy action

See also:

pytan.constants.GET_OBJ_MAPmaps objtype to supported 'search' keys

delete_dashboard(*name*, ***kwargs*)

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

Parametersname : str

•name of dashboard to delete

Returnsplugin_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

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()

Parameterspackage : 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

Returnsret : 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']
```

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

Returnsresult : str

- the contents of exporting *export_format*

See also:

pytan.constants.EXPORT_MAPSmaps 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_to_report_file(*obj*, *export_format='csv'*, ***kwargs*)
Exports a python API object to a file

Parametersobj : 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*

Returnsreport_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

pytan.handler.Handler.create_report_file() method that performs the actual work to write the report file

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.

get(*objtype*, ***kwargs*)
Get an object type

Parametersobjtype : 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

Returnsobj_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 (*objtype*, ***kwargs*)

Get all objects of a type

Parametersobjtype : string

•type of object to get

Returnsobj_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 (*name=''*, ***kwargs*)

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

Parametersname : str, optional

•default: ''

•name of dashboard to get, if empty will return all dashboards

Returnsplugin_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_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*.

Parametersobj : 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

Returnsrd : taniumpy.object_types.result_set.ResultSet

The return of GetResultData for *obj*

get_result_data_sse(*obj*, *sse_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

Parametersobj : taniumpy.object_types.base.BaseType

•object to get result data for

sse_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 sse_format 'cef' only, the string to prepend to each row

trailing : str, optional

•default: ''

•used for sse_format 'cef' only, the string to append to each row

Returnsexport_data : either *str* or taniumpy.object_types.result_set.ResultSet

•If sse_format is one of csv, xml, or cef, export_data will be a *str* containing the contents of the ResultSet in said format

•If sse_format is xml_obj, export_data will be a taniumpy.object_types.result_set.ResultSet

See also:

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

pytan.constants.SSE_RESTRICT_MAP maps sse_format integers to supported platform versions

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

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.

Parametersobj : 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

Returnsri : taniumpy.object_types.result_info.ResultInfo

•The return of GetResultData for *obj*

get_server_version(***kwargs*)
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.

Returnsself.server_version: str

•Version of Tanium Server in string format

parse_query(*question_text*, ***kwargs*)
Ask a parsed question as *question_text* and get a list of parsed results back

Parametersquestion_text : str

•The question text you want the server to parse into a list of parsed results

Returnsparse_job_results : taniumpy.object_types.parse_result_group.ParseResultGroup

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

Parametersobj : taniumpy.object_types.plugin.Plugin

•Plugin object to run

Returnsplugin_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

stop_action(*id*, ***kwargs*)
Stop an action

Parametersid : int

•id of action to stop

Returnsaction_stop_obj : taniumpy.object_types.action_stop.ActionStop

The object containing the ID of the action stop job

xml_to_result_set_obj(*x*, ***kwargs*)

Wraps a Result Set XML from a server side export in the appropriate tags and returns a ResultSet object

Parametersx : str

•str of XML to convert to a ResultSet object

Returnsrs : taniumpy.object_types.result_set.ResultSet

•x converted into a ResultSet object

1.2.2 pytan.exceptions module

Provides exceptions for the pytan module.

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.**DefinitionParserError**

Bases: exceptions.Exception

Exception thrown for errors while parsing definitions from pytan.handler

exception pytan.exceptions.**HandlerError**

Bases: exceptions.Exception

Exception thrown for errors in pytan.handler

exception pytan.exceptions.**HttpError**

Bases: exceptions.Exception

Exception thrown for HTTP errors in pytan.sessions

exception pytan.exceptions.**HumanParserError**

Bases: exceptions.Exception

Exception thrown for errors while parsing human strings from pytan.handler

exception pytan.exceptions.**NotFoundError**

Bases: exceptions.Exception

Exception thrown for Not Found messages from Tanium in pytan.handler

exception pytan.exceptions.**PickerError**

Bases: exceptions.Exception

Exception thrown for picker errors in pytan.handler

exception pytan.exceptions.**PollingError**
Bases: exceptions.Exception

Exception thrown for errors in pytan.polling

exception pytan.exceptions.**PytanHelp**
Bases: exceptions.Exception

Exception thrown when printing out help

exception pytan.exceptions.**RunFalse**
Bases: exceptions.Exception

Exception thrown when run=False from pytan.handler.Handler.deploy_action()

exception pytan.exceptions.**ServerParseError**
Bases: exceptions.Exception

Exception thrown for server parsing errors in pytan.handler

exception pytan.exceptions.**ServerSideExportError**
Bases: exceptions.Exception

Exception thrown for server side export errors in pytan.handler

exception pytan.exceptions.**TimeoutException**
Bases: exceptions.Exception

Exception thrown for timeout errors in pytan.polling

exception pytan.exceptions.**UnsupportedVersionError**
Bases: exceptions.Exception

Exception thrown for version checks in pytan.handler

exception pytan.exceptions.**VersionMismatchError**
Bases: exceptions.Exception

Exception thrown for version_check in pytan.utils

exception pytan.exceptions.**VersionParseError**
Bases: exceptions.Exception

Exception thrown for server version parsing errors in pytan.handler

1.2.3 pytan.sessions module

Session classes for the pytan module.

class pytan.sessions.**Session**(*host*, *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.

Examples

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')
```

ALL_REQUESTS_RESPONSES = []
This list will be updated with each requests response object that was received

AUTH_CONNECT_TIMEOUT_SEC = 5
number of seconds before timing out for a connection while authenticating

AUTH_FAIL_CODES = [401, 403]
List of HTTP response codes that equate to authorization failures

AUTH_RES = 'auth'
The URL to use for authentication requests

AUTH_RESPONSE_TIMEOUT_SEC = 15
number of seconds before timing out for a response while authenticating

BAD_RESPONSE_CMD_PRUNES = ['\n', 'XML Parse Error: ', 'SOAPProcessing Exception: class ', 'ERROR: 400 Bad Re
List of strings to remove from commands in responses that do not match the response in the request

BAD_SERVER_VERSIONS = [None, '', 'Unable to determine', 'Not yet determined']
List of server versions that are not valid

ELEMENT_RE_TXT = '<{0}>(.*?)</{0}>'
regex string to search for an element in XML bodies

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

HTTP_DEBUG = False
print requests package debug or not

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

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

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

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

LAST_REQUESTS_RESPONSE = None
This variable will be updated with the last requests response object that was received

LAST_RESPONSE_INFO = {}
This variable will be updated with the information from the most recent call to _get_response()

RECORD_ALL_REQUESTS = False
Controls whether each requests response object is appended to the self.ALL_REQUESTS_RESPONSES list

REQUESTS_SESSION = <requests.sessions.Session object at 0x10b384d90>
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

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

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

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

SOAP_RES = 'soap'
The URL to use for SOAP requests

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

STATS_LOOP_ENABLED = False
enable the statistics loop thread or not

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

STATS_LOOP_TARGETS = [{'Version': 'Settings/Version'}, {'Active Questions': 'Active Question Cache/Active Question
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

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

_build_body(*command*, *object_list*, *log_options=False*, ***kwargs*)
Utility method for building an XML Request Body

Parameterscommand : 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.

log_options : bool, optional

•default: False

•False: Do not print messages setting attributes in Options from keys in kwargs

•True: Print messages setting attributes in Options from keys in kwargs

Returnsbody : str

•The XML request body created from the string.template self.REQUEST_BODY_TEMPLATE

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

_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.

Parametersheaders : dict

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

Returnsheaders : dict

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

_create_add_object_body(*obj*, ***kwargs*)
Utility method for building an XML Request Body to add an object

Parametersobj : 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.

Returnsobj_body : str

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

_create_delete_object_body(*obj*, ***kwargs*)
Utility method for building an XML Request Body to delete an object

Parametersobj : 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.

Returnsobj_body : str

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

_create_get_object_body(*obj*, ***kwargs*)
Utility method for building an XML Request Body to get an object

Parametersobj : 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.

Returnsobj_body : str

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

_create_get_result_data_body(*obj*, ***kwargs*)
Utility method for building an XML Request Body to get result data for an object

Parametersobj : 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.

Returnsobj_body : str

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

_create_get_result_info_body(*obj*, ***kwargs*)
Utility method for building an XML Request Body to get result info for an object

Parametersobj : 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.

Returnsobj_body : str

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

_create_run_plugin_object_body(*obj*, ***kwargs*)
Utility method for building an XML Request Body to run a plugin

Parametersobj : 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.

Returnsobj_body : str

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

_create_update_object_body(*obj*, ***kwargs*)
Utility method for building an XML Request Body to update an object

Parametersobj : 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.

Returnsobj_body : str

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

_extract_resultxml(*response_body*)
Utility method to get the 'ResultXML' element from an XML body

Parametersresponse_body : str

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

Returnsret : str of ResultXML element

•str if 'export_id' element found in XML

_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(

Parameterstarget : 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

Returnsdict

•label : same as provided in *target* index0 (label)

•result : value resolved from pytan.sessions.Session._resolve_stat_target() for *target* index1 (search_path)

_flatten_server_info(*structure*)
Utility method for flattening the JSON structure for info.json into a more python usable format

Parametersstructure

•dict/tuple/list to flatten

Returnsflattened

•the dict/tuple/list flattened out

_full_url(*url*, ***kwargs*)
Utility method for constructing a full url

Parametersurl : str

•url to use in string

host : str, optional

•default: self.host

•hostname/IP address to use in string

port : str, optional

•default: self.port

•port to use in string

Returnsfull_url : str

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

_get_percentage(*part*, *whole*)
Utility method for getting percentage of part out of whole

Parameterspart: int, float

whole: int, float

Returnsstr : the percentage of part out of whole in 2 decimal places

_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.

Parametersrequest_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

pytan_help : str, optional

•default: ‘’

•help string to add to self.LAST_REQUESTS_RESPONSE.pytan_help

Returnsbody : str

•str containing body of response from server

See also:

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

_http_get(*host*, *port*, *url*, *headers=None*, *connect_timeout=15*, *response_timeout=180*, *debug=False*, *pytan_help=’‘*, ***kwargs*)

This is an HTTP GET method that utilizes the `requests` package.

Parametershost : 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

pytan_help : str, optional

•default: ''

•help string to add to self.LAST_REQUESTS_RESPONSE.pytan_help

perform_xml_clean : bool, optional

•default: False

•False: Do not run the response_body through an XML cleaner

•True: Run the response_body through an XML cleaner before returning it

clean_restricted : bool, optional

•default: True

•True: When XML cleaning the response_body, remove restricted characters as well as invalid characters

•False: When XML cleaning the response_body, remove only invalid characters

log_clean_messages : bool, optional

•default: True

•True: When XML cleaning the response_body, enable logging messages about invalid/restricted matches

•False: When XML cleaning the response_body, disable logging messages about invalid/restricted matches

log_bad_characters : bool, optional

•default: False

•False: When XML cleaning the response_body, disable logging messages about the actual characters that were invalid/restricted

•True: When XML cleaning the response_body, enable logging messages about the actual characters that were invalid/restricted

Returnsbody : str

•str containing body of response from server

_http_post(*host*, *port*, *url*, *body=None*, *headers=None*, *connect_timeout=15*, *response_timeout=180*, *debug=False*, *pytan_help=''*, ***kwargs*)

This is an HTTP POST method that utilizes the `requests` package.

Parametershost : 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

pytan_help : str, optional

- default: ''
- help string to add to self.LAST_REQUESTS_RESPONSE.pytan_help

perform_xml_clean : bool, optional

- default: True
- True: Run the response_body through an XML cleaner before returning it
- False: Do not run the response_body through an XML cleaner

clean_restricted : bool, optional

- default: True
- True: When XML cleaning the response_body, remove restricted characters as well as invalid characters
- False: When XML cleaning the response_body, remove only invalid characters

log_clean_messages : bool, optional

- default: True
- True: When XML cleaning the response_body, enable logging messages about invalid/restricted matches

•False: When XML cleaning the response_body, disable logging messages about invalid/restricted matches

log_bad_characters : bool, optional

•default: False

•False: When XML cleaning the response_body, disable logging messages about the actual characters that were invalid/restricted

•True: When XML cleaning the response_body, enable logging messages about the actual characters that were invalid/restricted

Returnsbody : str

•str containing body of response from server

See also:

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

_regex_body_for_element(*body*, *element*, *fail=True*)

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

Parametersbody : str

•XML to search

element : str

•element name 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*

Returnsret : str

•The first value that matches the regex ELEMENT_RE_TXT with element

Notes

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

_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

Parametersheaders : dict

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

Returnsheaders : dict

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

_resolve_stat_target(*search_path*, *diags*)

Utility method for resolving the value of search_path in info.json and returning the value

Parameterssearch_path : str

•/ seperated search path to find a given value from info.json

diags : dict

•flattened dictionary of info.json diagnostics

Returnsstr

•value resolved from *diags* for *search_path*

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

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

add(*obj*, ***kwargs*)
Creates and sends a AddObject XML Request body from *obj* and parses the response into an appropriate taniumpy object

Parametersobj : taniumpy.object_types.base.BaseType

•object to add

Returnsobj : taniumpy.object_types.base.BaseType

•added object

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

Parametersusername : 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)

pytan_help : str, optional

•default: ‘’

•help string to add to self.LAST_REQUESTS_RESPONSE.pytan_help

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.

`authlog` = <logging.Logger object at 0x10b2d4110>

`bodyhttplog` = <logging.Logger object at 0x10b2d4190>

`delete`(*obj*, ***kwargs*)

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

Parametersobj : `taniumpy.object_types.base.BaseType`

- object to delete

Returnsobj : `taniumpy.object_types.base.BaseType`

- deleted object

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_SEC

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()

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_SEC

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()

find(*obj*, ***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

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

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

get_result_data_sse(*obj*, ***kwargs*)
Creates and sends a GetResultData XML Request body that starts a server side export from *obj* and parses the response for an export_id.

Parameters
obj : taniumpy.object_types.base.BaseType

- object to start server side export

Returnsexport_id : str

•value of export_id element found in response

get_result_info(*obj*, ***kwargs*)

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

Parametersobj : taniumpy.object_types.base.BaseType

•object to get result info for

Returnsobj : taniumpy.object_types.result_info.ResultInfo

•ResultInfo for *obj*

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

Gets the /info.json

Parametersport : 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

Returnsinfo_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)

get_server_stats(***kwargs*)

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

Returnsstr

•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

get_server_version(***kwargs*)
Tries to parse the server version from /info.json

Returnsstr

•str containing server version from /info.json

host = None
host to connect to

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.

Parametersurl : str

•url to fetch on the server

host : str, optional

•default: self.host

•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

pytan_help : str, optional

•default: ‘’

•help string to add to self.LAST_REQUESTS_RESPONSE.pytan_help

Returnsbody : str

•str containing body of response from server

See also:

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

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.

Parametersurl : str, optional

•default: self.SOAP_RES

•url to fetch on the server

host : str, optional

•default: self.host

•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

pytan_help : str, optional

•default: ‘’

•help string to add to self.LAST_REQUESTS_RESPONSE.pytan_help

Returnsbody : str

•str containing body of response from server

See also:

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

`httplog` = <logging.Logger object at 0x10b2d4150>

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

Returnsbool

•True: if self._session_id or self._username and _self.password are set

•False: if not

`logout`(*all_session_ids=False*, ***kwargs*)
Logout a given session_id from Tanium. If not session_id currently set, it will authenticate to get one.

Parametersall_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

pytan_help : str, optional

•default: ‘’

•help string to add to self.LAST_REQUESTS_RESPONSE.pytan_help

`mylog` = <logging.Logger object at 0x10b2d40d0>

`port` = None
port to connect to

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

Parametersobj : `taniumpy.object_types.base.BaseType`

•object to run

Returnsobj : taniumpy.object_types.base.BaseType

•results from running object

save(*obj*, ***kwargs*)
Creates and sends a UpdateObject XML Request body from *obj* and parses the response into an appropriate taniumpy object

Parametersobj : taniumpy.object_types.base.BaseType

•object to save

Returnsobj : taniumpy.object_types.base.BaseType

•saved object

server_version = None
version string of server, will be updated if get_server_version() is called

session_id
Property to fetch the session_id for this object

Returnsself._session_id : str

setup_logging()

statslog = <logging.Logger object at 0x10b2d41d0>

1.2.4 pytan.pollers module

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

class pytan.pollers.**ActionPoller**(*handler*, *obj*, ***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 an action, and fire off events:

•'SeenProgressChanged'

•'SeenAnswersComplete'

•'FinishedProgressChanged'

•'FinishedAnswersComplete'

Parametershandler : 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

•default: 5

•Number of seconds to wait in between GetResultInfo loops

complete_pct : int/float, optional

•default: 100

•Percentage of mr_tested out of estimated_total to consider the question "done"

override_timeout_secs : int, optional

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

ACTION_DONE_KEY = 'success'
key in action_result_map that maps to an action being done

COMPLETE_PCT_DEFAULT = 100
default value for self.complete_pct

EXPIRATION_ATTR = 'expiration_time'
attribute of self.obj that contains the expiration for this object

OBJECT_TYPE
valid type of object that can be passed in as obj to __init__

alias of `Action`

RUNNING_STATUSES = ['active', 'open']
values for status attribute of action object that mean the action is running

_derive_object_info(***kwargs*)
Derive self.object_info from self.obj

_derive_package_spec(***kwargs*)
Get the package_spec attribute for self.obj, then fetch the full package_spec object

_derive_result_map(***kwargs*)
Determine what self.result_map should contain for the various statuses an action can have

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(***kwargs*)
Get the status attribute for self.obj

_derive_stopped_flag(***kwargs*)
Get the stopped_flag attribute for self.obj

_derive_target_group(***kwargs*)
Get the target_group attribute for self.obj, then fetch the full group object

_derive_verify_enabled(***kwargs*)
Determine if this action has verification enabled

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

_post_init(***kwargs*)
Post init class setup

finished_eq_passed_loop(*callbacks={}*, ***kwargs*)
Method to poll Result Info for self.obj until the percentage of 'finished_count' out of 'self.passed_count' is greater than or equal to self.complete_pct

- finished_count is calculated from a full GetResultData call that is parsed into self.action_result_map
- self.passed_count is calculated by the question asked before this method is called. that question has no selects, but has a group that is the same group as the action for this object

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

Parameterscallbacks : dict

•Callbacks should be a dict with any of these members:

–'SeenProgressChanged'

–'SeenAnswersComplete'

–'FinishedProgressChanged'

–'FinishedAnswersComplete'

•Each callback should be a function that accepts:

–'poller': a poller instance

–'pct': a percent complete

–'kwargs': a dict of other args

Notes

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

•Any callback can choose to stop the poller by calling `pytan.poller.QuestionPoller.stop()`

•Polling will be stopped only when one of the callbacks calls the `pytan.poller.QuestionPoller.stop()` method or the answers are complete.

•Any callbacks can call `pytan.poller.QuestionPoller.setPercentCompleteThreshold()` to change what "done" means on the fly

seen_eq_passed_loop (*callbacks={}*, ***kwargs*)
Method to poll Result Info for self.obj until the percentage of 'seen_count' out of 'self.passed_count' is greater than or equal to self.complete_pct

•seen_count is calculated from an aggregate GetResultData

•self.passed_count is calculated by the question asked before this method is called. that question has no selects, but has a group that is the same group as the action for this object

class `pytan.pollers.`**QuestionPoller** (*handler*, *obj*, ***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 a question, and fire off events:

•ProgressChanged

•AnswersChanged

•AnswersComplete

Parametershandler : `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

•default: 5

•Number of seconds to wait in between GetResultInfo loops

complete_pct : int/float, optional

•default: 99

•Percentage of mr_tested out of estimated_total to consider the question "done"

override_timeout_secs : int, optional

•default: 0

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

COMPLETE_PCT_DEFAULT = 99
default value for self.complete_pct

EXPIRATION_ATTR = 'expiration'
attribute of self.obj that contains the expiration for this object

EXPIRY_FALLBACK_SECS = 600
If the EXPIRATION_ATTR of *obj* can't be automatically determined, then this is used as a fallback for timeout - polling will failed after this many seconds if completion not reached

OBJECT_TYPE
valid type of object that can be passed in as obj to __init__

alias of `Question`

OVERRIDE_TIMEOUT_SECS_DEFAULT = 0
default value for self.override_timeout_secs

POLLING_SECS_DEFAULT = 5
default value for self.polling_secs

STR_ATTRS = ['object_info', 'polling_secs', 'override_timeout_secs', 'complete_pct', 'expiration']
Class attributes to include in __str__ output

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

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

Parametersattr : 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

Returnsval : perspective

The value of the attr from self.obj

_derive_expiration(***kwargs*)
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(***kwargs*)
Derive self.object_info from self.obj

_post_init(***kwargs*)
Post init class setup

_refetch_obj(***kwargs*)
Utility method to re-fetch a object

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

_stop = False
Controls whether a run() loop should stop or not

get_result_data(***kwargs*)
Simple utility wrapper around `pytan.handler.Handler.get_result_data()`

get_result_info(***kwargs*)
Simple utility wrapper around `pytan.handler.Handler.get_result_info()`

handler = None
The Handler object for this poller

mylog = <logging.Logger object at 0x10b2d4e50>

obj = None
The object for this poller

passed_eq_est_total_loop(*callbacks={}*, ***kwargs*)
Method to poll Result Info for self.obj until the percentage of 'passed' out of 'estimated_total' is greater than or equal to self.complete_pct

progresslog = <logging.Logger object at 0x10b2d4f50>

resolverlog = <logging.Logger object at 0x10b2d4f90>

result_info = None
This will be updated with the ResultInfo object during run() calls

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

Parameterscallbacks : dict

- **Callbacks should be a dict with any of these members:**
  - 'ProgressChanged'
  - 'AnswersChanged'
  - 'AnswersComplete'
- **Each callback should be a function that accepts:**
  - 'poller': a poller instance
  - 'pct': a percent complete
  - 'kwargs': a dict of other args

Notes

- 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.
- Any callback can call setPercentCompleteThreshold to change what "done" means on the fly

run_callback(*callbacks*, *callback*, *pct*, ***kwargs*)
Utility method to find a callback in callbacks dict and run it

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

Parametersval : int/float

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

setup_logging()
Setup loggers for this object

stop()

class pytan.pollers.**SSEPoller**(*handler*, *export_id*, ***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.

Parametershandler : pytan.handler.Handler

PyTan handler to use for GetResultInfo calls

export_id : str

- ID of server side export

polling_secs : int, optional

- default: 2
- Number of seconds to wait in between status check loops

timeout_secs : int, optional

- default: 600
- timeout in seconds for waiting for status completion, 0 does not time out

POLLING_SECS_DEFAULT = 2
default value for self.polling_secs

STR_ATTRS = ['export_id', 'polling_secs', 'timeout_secs', 'sse_status']
Class attributes to include in __str__ output

TIMEOUT_SECS_DEFAULT = 600
default value for self.timeout_secs

_post_init(***kwargs*)
Post init class setup

export_id = None
The export_id for this poller

get_sse_data(***kwargs*)
Function to get the data of a server side export

Constructs a URL via: export/${export_id}.gz and performs an authenticated HTTP get

get_sse_status(***kwargs*)
Function to get the status of a server side export

Constructs a URL via: export/${export_id}.status and performs an authenticated HTTP get

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

sse_status_has_completed_loop(***kwargs*)
Method to poll the status file for a server side export until it contains 'Completed'

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': {'xml': [], 'json': [], 'csv': [{'valid_list_types': ['str', 'unicode'], 'key': 'h**

Maps a given TaniumPy object to the list of supported export formats for each object type, and the valid optional argume

- key: the optional argument name itself
- valid_types: the valid python types that are allowed to be passed as a value to *key*
- valid_list_types: the valid python types in str format that are allowed to be passed in a list, if list is one of the *valid_types*

pytan.constants.**FILTER_MAPS = [{'operator': 'Less', 'not_flag': 0, 'help': 'Filter for less than VALUE', 'human': ['<', '**

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

- human: a list of human strings that can be used after '*, that*'. Ex: '*, that* `contains value`'
- operator: the filter operator used by the SOAP API when building a filter that matches *human*
- not_flag: the value to set on *not_flag* when building a filter that matches *human*
- pre_value: the prefix to add to the `value` when building a filter
- post_value: the postfix to add to the `value` when building a filter

pytan.constants.**FILTER_RE = ',\\s*that'**
The regex that is used to find filters in a string. Ex: *Sensor1, `that` contains blah*

pytan.constants.**GET_OBJ_MAP = {'user': {'search': ['id'], 'all': 'UserList', 'manual': True, 'multi': None, 'single': 'Use**

Maps an object type from a human friendly string into various aspects:

- single: The `TaniumPy` object used to find singular instances of this object type
- multi: The `TaniumPy` object used to find multiple instances of this object type
- all: The `TaniumPy` object used to find all instances of this object type
- search: The list of attributes that can be used with the Tanium SOAP API for searches

- manual: Whether or not this object type is allowed to do a manual search, that is – allow the user to specify an attribute that is not in search, which will get ALL objects of that type then search for a match based on attribute values for EVERY key/value pair supplied
- delete: Whether or not this object type can be deleted
- create_json: Whether or not this object type can be created by importing from JSON

pytan.constants.**INFO_FORMAT = '%(asctime)s %(levelname)-8s %(name)s: %(message)s'**
Logging format for debugformat=False

pytan.constants.**LOG_LEVEL_MAPS = [(0, {'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: 'IPA**
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 classes and methods used throughout pytan

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*)

pytan.utils.**apply_options_obj**(*options*, *obj*, *dest*)
Updates an object with options

Parametersoptions : 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*)

Returnsobj : 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

Parametersq_filter_defs : list of dict

•List of dict that are question filter definitions

q_option_defs : dict

•dict of question filter options

Returnsgroup_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

Parametersselectlist_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

Returnsadd_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

Parametersproperties : 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

Returnsmetadatalist_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

Parameterskey : 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

Returnsparam_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

Parametersobj : 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

Returnsparam_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

Parameterssensor_defs : list of dict

•List of dict that are sensor definitions

Returnsselect_objlist : taniumpy.object_types.select_list.SelectList

•SelectList object with list of taniumpy.object_types.select.Select built from *sensor_defs*

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

Utility method for getting percentage of whole

Parameterspercent: int, float

whole: int, float

Returnsint : the percentage of whole

pytan.utils.**change_console_format**(*debug=False*)

Changes the logging format for console handler to pytan.constants.DEBUG_FORMAT or pytan.constants.INFO_FORMAT

Parametersdebug : 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.**check_dictkey**(*d*, *key*, *valid_types*, *valid_list_types*)

Yet another method to check a dictionary for a key

Parametersd : 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.**check_for_help**(*kwargs*)

Utility method to check for any help arguments and raise a PytanHelp exception with the appropriate help

Parameterskwargs : dict

•dict of keyword args

pytan.utils.**chk_def_key**(*def_dict*, *key*, *keytypes*, *keysubtypes=None*, *req=False*)

Checks that def_dict has key

Parametersdef_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.**clean_kwargs**(*kwargs*, *keys=None*)

Removes each key from kwargs dict if found

Parameterskwargs : dict

•dict of keyword args

keys : list of str, optional

•default: ['obj', 'pytan_help', 'objtype']

•list of strs of keys to remove from kwargs

Returnsclean_kwargs : dict

•the new dict of kwargs with keys removed

pytan.utils.**copy_obj**(*obj*, *skip_attrs=None*)

Returns a new class of obj with with out any attributes in skip_attrs specified

Parametersobj : 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

Returnsnew_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

Parametersobj : 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']

Returnsnew_obj : taniumpy.object_types.base.BaseType

- Copied object with attributes in skip_attrs skipped

pytan.utils.**datetime_to_timestr**(*dt*)

Get a timestr for *dt*

Parametersdt : datetime.datetime

- datetime object

Returnstimestr: str

- the timestr for *dt* in taniums format

pytan.utils.**dehumanize_package**(*package*)

Turns a package str into a package definition

Parameterspackage : str

- A str that describes a package and optionally a selector and/or parameters

Returnspackage_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

Parametersquestion_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

Returnsquestion_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

Parametersquestion_options : str, list of str

- A str or list of str that describes question options

Returnsquestion_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

Parameterssensors : 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

Returnssensor_defs : list of dict

•list of dict parsed from *sensors*

pytan.utils.**derive_param_default**(*obj_param*)
Derive a parameter default

Parametersobj_param : dict

•parameter dict from TaniumPy object

Returnsdef_val : str

•default value derived from obj_param

pytan.utils.**empty_obj**(*taniumpy_object*)
Validate that a given TaniumPy object is not empty

Parameterstaniumpy_object : taniumpy.object_types.base.BaseType

•object to check if empty

Returnsbool

•True if *taniumpy_object* is considered empty, False otherwise

pytan.utils.**eval_timing**(*c*)
Yet another method to time things – c will be evaluated and timing information will be printed out

pytan.utils.**extract_filter**(*s*)
Extracts a filter from str *s*

Parameterss : str

•A str that may or may not have a filter identified by ', that HUMAN VALUE'

Returnss : 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*

Parameterss : str

•A str that may or may not have options identified by ', opt:name[:value]'

Returnss : 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*

Parameterss : str

•A str that may or may not have parameters identified by {key=value}

Returnss : 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*

Parameterss : 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

Returnss : str

•str *s* without the parsed_selector included

parsed_selector : str

•selector extracted from *s*, or 'name' if none found

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

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

pytan.utils.**get_dict_list_len**(*d*, *keys=[]*, *negate=False*)
Gets the sum of each list in dict *d*

Parametersd : 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

Returnslist_len : int

•sum of lists in *d* that match keys

pytan.utils.**get_filter_obj**(*sensor_def*)
Creates a Filter object from sensor_def

Parameterssensor_def : dict

•dict containing sensor definition

Returnsfilter_obj : taniumpy.object_types.filter.Filter

•Filter object created from *sensor_def*

pytan.utils.**get_kwargs_int**(*key*, *default=None*, ***kwargs*)
Gets key from kwargs and validates it is an int

Parameterskey : 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

Returnsval : int

value from key, or default if supplied

pytan.utils.**get_now**()

Get current time in human friendly format

Returnsstr :

str of current time return from `human_time()`

pytan.utils.**get_obj_map**(*objtype*)

Gets an object map for *objtype*

Parametersobjtype : str

•object type to get object map from in `pytan.constants.GET_OBJ_MAP`

Returnsobj_map : dict

•matching object map for *objtype* from `pytan.constants.GET_OBJ_MAP`

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

Returnsparams : dict

•JSON loaded dict of parameters from *obj*

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

Utility method for getting percentage of part out of whole

Parameterspart: int, float

whole: int, float

Returnsint : the percentage of part out of whole

pytan.utils.**get_q_obj_map**(*qtype*)

Gets an object map for *qtype*

Parametersqtype : str

•question type to get object map from in `pytan.constants.Q_OBJ_MAP`

Returnsobj_map : dict

•matching object map for *qtype* from `pytan.constants.Q_OBJ_MAP`

pytan.utils.**get_taniumpy_obj**(*obj_map*)

Gets a taniumpy object from *obj_map*

Parametersobj_map : str

•str of taniumpy object to fetch

Returnsobj : taniumpy.object_types.base.BaseType

•matching taniumpy object for *obj_map*

pytan.utils.**human_time**(*t*, *tformat='%Y_%m_%d-%H_%M_%S-%Z'*)
Get time in human friendly format

Parameterst : 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

Returnsstr :

•*t* converted to str

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

pytan.utils.**jsonify**(*v*, *indent=2*, *sort_keys=True*)
Turns python object *v* into a pretty printed JSON string

Parametersv : 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

Returnsstr :

•JSON pretty printed string

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

Parametersparameters_json_file : str

•path to JSON file that describes an API object

Returnsobj

•contents of parameters_json_file de-serialized

pytan.utils.**load_taniumpy_from_json**(*json_file*)
Opens a json file and parses it into an taniumpy object

Parametersjson_file : str

•path to JSON file that describes an API object

Returnsobj : taniumpy.object_types.base.BaseType

•TaniumPy object converted from json file

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

Parametersh : Handler object

•Handler object with session object containing last request and response body

pytan.utils.**map_filter**(*filter_str*)

Maps a filter str against constants.FILTER_MAPS

Parametersfilter_str : str

•filter_str str that should be validated

Returnsfilter_attrs : dict

•dict containing mapped filter attributes for SOAP API

pytan.utils.**map_option**(*opt*, *dest*)

Maps an opt str against constants.OPTION_MAPS

Parametersopt : str

•option str that should be validated

dest : list of str

•list of valid destinations (i.e. *filter* or *group*)

Returnsopt_attrs : dict

•dict containing mapped option attributes for SOAP API

pytan.utils.**map_options**(*options*, *dest*)

Maps a list of options using map_option()

Parametersoptions : list of str

•list of str that should be validated

dest : list of str

•list of valid destinations (i.e. *filter* or *group*)

Returnsmapped_options : dict

•dict of all mapped_options

pytan.utils.**parse_defs**(*defname*, *deftypes*, *strconv=None*, *empty_ok=True*, *defs=None*, ***kwargs*)

Parses and validates defs into new_defs

Parametersdefname : 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

Returnsnew_defs : list of dict

•parsed and validated defs

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

Maps columns to values for each row in a plugins sql_response and returns a list of dicts

Parametersp : taniumpy.object_types.plugin.Plugin

•plugin object

Returnsdict

•the columns and result_rows of the sql_response in Plugin object zipped up into a dictionary

pytan.utils.**port_check**(*address*, *port*, *timeout=5*)

Check if *address*:*port* can be reached within *timeout*

Parametersaddress : 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

Returnssocket or False :

•if connection succeeds, the socket object is returned, else False is returned

pytan.utils.**print_log_levels**()

Prints info about each loglevel from pytan.constants.LOG_LEVEL_MAPS

pytan.utils.**remove_logging_handler**(*name='all'*)

Removes a logging handler

Parametersname : str

•name of logging handler to remove. if name == 'all' then all logging handlers are removed

pytan.utils.**seconds_from_now**(*secs=0*, *tz='utc'*)

Get time in Tanium SOAP API format *secs* from now

Parameterssecs : 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

Returnsstr :

•time *secs* from now in Tanium SOAP API format

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

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

Parametersobj : 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

Returnsnew_obj : taniumpy.object_types.base.BaseType

- Shrunken object

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

Parameterst : str

- string to debug print

pytan.utils.**test_app_port**(*host*, *port*)
Validates that *host*:*port* can be reached using port_check()

Parametershost : str

- hostname/ip address to check *port* on

port : int

- port to check on *host*

Raisespytan.exceptions.HandlerError : pytan.exceptions.HandlerError

- if *host*:*port* can not be reached

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

Parameterstimestr : str

- date & time in taniums format

Returnsdatetime.datetime

- the datetime object for the timestr

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

Parameterspackage_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

Parametersq_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

Parameterssensor_defs : list of dict

•list of sensor definitions

pytan.utils.**xml_pretty**(*x*, *pretty=True*, *indent=' '*, ***kwargs*)
Uses xmltodict to pretty print an XML str *x*

Parametersx : str

•XML string to pretty print

Returnsstr :

•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*

Parametersx : str

•XML string to pretty print

Returnsstr :

•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*

Parametersx : str

•XML string to pretty print

Returnsstr :

•The pretty printed string of ResultXML in *x*

1.2.7 pytan.binsupport module

Collection of classes and methods used throughout pytan for command line support

class pytan.binsupport.**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.binsupport.**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*)

class pytan.binsupport.**HistoryConsole**(*locals=None*, *filename='<console>'*, *histfile='/Users/jolsen/.console-history'*)
Bases: code.InteractiveConsole

Class that provides an interactive python console with full auto complete, history, and history file support.

Examples

```
>>> console = pytan.binsupport.HistoryConsole()
```

init_history(*histfile*)

static save_history(*histfile*)

pytan.binsupport.**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.binsupport.**add_file_log**(*logfile*, *debug=False*)
Utility to add a log file from python's logging module

pytan.binsupport.**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.binsupport.**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.binsupport.**csvdictwriter**(*rows_list*, ***kwargs*)
returns the rows_list (list of dicts) as a CSV string

pytan.binsupport.**debug_list**(*debuglist*)
Utility function to print the variables for a list of objects

pytan.binsupport.**debug_obj**(*debugobj*)
Utility function to print the variables for an object

pytan.binsupport.**filter_filename**(*filename*)
Utility to filter a string into a valid filename

pytan.binsupport.**filter_sensors**(*sensors*, *filter_platforms=[]*, *filter_categories=[]*)
Utility to filter a list of sensors for specific platforms and/or categories

pytan.binsupport.**filter_sourced_sensors**(*sensors*)
Utility to filter out all sensors that have a source_id specified (i.e. they are temp sensors created by the API)

pytan.binsupport.**get_all_headers**(*rows_list*)
Utility to get all the keys for a list of dicts

pytan.binsupport.**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.binsupport.**input_prompts**(*args*)
Utility function to prompt for username, password, and host if empty

pytan.binsupport.**introspect**(*obj*, *depth=0*)
Utility function to dump all info about an object

pytan.binsupport.**parse_sensor_platforms**(*sensor*)
Utility to create a list of platforms for a given sensor

pytan.binsupport.**print_obj**(*d*, *indent=0*)
Pretty print a dictionary

pytan.binsupport.**process_ask_manual_args**(*parser*, *handler*, *args*)
Process command line args supplied by user for ask manual

Parameters
parser : argparse.ArgParse
•ArgParse object used to parse *all_args*
handler : pytan.handler.Handler
•Instance of Handler created from command line args
args : args object
•args parsed from *parser*
Returnsresponse
•response from pytan.handler.Handler.ask_manual()

pytan.binsupport.**process_ask_saved_args**(*parser*, *handler*, *args*)
Process command line args supplied by user for ask saved

Parameters
parser : argparse.ArgParse
•ArgParse object used to parse *all_args*
handler : pytan.handler.Handler
•Instance of Handler created from command line args
args : args object
•args parsed from *parser*
Returnsresponse

•response from pytan.handler.Handler.ask_saved()

pytan.binsupport.**process_create_group_args**(*parser*, *handler*, *args*)
Process command line args supplied by user for create group object

Parameters
parser : argparse.ArgParse
•ArgParse object used to parse *all_args*
handler : pytan.handler.Handler
•Instance of Handler created from command line args
args : args object
•args parsed from *parser*
Returns
response : taniumpy.object_types.base.BaseType
•response from pytan.handler.Handler.create_group()

pytan.binsupport.**process_create_json_object_args**(*parser*, *handler*, *obj*, *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
args : args object
•args parsed from *parser*
Returns
response : taniumpy.object_types.base.BaseType
•response from pytan.handler.Handler.create_from_json()

pytan.binsupport.**process_create_package_args**(*parser*, *handler*, *args*)
Process command line args supplied by user for create package object

Parameters
parser : argparse.ArgParse
•ArgParse object used to parse *all_args*
handler : pytan.handler.Handler
•Instance of Handler created from command line args
args : args object
•args parsed from *parser*
Returns
response : taniumpy.object_types.base.BaseType
•response from pytan.handler.Handler.create_package()

pytan.binsupport.**process_create_sensor_args**(*parser*, *handler*, *args*)
Process command line args supplied by user for create sensor object

Parameters
parser : argparse.ArgParse
•ArgParse object used to parse *all_args*

handler : pytan.handler.Handler

•Instance of Handler created from command line args

args : args object

•args parsed from *parser*

Returnsresponse : taniumpy.object_types.base.BaseType

•response from pytan.handler.Handler.create_sensor()

pytan.binsupport.**process_create_user_args**(*parser*, *handler*, *args*)
Process command line args supplied by user for create user object

Parametersparser : argparse.ArgParse

•ArgParse object used to parse *all_args*

handler : pytan.handler.Handler

•Instance of Handler created from command line args

args : args object

•args parsed from *parser*

Returnsresponse : taniumpy.object_types.base.BaseType

•response from pytan.handler.Handler.create_user()

pytan.binsupport.**process_create_whitelisted_url_args**(*parser*, *handler*, *args*)
Process command line args supplied by user for create group object

Parametersparser : argparse.ArgParse

•ArgParse object used to parse *all_args*

handler : pytan.handler.Handler

•Instance of Handler created from command line args

args : args object

•args parsed from *parser*

Returnsresponse : taniumpy.object_types.base.BaseType

•response from pytan.handler.Handler.create_group()

pytan.binsupport.**process_delete_object_args**(*parser*, *handler*, *obj*, *args*)
Process command line args supplied by user for delete object

Parametersparser : 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

args : args object

•args parsed from *parser*

Returnsresponse : taniumpy.object_types.base.BaseType

•response from pytan.handler.Handler.delete()

pytan.binsupport.**process_deploy_action_args**(*parser*, *handler*, *args*)
Process command line args supplied by user for deploy action

Parameters
parser : argparse.ArgParse
•ArgParse object used to parse *all_args*
handler : pytan.handler.Handler
•Instance of Handler created from command line args
args : args object
•args parsed from *parser*
Returnsresponse
•response from pytan.handler.Handler.deploy_action()

pytan.binsupport.**process_get_object_args**(*parser*, *handler*, *obj*, *args*, *report=True*)
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
args : args object
•args parsed from *parser*
Returnsresponse : taniumpy.object_types.base.BaseType
•response from pytan.handler.Handler.get()

pytan.binsupport.**process_get_results_args**(*parser*, *handler*, *args*)
Process command line args supplied by user for getting results

Parameters
parser : argparse.ArgParse
•ArgParse object used to parse *all_args*
handler : pytan.handler.Handler
•Instance of Handler created from command line args
args : args
•args object from parsing *parser*
Returnsreport_path, report_contents : tuple
•results from pytan.handler.Handler.export_to_report_file() on the return of pytan.handler.Handler.get_result_data()

pytan.binsupport.**process_handler_args**(*parser*, *args*)
Process command line args supplied by user for handler

Parameters
parser : argparse.ArgParse
•ArgParse object used to parse *all_args*

args : args

•args parsed from *parser*

Returnsh : pytan.handler.Handler

•Handler object

pytan.binsupport.**process_print_sensors_args**(*parser*, *handler*, *args*)
Process command line args supplied by user for printing sensors

Parametersparser : argparse.ArgParse

•ArgParse object used to parse *all_args*

handler : pytan.handler.Handler

•Instance of Handler created from command line args

args : args object

•args parsed from *parser*

pytan.binsupport.**process_print_server_info_args**(*parser*, *handler*, *args*)
Process command line args supplied by user for printing server info

Parametersparser : argparse.ArgParse

•ArgParse object used to parse *all_args*

handler : pytan.handler.Handler

•Instance of Handler created from command line args

args : args object

•args parsed from *parser*

pytan.binsupport.**process_pytan_shell_args**(*parser*, *handler*, *args*)
Process command line args supplied by user for a python shell

Parametersparser : argparse.ArgParse

•ArgParse object used to parse *all_args*

handler : pytan.handler.Handler

•Instance of Handler created from command line args

args : args object

•args parsed from *parser*

pytan.binsupport.**process_stop_action_args**(*parser*, *handler*, *args*)
Process command line args supplied by user for getting results

Parametersparser : argparse.ArgParse

•ArgParse object used to parse *all_args*

handler : pytan.handler.Handler

•Instance of Handler created from command line args

args : args

•args object from parsing *parser*

Returnsreport_path, report_contents : tuple

•results from pytan.handler.Handler.export_to_report_file() on the return of pytan.handler.Handler.get_result_data()

pytan.binsupport.**process_tsat_args**(*parser*, *handler*, *args*)
Process command line args supplied by user for tsat

Parametersparser : argparse.ArgParse

•ArgParse object used to parse *all_args*

handler : pytan.handler.Handler

•Instance of Handler created from command line args

args : args object

•args parsed from *parser*

pytan.binsupport.**remove_file_log**(*logfile*)
Utility to remove a log file from python's logging module

pytan.binsupport.**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.binsupport.**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.binsupport.**setup_create_group_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 create a group.

pytan.binsupport.**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.binsupport.**setup_create_package_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 create a package.

pytan.binsupport.**setup_create_sensor_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 create a sensor.

pytan.binsupport.**setup_create_user_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 create a user.

pytan.binsupport.**setup_create_whitelisted_url_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 create a whitelisted_url.

pytan.binsupport.**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.binsupport.**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.binsupport.**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.binsupport.**setup_get_results_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.binsupport.**setup_parent_parser**(*doc*)
Method to setup the base pytan.utils.CustomArgParse class for command line scripts using pytan.utils.setup_parser() and return a parser object for adding arguments to

pytan.binsupport.**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

pytan.binsupport.**setup_print_sensors_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 print server info.

pytan.binsupport.**setup_print_server_info_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 print sensor info.

pytan.binsupport.**setup_pytan_shell_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 create a python shell.

pytan.binsupport.**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.binsupport.**setup_tsat_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 objects.

pytan.binsupport.**version_check**(*reqver*)
Allows scripts using pytan to validate the version of the script aginst the version of pytan

Parametersreqver : str

•string containing version number to check against Exception

RaisesVersionMismatchError : Exception

•if pytan.__version__ is not greater or equal to *reqver*

1.2.8 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 0x7fd8be349a00>**
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 0x7fd8be349da0>**
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*

Parameterstext : 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

Returnsstr, 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_clean_messages=True*, *log_bad_characters=False*, *replacement=None*, ***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_clean_messages : bool, optional

- default: True
- log messages using python logging or not

log_bad_characters : bool, optional

- default: False
- log bad character matches or not

Returns

str

- the cleaned version of *s*

1.2.9 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

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

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

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

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

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

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

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

test_invalid1()

test_invalid2()

test_valid1()

test_valid2()

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

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

test_invalid1()

test_valid1()

test_valid2()

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

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

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

test_invalid1()

test_invalid2()

test_invalid3()

test_invalid4()

test_valid1()

test_valid2()

test_valid3()

test_valid4()

1.2.10 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

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_optio

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

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*)

1.2.11 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

```
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
```

Output from Python Code

```
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

```
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["refresh_data"] = True
kwargs["qtype"] = u'saved'
kwargs["name"] = u'Installed Applications'

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

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

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

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

```
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
```

```
ARDAgent,nothing,Not Uninstallable,3.8.4
OBEXAgent,nothing,Not Uninstallable,4.3.5
PressAndHold,nothing,Not Uninstallable,1.2
..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

```
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["qtype"] = u'saved'
kwargs["name"] = u'Installed Applications'

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

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

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

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

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

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

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

Output from Python Code

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

Type of response:  <type 'dict'>

Pretty print of response:
{'poller_object': None,
 'poller_success': None,
 'question_object': <taniumpy.object_types.question.Question object at 0x10a80a150>,
 'question_results': <taniumpy.object_types.result_set.ResultSet object at 0x10a810c10>,
 'saved_question_object': <taniumpy.object_types.saved_question.SavedQuestion object at 0x10a810a50>

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

CSV Results of response:
Name,Silent Uninstall String,Uninstallable,Version
Image Capture Extension,nothing,Not Uninstallable,10.2
Dictation,nothing,Not Uninstallable,1.6.1
Wish,nothing,Not Uninstallable,8.5.9
Uninstall AnyConnect,nothing,Not Uninstallable,3.1.08009
Time Machine,nothing,Not Uninstallable,1.3
```

```
AppleGraphicsWarning,nothing,Not Uninstallable,2.3.0
soagent,nothing,Not Uninstallable,7.0
Feedback Assistant,nothing,Not Uninstallable,4.1.3
AinuIM,nothing,Not Uninstallable,1.0
vpndownloader,nothing,Not Uninstallable,3.1.08009
Pass Viewer,nothing,Not Uninstallable,1.0
ARDAgent,nothing,Not Uninstallable,3.8.4
OBEXAgent,nothing,Not Uninstallable,4.3.5
PressAndHold,nothing,Not Uninstallable,1.2
..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

```
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["qtype"] = u'saved'
kwargs["name"] = [u'Installed Applications']

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

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

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

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

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

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

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

Output from Python Code

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

Type of response:  <type 'dict'>

Pretty print of response:
{'poller_object': None,
 'poller_success': None,
 'question_object': <taniumpy.object_types.question.Question object at 0x10a613d90>,
 'question_results': <taniumpy.object_types.result_set.ResultSet object at 0x10a6c0410>,
 'saved_question_object': <taniumpy.object_types.saved_question.SavedQuestion object at 0x10a808b10>

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

CSV Results of response:
```

```
Name,Silent Uninstall String,Uninstallable,Version
Image Capture Extension,nothing,Not Uninstallable,10.2
Dictation,nothing,Not Uninstallable,1.6.1
Wish,nothing,Not Uninstallable,8.5.9
Uninstall AnyConnect,nothing,Not Uninstallable,3.1.08009
Time Machine,nothing,Not Uninstallable,1.3
AppleGraphicsWarning,nothing,Not Uninstallable,2.3.0
soagent,nothing,Not Uninstallable,7.0
Feedback Assistant,nothing,Not Uninstallable,4.1.3
AinuIM,nothing,Not Uninstallable,1.0
vpndownloader,nothing,Not Uninstallable,3.1.08009
Pass Viewer,nothing,Not Uninstallable,1.0
ARDAgent,nothing,Not Uninstallable,3.8.4
OBEXAgent,nothing,Not Uninstallable,4.3.5
PressAndHold,nothing,Not Uninstallable,1.2
..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
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'Computer Name', u'Installed Applications']
kwargs["qtype"] = u'manual'

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

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

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

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

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

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

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

Output from Python Code

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

```
2015-08-07 19:37:55,255 DEBUG    pytan.handler.QuestionPoller: ID 1280: Object Info resolved to Ques
2015-08-07 19:37:55,260 DEBUG    pytan.handler.QuestionPoller: ID 1280: Progress: Tested: 0, Passed:
2015-08-07 19:37:55,260 DEBUG    pytan.handler.QuestionPoller: ID 1280: Timing: Started: 2015-08-07
2015-08-07 19:37:55,260 INFO     pytan.handler.QuestionPoller: ID 1280: Progress Changed 0% (0 of 2)
2015-08-07 19:38:00,263 DEBUG    pytan.handler.QuestionPoller: ID 1280: Progress: Tested: 1, Passed:
2015-08-07 19:38:00,263 DEBUG    pytan.handler.QuestionPoller: ID 1280: Timing: Started: 2015-08-07
2015-08-07 19:38:00,264 INFO     pytan.handler.QuestionPoller: ID 1280: Progress Changed 50% (1 of 2
2015-08-07 19:38:05,271 DEBUG    pytan.handler.QuestionPoller: ID 1280: Progress: Tested: 1, Passed:
2015-08-07 19:38:05,271 DEBUG    pytan.handler.QuestionPoller: ID 1280: Timing: Started: 2015-08-07
2015-08-07 19:38:10,277 DEBUG    pytan.handler.QuestionPoller: ID 1280: Progress: Tested: 2, Passed:
2015-08-07 19:38:10,277 DEBUG    pytan.handler.QuestionPoller: ID 1280: Timing: Started: 2015-08-07
2015-08-07 19:38:10,277 INFO     pytan.handler.QuestionPoller: ID 1280: Progress Changed 100% (2 of
2015-08-07 19:38:10,277 INFO     pytan.handler.QuestionPoller: ID 1280: Reached Threshold of 99% (2

Type of response:  <type 'dict'>

Pretty print of response:
{'poller_object': <pytan.pollers.QuestionPoller object at 0x10a808810>,
 'poller_success': True,
 'question_object': <taniumpy.object_types.question.Question object at 0x10a810650>,
 'question_results': <taniumpy.object_types.result_set.ResultSet object at 0x10a6c0410>}

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

CSV Results of response:
Computer Name,Name,Silent Uninstall String,Uninstallable,Version
Casus-Belli.local,"Image Capture Extension
Dictation
Wish
Uninstall AnyConnect
Time Machine
AppleGraphicsWarning
soagent
Feedback Assistant
AinuIM
vpndownloader
Pass Viewer
ARDAgent
OBEXAgent
PressAndHold
..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

```
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'Computer Name'
kwargs["qtype"] = u'manual'

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

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

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

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

```

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

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

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

Output from Python Code

```
Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2015-08-07 19:38:10,340 DEBUG    pytan.handler.QuestionPoller: ID 1281: id resolved to 1281
2015-08-07 19:38:10,340 DEBUG    pytan.handler.QuestionPoller: ID 1281: expiration resolved to 2015-
2015-08-07 19:38:10,340 DEBUG    pytan.handler.QuestionPoller: ID 1281: query_text resolved to Get C
2015-08-07 19:38:10,340 DEBUG    pytan.handler.QuestionPoller: ID 1281: id resolved to 1281
2015-08-07 19:38:10,340 DEBUG    pytan.handler.QuestionPoller: ID 1281: Object Info resolved to Ques
2015-08-07 19:38:10,343 DEBUG    pytan.handler.QuestionPoller: ID 1281: Progress: Tested: 0, Passed:
2015-08-07 19:38:10,343 DEBUG    pytan.handler.QuestionPoller: ID 1281: Timing: Started: 2015-08-07
2015-08-07 19:38:10,343 INFO     pytan.handler.QuestionPoller: ID 1281: Progress Changed 0% (0 of 2)
2015-08-07 19:38:15,351 DEBUG    pytan.handler.QuestionPoller: ID 1281: Progress: Tested: 1, Passed:
2015-08-07 19:38:15,351 DEBUG    pytan.handler.QuestionPoller: ID 1281: Timing: Started: 2015-08-07
2015-08-07 19:38:15,351 INFO     pytan.handler.QuestionPoller: ID 1281: Progress Changed 50% (1 of 2
2015-08-07 19:38:20,357 DEBUG    pytan.handler.QuestionPoller: ID 1281: Progress: Tested: 2, Passed:
2015-08-07 19:38:20,357 DEBUG    pytan.handler.QuestionPoller: ID 1281: Timing: Started: 2015-08-07
2015-08-07 19:38:20,357 INFO     pytan.handler.QuestionPoller: ID 1281: Progress Changed 100% (2 of
2015-08-07 19:38:20,357 INFO     pytan.handler.QuestionPoller: ID 1281: Reached Threshold of 99% (2

Type of response:  <type 'dict'>

Pretty print of response:
{'poller_object': <pytan.pollers.QuestionPoller object at 0x10a7ecb90>,
 'poller_success': True,
 'question_object': <taniumpy.object_types.question.Question object at 0x10a808290>,
 'question_results': <taniumpy.object_types.result_set.ResultSet object at 0x10a6133d0>}

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

CSV Results of response:
Computer Name
Casus-Belli.local
JTANIUM1.localdomain
```

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

```
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
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 19:38:20,411 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
```

```
Time Machine
AppleGraphicsWarning
soagent
Feedback Assistant
AinuIM
vpndownloader
Pass Viewer
ARDAgent
OBEXAgent
PressAndHold
..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

```
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'Folder Name Search with RegEx Match{dirname=Program Files,regex=Microsoft.*}'
kwargs["qtype"] = u'manual'

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

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

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

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

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

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

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

Output from Python Code

```
Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2015-08-07 19:38:25,510 DEBUG    pytan.handler.QuestionPoller: ID 1283: id resolved to 1283
2015-08-07 19:38:25,510 DEBUG    pytan.handler.QuestionPoller: ID 1283: expiration resolved to 2015-
2015-08-07 19:38:25,510 DEBUG    pytan.handler.QuestionPoller: ID 1283: query_text resolved to Get F
2015-08-07 19:38:25,510 DEBUG    pytan.handler.QuestionPoller: ID 1283: id resolved to 1283
2015-08-07 19:38:25,510 DEBUG    pytan.handler.QuestionPoller: ID 1283: Object Info resolved to Ques
2015-08-07 19:38:25,513 DEBUG    pytan.handler.QuestionPoller: ID 1283: Progress: Tested: 0, Passed:
2015-08-07 19:38:25,513 DEBUG    pytan.handler.QuestionPoller: ID 1283: Timing: Started: 2015-08-07
2015-08-07 19:38:25,513 INFO     pytan.handler.QuestionPoller: ID 1283: Progress Changed 0% (0 of 2)
2015-08-07 19:38:30,521 DEBUG    pytan.handler.QuestionPoller: ID 1283: Progress: Tested: 0, Passed:
```

```
2015-08-07 19:38:30,521 DEBUG    pytan.handler.QuestionPoller: ID 1283: Timing: Started: 2015-08-07
2015-08-07 19:38:35,526 DEBUG    pytan.handler.QuestionPoller: ID 1283: Progress: Tested: 0, Passed:
2015-08-07 19:38:35,526 DEBUG    pytan.handler.QuestionPoller: ID 1283: Timing: Started: 2015-08-07
2015-08-07 19:38:40,532 DEBUG    pytan.handler.QuestionPoller: ID 1283: Progress: Tested: 0, Passed:
2015-08-07 19:38:40,532 DEBUG    pytan.handler.QuestionPoller: ID 1283: Timing: Started: 2015-08-07
2015-08-07 19:38:45,536 DEBUG    pytan.handler.QuestionPoller: ID 1283: Progress: Tested: 0, Passed:
2015-08-07 19:38:45,536 DEBUG    pytan.handler.QuestionPoller: ID 1283: Timing: Started: 2015-08-07
2015-08-07 19:38:50,539 DEBUG    pytan.handler.QuestionPoller: ID 1283: Progress: Tested: 0, Passed:
2015-08-07 19:38:50,539 DEBUG    pytan.handler.QuestionPoller: ID 1283: Timing: Started: 2015-08-07
2015-08-07 19:38:55,543 DEBUG    pytan.handler.QuestionPoller: ID 1283: Progress: Tested: 0, Passed:
2015-08-07 19:38:55,543 DEBUG    pytan.handler.QuestionPoller: ID 1283: Timing: Started: 2015-08-07
2015-08-07 19:39:00,547 DEBUG    pytan.handler.QuestionPoller: ID 1283: Progress: Tested: 0, Passed:
2015-08-07 19:39:00,547 DEBUG    pytan.handler.QuestionPoller: ID 1283: Timing: Started: 2015-08-07
2015-08-07 19:39:05,554 DEBUG    pytan.handler.QuestionPoller: ID 1283: Progress: Tested: 0, Passed:
2015-08-07 19:39:05,554 DEBUG    pytan.handler.QuestionPoller: ID 1283: Timing: Started: 2015-08-07
2015-08-07 19:39:10,558 DEBUG    pytan.handler.QuestionPoller: ID 1283: Progress: Tested: 0, Passed:
2015-08-07 19:39:10,558 DEBUG    pytan.handler.QuestionPoller: ID 1283: Timing: Started: 2015-08-07
2015-08-07 19:39:15,561 DEBUG    pytan.handler.QuestionPoller: ID 1283: Progress: Tested: 0, Passed:
2015-08-07 19:39:15,561 DEBUG    pytan.handler.QuestionPoller: ID 1283: Timing: Started: 2015-08-07
2015-08-07 19:39:20,566 DEBUG    pytan.handler.QuestionPoller: ID 1283: Progress: Tested: 0, Passed:
2015-08-07 19:39:20,566 DEBUG    pytan.handler.QuestionPoller: ID 1283: Timing: Started: 2015-08-07
2015-08-07 19:39:25,571 DEBUG    pytan.handler.QuestionPoller: ID 1283: Progress: Tested: 0, Passed:
2015-08-07 19:39:25,571 DEBUG    pytan.handler.QuestionPoller: ID 1283: Timing: Started: 2015-08-07
2015-08-07 19:39:30,577 DEBUG    pytan.handler.QuestionPoller: ID 1283: Progress: Tested: 0, Passed:
2015-08-07 19:39:30,577 DEBUG    pytan.handler.QuestionPoller: ID 1283: Timing: Started: 2015-08-07
2015-08-07 19:39:35,581 DEBUG    pytan.handler.QuestionPoller: ID 1283: Progress: Tested: 0, Passed:
2015-08-07 19:39:35,581 DEBUG    pytan.handler.QuestionPoller: ID 1283: Timing: Started: 2015-08-07
2015-08-07 19:39:40,585 DEBUG    pytan.handler.QuestionPoller: ID 1283: Progress: Tested: 0, Passed:
2015-08-07 19:39:40,585 DEBUG    pytan.handler.QuestionPoller: ID 1283: Timing: Started: 2015-08-07
2015-08-07 19:39:45,588 DEBUG    pytan.handler.QuestionPoller: ID 1283: Progress: Tested: 0, Passed:
2015-08-07 19:39:45,588 DEBUG    pytan.handler.QuestionPoller: ID 1283: Timing: Started: 2015-08-07
2015-08-07 19:39:50,592 DEBUG    pytan.handler.QuestionPoller: ID 1283: Progress: Tested: 0, Passed:
2015-08-07 19:39:50,592 DEBUG    pytan.handler.QuestionPoller: ID 1283: Timing: Started: 2015-08-07
2015-08-07 19:39:55,597 DEBUG    pytan.handler.QuestionPoller: ID 1283: Progress: Tested: 0, Passed:
2015-08-07 19:39:55,597 DEBUG    pytan.handler.QuestionPoller: ID 1283: Timing: Started: 2015-08-07
2015-08-07 19:40:00,603 DEBUG    pytan.handler.QuestionPoller: ID 1283: Progress: Tested: 0, Passed:
2015-08-07 19:40:00,603 DEBUG    pytan.handler.QuestionPoller: ID 1283: Timing: Started: 2015-08-07
2015-08-07 19:40:05,612 DEBUG    pytan.handler.QuestionPoller: ID 1283: Progress: Tested: 0, Passed:
2015-08-07 19:40:05,613 DEBUG    pytan.handler.QuestionPoller: ID 1283: Timing: Started: 2015-08-07
2015-08-07 19:40:10,618 DEBUG    pytan.handler.QuestionPoller: ID 1283: Progress: Tested: 1, Passed:
2015-08-07 19:40:10,618 DEBUG    pytan.handler.QuestionPoller: ID 1283: Timing: Started: 2015-08-07
2015-08-07 19:40:10,618 INFO     pytan.handler.QuestionPoller: ID 1283: Progress Changed 50% (1 of 2
2015-08-07 19:40:15,626 DEBUG    pytan.handler.QuestionPoller: ID 1283: Progress: Tested: 1, Passed:
2015-08-07 19:40:15,626 DEBUG    pytan.handler.QuestionPoller: ID 1283: Timing: Started: 2015-08-07
2015-08-07 19:40:20,631 DEBUG    pytan.handler.QuestionPoller: ID 1283: Progress: Tested: 1, Passed:
2015-08-07 19:40:20,631 DEBUG    pytan.handler.QuestionPoller: ID 1283: Timing: Started: 2015-08-07
2015-08-07 19:40:25,635 DEBUG    pytan.handler.QuestionPoller: ID 1283: Progress: Tested: 1, Passed:
2015-08-07 19:40:25,635 DEBUG    pytan.handler.QuestionPoller: ID 1283: Timing: Started: 2015-08-07
2015-08-07 19:40:30,639 DEBUG    pytan.handler.QuestionPoller: ID 1283: Progress: Tested: 2, Passed:
2015-08-07 19:40:30,639 DEBUG    pytan.handler.QuestionPoller: ID 1283: Timing: Started: 2015-08-07
2015-08-07 19:40:30,639 INFO     pytan.handler.QuestionPoller: ID 1283: Progress Changed 100% (2 of
2015-08-07 19:40:30,639 INFO     pytan.handler.QuestionPoller: ID 1283: Reached Threshold of 99% (2

Type of response:  <type 'dict'>

Pretty print of response:
{'poller_object': <pytan.pollers.QuestionPoller object at 0x10a613090>,
 'poller_success': True,
```

```
 'question_object': <taniumpy.object_types.question.Question object at 0x10a613cd0>,
 'question_results': <taniumpy.object_types.result_set.ResultSet object at 0x10a810650>}

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

CSV Results of response:
"Folder Name Search with RegEx Match[Program Files, , No, No, Microsoft.*]"
C:\Program Files\VMware\VMware Tools\plugins\vmsvc
C:\Program Files\Common Files\Microsoft Shared\VS7Debug
C:\Program Files\Tanium\Tanium Server\http\taniumjs\sensor-query\src
C:\Program Files\Microsoft SQL Server\110\LocalDB\Binn\Resources\1033
C:\Program Files\Tanium\Tanium Server\http\tux\spin\src
C:\Program Files\Tanium\Tanium Server\http\taniumjs\archived-question\src
C:\Program Files\Tanium\Tanium Module Server\plugins\content
C:\Program Files\Tanium\Tanium Server\http\libraries\kendoui\styles\Moonlight
C:\Program Files\Common Files\VMware\Drivers\vmci\sockets\include
C:\Program Files\Tanium\Tanium Server\http\taniumjs\plugin
C:\Program Files\Common Files\Microsoft Shared\ink\ar-SA
C:\Program Files\Tanium\Tanium Server\plugins\console\WorkbenchesManager
C:\Program Files\Tanium\Tanium Module Server\logs
C:\Program Files\Common Files\SpeechEngines\Microsoft
..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
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'Folder Name Search with RegEx Match{dirname=Program Files,regex=Microsoft.*}'
 u'Computer Name']
kwargs["qtype"] = u'manual'

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

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

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

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

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

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

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

```
print out
```

Output from Python Code

```
Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2015-08-07 19:40:30,754 DEBUG    pytan.handler.QuestionPoller: ID 1284: id resolved to 1284
2015-08-07 19:40:30,754 DEBUG    pytan.handler.QuestionPoller: ID 1284: expiration resolved to 2015-
2015-08-07 19:40:30,754 DEBUG    pytan.handler.QuestionPoller: ID 1284: query_text resolved to Get F
2015-08-07 19:40:30,754 DEBUG    pytan.handler.QuestionPoller: ID 1284: id resolved to 1284
2015-08-07 19:40:30,754 DEBUG    pytan.handler.QuestionPoller: ID 1284: Object Info resolved to Ques
2015-08-07 19:40:30,757 DEBUG    pytan.handler.QuestionPoller: ID 1284: Progress: Tested: 0, Passed:
2015-08-07 19:40:30,757 DEBUG    pytan.handler.QuestionPoller: ID 1284: Timing: Started: 2015-08-07
2015-08-07 19:40:30,757 INFO     pytan.handler.QuestionPoller: ID 1284: Progress Changed 0% (0 of 2)
2015-08-07 19:40:35,761 DEBUG    pytan.handler.QuestionPoller: ID 1284: Progress: Tested: 0, Passed:
2015-08-07 19:40:35,761 DEBUG    pytan.handler.QuestionPoller: ID 1284: Timing: Started: 2015-08-07
2015-08-07 19:40:40,766 DEBUG    pytan.handler.QuestionPoller: ID 1284: Progress: Tested: 0, Passed:
2015-08-07 19:40:40,766 DEBUG    pytan.handler.QuestionPoller: ID 1284: Timing: Started: 2015-08-07
2015-08-07 19:40:45,773 DEBUG    pytan.handler.QuestionPoller: ID 1284: Progress: Tested: 0, Passed:
2015-08-07 19:40:45,773 DEBUG    pytan.handler.QuestionPoller: ID 1284: Timing: Started: 2015-08-07
2015-08-07 19:40:50,777 DEBUG    pytan.handler.QuestionPoller: ID 1284: Progress: Tested: 0, Passed:
2015-08-07 19:40:50,777 DEBUG    pytan.handler.QuestionPoller: ID 1284: Timing: Started: 2015-08-07
2015-08-07 19:40:55,782 DEBUG    pytan.handler.QuestionPoller: ID 1284: Progress: Tested: 0, Passed:
2015-08-07 19:40:55,782 DEBUG    pytan.handler.QuestionPoller: ID 1284: Timing: Started: 2015-08-07
2015-08-07 19:41:00,790 DEBUG    pytan.handler.QuestionPoller: ID 1284: Progress: Tested: 0, Passed:
2015-08-07 19:41:00,790 DEBUG    pytan.handler.QuestionPoller: ID 1284: Timing: Started: 2015-08-07
2015-08-07 19:41:05,798 DEBUG    pytan.handler.QuestionPoller: ID 1284: Progress: Tested: 0, Passed:
2015-08-07 19:41:05,798 DEBUG    pytan.handler.QuestionPoller: ID 1284: Timing: Started: 2015-08-07
2015-08-07 19:41:10,805 DEBUG    pytan.handler.QuestionPoller: ID 1284: Progress: Tested: 0, Passed:
2015-08-07 19:41:10,805 DEBUG    pytan.handler.QuestionPoller: ID 1284: Timing: Started: 2015-08-07
2015-08-07 19:41:15,809 DEBUG    pytan.handler.QuestionPoller: ID 1284: Progress: Tested: 0, Passed:
2015-08-07 19:41:15,809 DEBUG    pytan.handler.QuestionPoller: ID 1284: Timing: Started: 2015-08-07
2015-08-07 19:41:20,813 DEBUG    pytan.handler.QuestionPoller: ID 1284: Progress: Tested: 0, Passed:
2015-08-07 19:41:20,813 DEBUG    pytan.handler.QuestionPoller: ID 1284: Timing: Started: 2015-08-07
2015-08-07 19:41:25,817 DEBUG    pytan.handler.QuestionPoller: ID 1284: Progress: Tested: 2, Passed:
2015-08-07 19:41:25,817 DEBUG    pytan.handler.QuestionPoller: ID 1284: Timing: Started: 2015-08-07
2015-08-07 19:41:25,817 INFO     pytan.handler.QuestionPoller: ID 1284: Progress Changed 100% (2 of
2015-08-07 19:41:25,817 INFO     pytan.handler.QuestionPoller: ID 1284: Reached Threshold of 99% (2

Type of response:  <type 'dict'>

Pretty print of response:
{'poller_object': <pytan.pollers.QuestionPoller object at 0x10a614b50>,
 'poller_success': True,
 'question_object': <taniumpy.object_types.question.Question object at 0x10a5f51d0>,
 'question_results': <taniumpy.object_types.result_set.ResultSet object at 0x10a614f10>}

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

CSV Results of response:
Computer Name,"Folder Name Search with RegEx Match[Program Files, , No, No, Microsoft.*]"
Casus-Belli.local,Windows Only
JTANIUM1.localdomain,"C:\Program Files\VMware\VMware Tools\plugins\vmsvc
C:\Program Files\Common Files\Microsoft Shared\VS7Debug
C:\Program Files\Tanium\Tanium Server\http\taniumjs\sensor-query\src
C:\Program Files\Microsoft SQL Server\110\LocalDB\Binn\Resources\1033
C:\Program Files\Tanium\Tanium Server\http\tux\spin\src
```

```
C:\Program Files\Tanium\Tanium Server\http\taniumjs\archived-question\src
C:\Program Files\Tanium\Tanium Module Server\plugins\content
C:\Program Files\Tanium\Tanium Server\http\libraries\kendoui\styles\Moonlight
C:\Program Files\Common Files\VMware\Drivers\vmci\sockets\include
C:\Program Files\Tanium\Tanium Server\http\taniumjs\plugin
C:\Program Files\Common Files\Microsoft Shared\ink\ar-SA
C:\Program Files\Tanium\Tanium Server\plugins\console\WorkbenchesManager
C:\Program Files\Tanium\Tanium Module Server\logs
..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
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'Computer Name{fake=Dweedle}'
kwargs["qtype"] = u'manual'

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

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

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

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

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

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

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

Output from Python Code

```
Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2015-08-07 19:41:25,887 DEBUG    pytan.handler.QuestionPoller: ID 1286: id resolved to 1286
2015-08-07 19:41:25,887 DEBUG    pytan.handler.QuestionPoller: ID 1286: expiration resolved to 2015-
2015-08-07 19:41:25,887 DEBUG    pytan.handler.QuestionPoller: ID 1286: query_text resolved to Get C
2015-08-07 19:41:25,887 DEBUG    pytan.handler.QuestionPoller: ID 1286: id resolved to 1286
2015-08-07 19:41:25,887 DEBUG    pytan.handler.QuestionPoller: ID 1286: Object Info resolved to Ques
2015-08-07 19:41:25,892 DEBUG    pytan.handler.QuestionPoller: ID 1286: Progress: Tested: 0, Passed:
2015-08-07 19:41:25,892 DEBUG    pytan.handler.QuestionPoller: ID 1286: Timing: Started: 2015-08-07
2015-08-07 19:41:25,892 INFO     pytan.handler.QuestionPoller: ID 1286: Progress Changed 0% (0 of 2)
2015-08-07 19:41:30,900 DEBUG    pytan.handler.QuestionPoller: ID 1286: Progress: Tested: 0, Passed:
2015-08-07 19:41:30,900 DEBUG    pytan.handler.QuestionPoller: ID 1286: Timing: Started: 2015-08-07
2015-08-07 19:41:35,905 DEBUG    pytan.handler.QuestionPoller: ID 1286: Progress: Tested: 0, Passed:
2015-08-07 19:41:35,905 DEBUG    pytan.handler.QuestionPoller: ID 1286: Timing: Started: 2015-08-07
2015-08-07 19:41:40,908 DEBUG    pytan.handler.QuestionPoller: ID 1286: Progress: Tested: 0, Passed:
```

```
2015-08-07 19:41:40,908 DEBUG    pytan.handler.QuestionPoller: ID 1286: Timing: Started: 2015-08-07
2015-08-07 19:41:45,915 DEBUG    pytan.handler.QuestionPoller: ID 1286: Progress: Tested: 0, Passed:
2015-08-07 19:41:45,915 DEBUG    pytan.handler.QuestionPoller: ID 1286: Timing: Started: 2015-08-07
2015-08-07 19:41:50,919 DEBUG    pytan.handler.QuestionPoller: ID 1286: Progress: Tested: 0, Passed:
2015-08-07 19:41:50,919 DEBUG    pytan.handler.QuestionPoller: ID 1286: Timing: Started: 2015-08-07
2015-08-07 19:41:55,923 DEBUG    pytan.handler.QuestionPoller: ID 1286: Progress: Tested: 1, Passed:
2015-08-07 19:41:55,924 DEBUG    pytan.handler.QuestionPoller: ID 1286: Timing: Started: 2015-08-07
2015-08-07 19:41:55,924 INFO     pytan.handler.QuestionPoller: ID 1286: Progress Changed 50% (1 of 2
2015-08-07 19:42:00,928 DEBUG    pytan.handler.QuestionPoller: ID 1286: Progress: Tested: 1, Passed:
2015-08-07 19:42:00,929 DEBUG    pytan.handler.QuestionPoller: ID 1286: Timing: Started: 2015-08-07
2015-08-07 19:42:05,933 DEBUG    pytan.handler.QuestionPoller: ID 1286: Progress: Tested: 2, Passed:
2015-08-07 19:42:05,933 DEBUG    pytan.handler.QuestionPoller: ID 1286: Timing: Started: 2015-08-07
2015-08-07 19:42:05,933 INFO     pytan.handler.QuestionPoller: ID 1286: Progress Changed 100% (2 of
2015-08-07 19:42:05,933 INFO     pytan.handler.QuestionPoller: ID 1286: Reached Threshold of 99% (2

Type of response:  <type 'dict'>

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

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

CSV Results of response:
Computer Name[Dweedle]
[no results]
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)
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'Folder Name Search with RegEx Match'
kwargs["qtype"] = u'manual'

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

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

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

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

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

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

print ""
print "CSV Results of response: "
out = out.getvalue()
```

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

Output from Python Code

```
Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2015-08-07 19:42:06,009 DEBUG    pytan.handler.QuestionPoller: ID 1288: id resolved to 1288
2015-08-07 19:42:06,010 DEBUG    pytan.handler.QuestionPoller: ID 1288: expiration resolved to 2015-
2015-08-07 19:42:06,010 DEBUG    pytan.handler.QuestionPoller: ID 1288: query_text resolved to Get F
2015-08-07 19:42:06,010 DEBUG    pytan.handler.QuestionPoller: ID 1288: id resolved to 1288
2015-08-07 19:42:06,010 DEBUG    pytan.handler.QuestionPoller: ID 1288: Object Info resolved to Ques
2015-08-07 19:42:06,013 DEBUG    pytan.handler.QuestionPoller: ID 1288: Progress: Tested: 0, Passed:
2015-08-07 19:42:06,013 DEBUG    pytan.handler.QuestionPoller: ID 1288: Timing: Started: 2015-08-07
2015-08-07 19:42:06,013 INFO     pytan.handler.QuestionPoller: ID 1288: Progress Changed 0% (0 of 2)
2015-08-07 19:42:11,021 DEBUG    pytan.handler.QuestionPoller: ID 1288: Progress: Tested: 0, Passed:
2015-08-07 19:42:11,021 DEBUG    pytan.handler.QuestionPoller: ID 1288: Timing: Started: 2015-08-07
2015-08-07 19:42:16,025 DEBUG    pytan.handler.QuestionPoller: ID 1288: Progress: Tested: 0, Passed:
2015-08-07 19:42:16,025 DEBUG    pytan.handler.QuestionPoller: ID 1288: Timing: Started: 2015-08-07
2015-08-07 19:42:21,032 DEBUG    pytan.handler.QuestionPoller: ID 1288: Progress: Tested: 0, Passed:
2015-08-07 19:42:21,032 DEBUG    pytan.handler.QuestionPoller: ID 1288: Timing: Started: 2015-08-07
2015-08-07 19:42:26,037 DEBUG    pytan.handler.QuestionPoller: ID 1288: Progress: Tested: 2, Passed:
2015-08-07 19:42:26,037 DEBUG    pytan.handler.QuestionPoller: ID 1288: Timing: Started: 2015-08-07
2015-08-07 19:42:26,038 INFO     pytan.handler.QuestionPoller: ID 1288: Progress Changed 100% (2 of
2015-08-07 19:42:26,038 INFO     pytan.handler.QuestionPoller: ID 1288: Reached Threshold of 99% (2

Type of response:  <type 'dict'>

Pretty print of response:
{'poller_object': <pytan.pollers.QuestionPoller object at 0x10a6147d0>,
 'poller_success': True,
 'question_object': <taniumpy.object_types.question.Question object at 0x10a613790>,
 'question_results': <taniumpy.object_types.result_set.ResultSet object at 0x10a5f5190>}

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

CSV Results of response:
Count,"Folder Name Search with RegEx Match[, , No, No]"
24705,[too many results]
1,C:\Windows\winsxs\amd64_microsoft-windows-s..structure.resources_31bf3856ad364e35_6.1.7600.16385_e
1,C:\Windows\winsxs\x86_microsoft-windows-e..-host-authenticator_31bf3856ad364e35_6.1.7601.17514_non
1,C:\Windows\winsxs\amd64_microsoft-windows-ocspsvc_31bf3856ad364e35_6.1.7601.22807_none_3bfeae72930
1,C:\Windows\winsxs\amd64_microsoft-windows-c..ityclient.resources_31bf3856ad364e35_6.1.7601.22865_e
1,C:\Windows\assembly\NativeImages_v2.0.50727_64\System.Xml
1,C:\Windows\winsxs\amd64_microsoft-windows-winsetupui_31bf3856ad364e35_6.1.7601.18804_none_bd3cf1bb
1,C:\Windows\winsxs\amd64_microsoft-windows-scripting.resources_31bf3856ad364e35_6.1.7600.16385_en-u
1,C:\Windows\winsxs\x86_microsoft-windows-mlang.resources_31bf3856ad364e35_6.1.7600.16385_ru-ru_cf3a
1,C:\Windows\winsxs\x86_microsoft-windows-minkernelapinamespace_31bf3856ad364e35_6.1.7601.21728_none
1,C:\Users\Jim Olsen\AppData\Local\Google
1,C:\Windows\winsxs\x86_microsoft-windows-e..nt-client.resources_31bf3856ad364e35_6.1.7600.16385_en-
1,C:\Windows\winsxs\amd64_microsoft-windows-d..e-eashared-kjshared_31bf3856ad364e35_6.1.7600.16385_n
1,C:\Windows\assembly\NativeImages_v4.0.30319_32\RadLangSvc
..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

```
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'Folder Name Search with RegEx Match{dirname=Program Files,regex=Microsoft.*},
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
```

```
Type of response:  <type 'dict'>

Pretty print of response:
{'poller_object': <pytan.pollers.QuestionPoller object at 0x10a614f50>,
 'poller_success': True,
 'question_object': <taniumpy.object_types.question.Question object at 0x10a5f5190>,
 'question_results': <taniumpy.object_types.result_set.ResultSet object at 0x10a615c10>}

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

CSV Results of response:
"Folder Name Search with RegEx Match[Program Files, , No, No, Microsoft.*]"
[no results]
C:\Program Files\Common Files\Microsoft Shared\VS7Debug
C:\Program Files\Common Files\Microsoft Shared\ink\ar-SA
C:\Program Files\Common Files\Microsoft Shared\ink\ru-RU
C:\Program Files\Common Files\Microsoft Shared\ink\fsdefinitions\keypad
C:\Program Files\Common Files\Microsoft Shared\ink
C:\Program Files\Common Files\Microsoft Shared\ink\sv-SE
C:\Program Files\Common Files\Microsoft Shared\ink\uk-UA
C:\Program Files\Common Files\Microsoft Shared\ink\sl-SI
C:\Program Files\Common Files\Microsoft Shared\ink\hu-HU
C:\Program Files\Common Files\Microsoft Shared\ink\zh-TW
C:\Program Files\Common Files\Microsoft Shared\ink\zh-CN
C:\Program Files\Common Files\Microsoft Shared\ink\fi-FI
C:\Program Files\Common Files\Microsoft Shared
..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
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, opt:max_data_age:3600, opt:value_type
kwargs["qtype"] = u'manual'

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

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

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

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

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

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

print ""
```

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

Output from Python Code

```
Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2015-08-07 19:42:56,259 DEBUG    pytan.handler.QuestionPoller: ID 1290: id resolved to 1290
2015-08-07 19:42:56,259 DEBUG    pytan.handler.QuestionPoller: ID 1290: expiration resolved to 2015-
2015-08-07 19:42:56,259 DEBUG    pytan.handler.QuestionPoller: ID 1290: query_text resolved to Get C
2015-08-07 19:42:56,259 DEBUG    pytan.handler.QuestionPoller: ID 1290: id resolved to 1290
2015-08-07 19:42:56,259 DEBUG    pytan.handler.QuestionPoller: ID 1290: Object Info resolved to Ques
2015-08-07 19:42:56,262 DEBUG    pytan.handler.QuestionPoller: ID 1290: Progress: Tested: 0, Passed:
2015-08-07 19:42:56,262 DEBUG    pytan.handler.QuestionPoller: ID 1290: Timing: Started: 2015-08-07
2015-08-07 19:42:56,262 INFO     pytan.handler.QuestionPoller: ID 1290: Progress Changed 0% (0 of 2)
2015-08-07 19:43:01,266 DEBUG    pytan.handler.QuestionPoller: ID 1290: Progress: Tested: 1, Passed:
2015-08-07 19:43:01,266 DEBUG    pytan.handler.QuestionPoller: ID 1290: Timing: Started: 2015-08-07
2015-08-07 19:43:01,266 INFO     pytan.handler.QuestionPoller: ID 1290: Progress Changed 50% (1 of 2
2015-08-07 19:43:06,271 DEBUG    pytan.handler.QuestionPoller: ID 1290: Progress: Tested: 2, Passed:
2015-08-07 19:43:06,271 DEBUG    pytan.handler.QuestionPoller: ID 1290: Timing: Started: 2015-08-07
2015-08-07 19:43:06,271 INFO     pytan.handler.QuestionPoller: ID 1290: Progress Changed 100% (2 of
2015-08-07 19:43:06,271 INFO     pytan.handler.QuestionPoller: ID 1290: Reached Threshold of 99% (2

Type of response:  <type 'dict'>

Pretty print of response:
{'poller_object': <pytan.pollers.QuestionPoller object at 0x10a5f5f90>,
 'poller_success': True,
 'question_object': <taniumpy.object_types.question.Question object at 0x10a613b10>,
 'question_results': <taniumpy.object_types.result_set.ResultSet object at 0x10a5b9cd0>}

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

CSV Results of response:
Operating System
[no results]
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

```
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)

print ""
print "Pretty print of response:"
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 C
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

```
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'Folder Name Search with RegEx Match{dirname=Program Files,regex=Microsoft.*},
```

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

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

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

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

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

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

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

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

Output from Python Code

```
Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2015-08-07 19:43:16,405 DEBUG    pytan.handler.QuestionPoller: ID 1294: id resolved to 1294
2015-08-07 19:43:16,405 DEBUG    pytan.handler.QuestionPoller: ID 1294: expiration resolved to 2015-
2015-08-07 19:43:16,405 DEBUG    pytan.handler.QuestionPoller: ID 1294: query_text resolved to Get F
2015-08-07 19:43:16,405 DEBUG    pytan.handler.QuestionPoller: ID 1294: id resolved to 1294
2015-08-07 19:43:16,405 DEBUG    pytan.handler.QuestionPoller: ID 1294: Object Info resolved to Ques
2015-08-07 19:43:16,408 DEBUG    pytan.handler.QuestionPoller: ID 1294: Progress: Tested: 0, Passed:
2015-08-07 19:43:16,408 DEBUG    pytan.handler.QuestionPoller: ID 1294: Timing: Started: 2015-08-07
2015-08-07 19:43:16,408 INFO     pytan.handler.QuestionPoller: ID 1294: Progress Changed 0% (0 of 2)
2015-08-07 19:43:21,414 DEBUG    pytan.handler.QuestionPoller: ID 1294: Progress: Tested: 0, Passed:
2015-08-07 19:43:21,414 DEBUG    pytan.handler.QuestionPoller: ID 1294: Timing: Started: 2015-08-07
2015-08-07 19:43:26,420 DEBUG    pytan.handler.QuestionPoller: ID 1294: Progress: Tested: 0, Passed:
2015-08-07 19:43:26,420 DEBUG    pytan.handler.QuestionPoller: ID 1294: Timing: Started: 2015-08-07
2015-08-07 19:43:31,424 DEBUG    pytan.handler.QuestionPoller: ID 1294: Progress: Tested: 0, Passed:
2015-08-07 19:43:31,424 DEBUG    pytan.handler.QuestionPoller: ID 1294: Timing: Started: 2015-08-07
2015-08-07 19:43:36,428 DEBUG    pytan.handler.QuestionPoller: ID 1294: Progress: Tested: 1, Passed:
2015-08-07 19:43:36,428 DEBUG    pytan.handler.QuestionPoller: ID 1294: Timing: Started: 2015-08-07
2015-08-07 19:43:36,428 INFO     pytan.handler.QuestionPoller: ID 1294: Progress Changed 50% (1 of 2
2015-08-07 19:43:41,432 DEBUG    pytan.handler.QuestionPoller: ID 1294: Progress: Tested: 1, Passed:
2015-08-07 19:43:41,432 DEBUG    pytan.handler.QuestionPoller: ID 1294: Timing: Started: 2015-08-07
2015-08-07 19:43:46,441 DEBUG    pytan.handler.QuestionPoller: ID 1294: Progress: Tested: 1, Passed:
2015-08-07 19:43:46,442 DEBUG    pytan.handler.QuestionPoller: ID 1294: Timing: Started: 2015-08-07
2015-08-07 19:43:51,449 DEBUG    pytan.handler.QuestionPoller: ID 1294: Progress: Tested: 2, Passed:
```

```
2015-08-07 19:43:51,449 DEBUG    pytan.handler.QuestionPoller: ID 1294: Timing: Started: 2015-08-07
2015-08-07 19:43:51,449 INFO     pytan.handler.QuestionPoller: ID 1294: Progress Changed 100% (2 of
2015-08-07 19:43:51,449 INFO     pytan.handler.QuestionPoller: ID 1294: Reached Threshold of 99% (2

Type of response:  <type 'dict'>

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

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

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

```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, opt:match_all_values, opt:ignore_case
kwargs["qtype"] = u'manual'

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

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

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

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

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

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

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

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

Output from Python Code

```
Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2015-08-07 19:43:51,504 DEBUG    pytan.handler.QuestionPoller: ID 1295: id resolved to 1295
2015-08-07 19:43:51,504 DEBUG    pytan.handler.QuestionPoller: ID 1295: expiration resolved to 2015-
2015-08-07 19:43:51,504 DEBUG    pytan.handler.QuestionPoller: ID 1295: query_text resolved to Get C
2015-08-07 19:43:51,504 DEBUG    pytan.handler.QuestionPoller: ID 1295: id resolved to 1295
2015-08-07 19:43:51,504 DEBUG    pytan.handler.QuestionPoller: ID 1295: Object Info resolved to Ques
2015-08-07 19:43:51,508 DEBUG    pytan.handler.QuestionPoller: ID 1295: Progress: Tested: 0, Passed:
2015-08-07 19:43:51,508 DEBUG    pytan.handler.QuestionPoller: ID 1295: Timing: Started: 2015-08-07
2015-08-07 19:43:51,508 INFO     pytan.handler.QuestionPoller: ID 1295: Progress Changed 0% (0 of 2)
2015-08-07 19:43:56,512 DEBUG    pytan.handler.QuestionPoller: ID 1295: Progress: Tested: 0, Passed:
2015-08-07 19:43:56,512 DEBUG    pytan.handler.QuestionPoller: ID 1295: Timing: Started: 2015-08-07
2015-08-07 19:44:01,520 DEBUG    pytan.handler.QuestionPoller: ID 1295: Progress: Tested: 2, Passed:
2015-08-07 19:44:01,520 DEBUG    pytan.handler.QuestionPoller: ID 1295: Timing: Started: 2015-08-07
2015-08-07 19:44:01,520 INFO     pytan.handler.QuestionPoller: ID 1295: Progress Changed 100% (2 of
2015-08-07 19:44:01,520 INFO     pytan.handler.QuestionPoller: ID 1295: Reached Threshold of 99% (2

Type of response:  <type 'dict'>

Pretty print of response:
{'poller_object': <pytan.pollers.QuestionPoller object at 0x10a5b98d0>,
 'poller_success': True,
 'question_object': <taniumpy.object_types.question.Question object at 0x10a5b9810>,
 'question_results': <taniumpy.object_types.result_set.ResultSet object at 0x10a615710>}

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

CSV Results of response:
Operating System
[no results]
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

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

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

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

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

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

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

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

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

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

Output from Python Code

```
Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2015-08-07 19:44:01,651 DEBUG    pytan.handler.QuestionPoller: ID 1296: id resolved to 1296
2015-08-07 19:44:01,651 DEBUG    pytan.handler.QuestionPoller: ID 1296: expiration resolved to 2015-
2015-08-07 19:44:01,651 DEBUG    pytan.handler.QuestionPoller: ID 1296: query_text resolved to Get C
2015-08-07 19:44:01,651 DEBUG    pytan.handler.QuestionPoller: ID 1296: id resolved to 1296
2015-08-07 19:44:01,651 DEBUG    pytan.handler.QuestionPoller: ID 1296: Object Info resolved to Ques
2015-08-07 19:44:01,655 DEBUG    pytan.handler.QuestionPoller: ID 1296: Progress: Tested: 0, Passed:
2015-08-07 19:44:01,655 DEBUG    pytan.handler.QuestionPoller: ID 1296: Timing: Started: 2015-08-07
2015-08-07 19:44:01,655 INFO     pytan.handler.QuestionPoller: ID 1296: Progress Changed 0% (0 of 2)
2015-08-07 19:44:06,659 DEBUG    pytan.handler.QuestionPoller: ID 1296: Progress: Tested: 0, Passed:
2015-08-07 19:44:06,659 DEBUG    pytan.handler.QuestionPoller: ID 1296: Timing: Started: 2015-08-07
2015-08-07 19:44:11,666 DEBUG    pytan.handler.QuestionPoller: ID 1296: Progress: Tested: 0, Passed:
2015-08-07 19:44:11,667 DEBUG    pytan.handler.QuestionPoller: ID 1296: Timing: Started: 2015-08-07
2015-08-07 19:44:16,670 DEBUG    pytan.handler.QuestionPoller: ID 1296: Progress: Tested: 0, Passed:
2015-08-07 19:44:16,670 DEBUG    pytan.handler.QuestionPoller: ID 1296: Timing: Started: 2015-08-07
2015-08-07 19:44:21,677 DEBUG    pytan.handler.QuestionPoller: ID 1296: Progress: Tested: 1, Passed:
2015-08-07 19:44:21,677 DEBUG    pytan.handler.QuestionPoller: ID 1296: Timing: Started: 2015-08-07
2015-08-07 19:44:21,677 INFO     pytan.handler.QuestionPoller: ID 1296: Progress Changed 50% (1 of 2
2015-08-07 19:44:26,687 DEBUG    pytan.handler.QuestionPoller: ID 1296: Progress: Tested: 1, Passed:
2015-08-07 19:44:26,687 DEBUG    pytan.handler.QuestionPoller: ID 1296: Timing: Started: 2015-08-07
2015-08-07 19:44:31,691 DEBUG    pytan.handler.QuestionPoller: ID 1296: Progress: Tested: 2, Passed:
2015-08-07 19:44:31,692 DEBUG    pytan.handler.QuestionPoller: ID 1296: Timing: Started: 2015-08-07
2015-08-07 19:44:31,692 INFO     pytan.handler.QuestionPoller: ID 1296: Progress Changed 100% (2 of
2015-08-07 19:44:31,692 INFO     pytan.handler.QuestionPoller: ID 1296: Reached Threshold of 99% (2

Type of response:  <type 'dict'>
```

```

Pretty print of response:
{'poller_object': <pytan.pollers.QuestionPoller object at 0x10a5c9690>,
 'poller_success': True,
 'question_object': <taniumpy.object_types.question.Question object at 0x10a5e1610>,
 'question_results': <taniumpy.object_types.result_set.ResultSet object at 0x10a615510>}

Equivalent Question if it were to be asked in the Tanium Console:
Get Computer Name and Folder Name Search with RegEx Match[Program Files, , No, No, Microsoft.*, test

CSV Results of response:
Computer Name,"Folder Name Search with RegEx Match[Program Files, , No, No, Microsoft.*, test]"
Casus-Belli.local,[no results]
JTANIUM1.localdomain,"C:\Program Files\Common Files\Microsoft Shared\VS7Debug
C:\Program Files\Common Files\Microsoft Shared\ink\ar-SA
C:\Program Files\Common Files\Microsoft Shared\ink\ru-RU
C:\Program Files\Common Files\Microsoft Shared\ink\fsdefinitions\keypad
C:\Program Files\Common Files\Microsoft Shared\ink
C:\Program Files\Common Files\Microsoft Shared\ink\sv-SE
C:\Program Files\Common Files\Microsoft Shared\ink\uk-UA
C:\Program Files\Common Files\Microsoft Shared\ink\sl-SI
C:\Program Files\Common Files\Microsoft Shared\ink\hu-HU
C:\Program Files\Common Files\Microsoft Shared\ink\zh-TW
C:\Program Files\Common Files\Microsoft Shared\ink\zh-CN
C:\Program Files\Common Files\Microsoft Shared\ink\fi-FI
C:\Program Files\Common Files\Microsoft Shared
..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
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'Installed Applications, that regex match:.*Google (Search|Chrome).*'
kwargs["sensors"] = [u'Computer Name',
 u'Last Logged In User',
 u'Installed Applications, that regex match:.*Google (Search|Chrome).*']
kwargs["question_options"] = [u'ignore_case', u'or']
kwargs["qtype"] = u'manual'

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

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

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

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

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

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

print ""
print "CSV Results of response: "
out = out.getvalue()
if len(out.splitlines()) > 15:
```

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

Output from Python Code

```
Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2015-08-07 19:44:31,762 DEBUG    pytan.handler.QuestionPoller: ID 1297: id resolved to 1297
2015-08-07 19:44:31,763 DEBUG    pytan.handler.QuestionPoller: ID 1297: expiration resolved to 2015-
2015-08-07 19:44:31,763 DEBUG    pytan.handler.QuestionPoller: ID 1297: query_text resolved to Get C
2015-08-07 19:44:31,763 DEBUG    pytan.handler.QuestionPoller: ID 1297: id resolved to 1297
2015-08-07 19:44:31,763 DEBUG    pytan.handler.QuestionPoller: ID 1297: Object Info resolved to Ques
2015-08-07 19:44:31,766 DEBUG    pytan.handler.QuestionPoller: ID 1297: Progress: Tested: 0, Passed:
2015-08-07 19:44:31,766 DEBUG    pytan.handler.QuestionPoller: ID 1297: Timing: Started: 2015-08-07
2015-08-07 19:44:31,766 INFO     pytan.handler.QuestionPoller: ID 1297: Progress Changed 0% (0 of 2)
2015-08-07 19:44:36,774 DEBUG    pytan.handler.QuestionPoller: ID 1297: Progress: Tested: 0, Passed:
2015-08-07 19:44:36,774 DEBUG    pytan.handler.QuestionPoller: ID 1297: Timing: Started: 2015-08-07
2015-08-07 19:44:41,779 DEBUG    pytan.handler.QuestionPoller: ID 1297: Progress: Tested: 1, Passed:
2015-08-07 19:44:41,779 DEBUG    pytan.handler.QuestionPoller: ID 1297: Timing: Started: 2015-08-07
2015-08-07 19:44:41,779 INFO     pytan.handler.QuestionPoller: ID 1297: Progress Changed 50% (1 of 2
2015-08-07 19:44:46,783 DEBUG    pytan.handler.QuestionPoller: ID 1297: Progress: Tested: 2, Passed:
2015-08-07 19:44:46,783 DEBUG    pytan.handler.QuestionPoller: ID 1297: Timing: Started: 2015-08-07
2015-08-07 19:44:46,783 INFO     pytan.handler.QuestionPoller: ID 1297: Progress Changed 100% (2 of
2015-08-07 19:44:46,783 INFO     pytan.handler.QuestionPoller: ID 1297: Reached Threshold of 99% (2

Type of response:  <type 'dict'>

Pretty print of response:
{'poller_object': <pytan.pollers.QuestionPoller object at 0x10a615950>,
 'poller_success': True,
 'question_object': <taniumpy.object_types.question.Question object at 0x10a5e1bd0>,
 'question_results': <taniumpy.object_types.result_set.ResultSet object at 0x10a5e1d90>}

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

CSV Results of response:
Computer Name,Last Logged In User,Name,Silent Uninstall String,Uninstallable,Version
JTANIUM1.localdomain,Uninitialized - waiting for login,Google Chrome,"""C:\Program Files (x86)\Googl
Casus-Belli.local,jolsen,"Google Search
Google Search
Google Chrome","nothing
nothing
nothing","Not Uninstallable
Not Uninstallable
Not Uninstallable","42.0.2311.90
41.0.2272.104
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

```
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_filter_defs"] = [{u'filter': {u'not_flag': 0,
              u'operator': u'RegexMatch',
              u'value': u'.*Windows.*'},
```

```
  u'name': u'Operating System'}]
kwargs["sensor_defs"] = [u'Computer Name',
 {u'filter': {u'not_flag': 0,
              u'operator': u'RegexMatch',
              u'value': u'.*Shared.*'},
  u'name': u'Folder Name Search with RegEx Match',
  u'options': {u'ignore_case_flag': 0,
               u'max_age_seconds': 3600,
               u'value_type': u'string'},
  u'params': {u'dirname': u'Program Files'}}]
kwargs["question_option_defs"] = {u'and_flag': 0, u'ignore_case_flag': 0, u'max_age_seconds': 3600}
kwargs["qtype"] = u'_manual'

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

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

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

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

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

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

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

Output from Python Code

```
Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2015-08-07 19:44:46,869 DEBUG    pytan.handler.QuestionPoller: ID 1298: id resolved to 1298
2015-08-07 19:44:46,869 DEBUG    pytan.handler.QuestionPoller: ID 1298: expiration resolved to 2015-
2015-08-07 19:44:46,869 DEBUG    pytan.handler.QuestionPoller: ID 1298: query_text resolved to Get C
2015-08-07 19:44:46,869 DEBUG    pytan.handler.QuestionPoller: ID 1298: id resolved to 1298
2015-08-07 19:44:46,869 DEBUG    pytan.handler.QuestionPoller: ID 1298: Object Info resolved to Ques
2015-08-07 19:44:46,872 DEBUG    pytan.handler.QuestionPoller: ID 1298: Progress: Tested: 0, Passed:
2015-08-07 19:44:46,873 DEBUG    pytan.handler.QuestionPoller: ID 1298: Timing: Started: 2015-08-07
2015-08-07 19:44:46,873 INFO     pytan.handler.QuestionPoller: ID 1298: Progress Changed 0% (0 of 2)
2015-08-07 19:44:51,877 DEBUG    pytan.handler.QuestionPoller: ID 1298: Progress: Tested: 1, Passed:
2015-08-07 19:44:51,877 DEBUG    pytan.handler.QuestionPoller: ID 1298: Timing: Started: 2015-08-07
2015-08-07 19:44:51,877 INFO     pytan.handler.QuestionPoller: ID 1298: Progress Changed 50% (1 of 2
```

```
2015-08-07 19:44:56,881 DEBUG    pytan.handler.QuestionPoller: ID 1298: Progress: Tested: 1, Passed:
2015-08-07 19:44:56,881 DEBUG    pytan.handler.QuestionPoller: ID 1298: Timing: Started: 2015-08-07
2015-08-07 19:45:01,885 DEBUG    pytan.handler.QuestionPoller: ID 1298: Progress: Tested: 1, Passed:
2015-08-07 19:45:01,885 DEBUG    pytan.handler.QuestionPoller: ID 1298: Timing: Started: 2015-08-07
2015-08-07 19:45:06,890 DEBUG    pytan.handler.QuestionPoller: ID 1298: Progress: Tested: 2, Passed:
2015-08-07 19:45:06,890 DEBUG    pytan.handler.QuestionPoller: ID 1298: Timing: Started: 2015-08-07
2015-08-07 19:45:06,890 INFO     pytan.handler.QuestionPoller: ID 1298: Progress Changed 100% (2 of
2015-08-07 19:45:06,890 INFO     pytan.handler.QuestionPoller: ID 1298: Reached Threshold of 99% (2

Type of response:  <type 'dict'>

Pretty print of response:
{'poller_object': <pytan.pollers.QuestionPoller object at 0x10a5c9c90>,
 'poller_success': True,
 'question_object': <taniumpy.object_types.question.Question object at 0x10a5b98d0>,
 'question_results': <taniumpy.object_types.result_set.ResultSet object at 0x10a5e1410>}

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

CSV Results of response:
Computer Name,"Folder Name Search with RegEx Match[Program Files, , No, No]"
JTANIUM1.localdomain,"C:\Program Files\Common Files\Microsoft Shared\VS7Debug
C:\Program Files\Common Files\Microsoft Shared\ink\ar-SA
C:\Program Files\Common Files\Microsoft Shared\ink\ru-RU
C:\Program Files\Common Files\Microsoft Shared\ink\fsdefinitions\keypad
C:\Program Files\Common Files\Microsoft Shared\ink
C:\Program Files\Common Files\Microsoft Shared\ink\sv-SE
C:\Program Files\Common Files\Microsoft Shared\ink\uk-UA
C:\Program Files\Common Files\Microsoft Shared\ink\sl-SI
C:\Program Files\Common Files\Microsoft Shared\ink\hu-HU
C:\Program Files\Common Files\Microsoft Shared\ink\zh-TW
C:\Program Files\Common Files\Microsoft Shared\ink\zh-CN
C:\Program Files\Common Files\Microsoft Shared\ink\fi-FI
C:\Program Files\Common Files\Microsoft Shared
C:\Program Files\Common Files\Microsoft Shared\ink\da-DK
..trimmed for brevity..
```

PyTan API Invalid Question Examples

Invalid ask manual question sensor help

Have ask_manual() return the help for sensors

Example Python Code

```
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["qtype"] = u'manual'
kwargs["sensors_help"] = True


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

Output from Python Code

```
Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
Traceback (most recent call last):
  File "<string>", line 55, in <module>
  File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 130, in ask
    result = getattr(self, q_obj_map['handler'])(**kwargs)
  File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 271, in ask_manual
    raise pytan.exceptions.PytanHelp(pytan.help.help_sensors())
```

```
PytanHelp:
Sensors Help
============

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

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

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

Supplying a single sensor:

    'Computer Name'

Supplying two sensors in a list of strings:

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

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

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

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

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

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

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

    * if the parameter does not require a default value, the
    parameter is left blank and not supplied.
    * if the parameter does require a value (pulldowns, for
    example), a default value is derived (for pulldowns,
    the first value available as a pulldown entry is used).

Examples for sensors with parameters
------------------------------------
```

```
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
```

```
appropriate value required by accordant OPTION.

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

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

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

    'Computer Name, opt:max_data_age:60'

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

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

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

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

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

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

Invalid ask manual question filter help

Have ask_manual() return the help for filters

Example Python Code

```
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)
```

```
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
```

```
define group membership, deploy actions to determine which machines
should have the action deployed to it, and more.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

```

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

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

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

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

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

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

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

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

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

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

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

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

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

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

    'ge'
        Help: Filter for greater than or equal to VALUE
```

```
        Example: "Sensor1, that ge:VALUE"

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

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

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

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

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

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

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

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

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

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

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

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

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

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

```

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

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

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

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

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

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

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

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

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

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

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

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

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

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

    'not ends with'
        Help: Filter for does bit end with VALUE (adds .* before VALUE)
```

```
        Example: "Sensor1, that not ends with:VALUE"

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

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

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

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

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

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

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

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

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

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

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

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

Invalid ask manual question option help

Have ask_manual() return the help for options

Example Python Code

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

Output from Python Code

```
Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
Traceback (most recent call last):
  File "<string>", line 55, in <module>
  File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 130, in ask
    result = getattr(self, q_obj_map['handler'])(**kwargs)
  File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 277, in ask_manual
    raise pytan.exceptions.PytanHelp(pytan.help.help_options())
PytanHelp:
Options Help
============

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

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

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

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

Valid Options
-------------

    'ignore_case'
        Help: Make the filter do a case insensitive match
        Usable on: filter
        Example for sensor: "Sensor1, opt:ignore_case"
        Example for question: "ignore_case"

    'match_case'
        Help: Make the filter do a case sensitive match
        Usable on: filter
        Example for sensor: "Sensor1, opt:match_case"
        Example for question: "match_case"

    'match_any_value'
        Help: Make the filter match any value
        Usable on: filter
        Example for sensor: "Sensor1, opt:match_any_value"
        Example for question: "match_any_value"

    'match_all_values'
        Help: Make the filter match all values
        Usable on: filter
        Example for sensor: "Sensor1, opt:match_all_values"
        Example for question: "match_all_values"

    'max_data_age'
```

```
        Help: Re-fetch cached values older than N seconds
        Usable on: filter
        VALUE description and type: seconds, <type 'int'>
        Example for sensor: "Sensor1, opt:max_data_age:seconds"
        Example for question: "max_data_age:seconds"

    'value_type'
        Help: Make the filter consider the value type as VALUE_TYPE
        Usable on: filter
        VALUE description and type: value_type, <type 'str'>
        Example for sensor: "Sensor1, opt:value_type:value_type"
        Example for question: "value_type:value_type"

    'and'
        Help: Use 'and' for all of the filters supplied
        Usable on: group
        Example for sensor: "Sensor1, opt:and"
        Example for question: "and"

    'or'
        Help: Use 'or' for all of the filters supplied
        Usable on: group
        Example for sensor: "Sensor1, opt:or"
        Example for question: "or"
```

Invalid ask manual question sensor

Ask a question using a sensor that does not exist

Example Python Code

```
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"
```

```
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
```

```
    raise pytan.exceptions.HandlerError(err(search_str))
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

```
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'Computer name, that does not meet:little'
```

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

Output from Python Code

```
Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
Traceback (most recent call last):
  File "<string>", line 55, in <module>
  File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 130, in ask
    result = getattr(self, q_obj_map['handler'])(**kwargs)
  File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 297, in ask_manual
    sensor_defs = pytan.utils.dehumanize_sensors(sensors)
  File "/Users/jolsen/gh/pytan/lib/pytan/utils.py", line 1478, in dehumanize_sensors
    s, parsed_filter = extract_filter(s)
  File "/Users/jolsen/gh/pytan/lib/pytan/utils.py", line 1846, in extract_filter
    raise pytan.exceptions.HumanParserError(err(split_filter[1]))
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

```
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'Folder Name Search with RegEx Match{dirname=Program Files,regex=.*}{}'
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
try:
    handler.ask(**kwargs)
except Exception as e:
    traceback.print_exc(file=sys.stdout)
```

Output from Python Code

```
Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
Traceback (most recent call last):
  File "<string>", line 55, in <module>
  File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 130, in ask
    result = getattr(self, q_obj_map['handler'])(**kwargs)
  File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 297, in ask_manual
    sensor_defs = pytan.utils.dehumanize_sensors(sensors)
  File "/Users/jolsen/gh/pytan/lib/pytan/utils.py", line 1476, in dehumanize_sensors
    s, parsed_params = extract_params(s)
  File "/Users/jolsen/gh/pytan/lib/pytan/utils.py", line 1646, in extract_params
    raise pytan.exceptions.HumanParserError(err(s))
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

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

Output from Python Code

```
Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
Traceback (most recent call last):
  File "<string>", line 55, in <module>
  File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 130, in ask
    result = getattr(self, q_obj_map['handler'])(**kwargs)
  File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 297, in ask_manual
    sensor_defs = pytan.utils.dehumanize_sensors(sensors)
  File "/Users/jolsen/gh/pytan/lib/pytan/utils.py", line 1477, in dehumanize_sensors
    s, parsed_options = extract_options(s)
  File "/Users/jolsen/gh/pytan/lib/pytan/utils.py", line 1721, in extract_options
    parsed_options = map_options(parsed_options, ['filter'])
  File "/Users/jolsen/gh/pytan/lib/pytan/utils.py", line 1751, in map_options
    raise pytan.exceptions.HumanParserError(err(option))
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

```
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'Computer Name{Dweedle}'
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
try:
    handler.ask(**kwargs)
except Exception as e:
    traceback.print_exc(file=sys.stdout)
```

Output from Python Code

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

PyTan API Valid Get Object Examples

Get action by id

Get an action by id

Example Python Code

```
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)
```

```
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])
```

```
if len(out.splitlines()) > 15:
    out = out.splitlines()[0:15]
    out.append('..trimmed for brevity..')
    out = '\n'.join(out)

print out
```

Output from Python Code

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

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

print of response:
ActionList, len: 1

length of response (number of objects returned):
1

print the first object returned in JSON format:
{
  "_type": "action",
  "action_group": {
    "_type": "group",
    "id": 0,
    "name": "Default"
  },
  "approver": {
    "_type": "user",
    "id": 1,
    "name": "Jim Olsen"
  },
  "comment": "Distribute Tanium Standard Utilities",
  "creation_time": "2015-08-07T13:22:26",
  "distribute_seconds": 3200,
..trimmed for brevity..
```

Get question by id

Get a question by id

Example Python Code

```
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'question'
kwargs["id"] = 1

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

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

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

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

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

```

print out
```

Output from Python Code

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

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

print of response:
QuestionList, len: 1

length of response (number of objects returned):
1

print the first object returned in JSON format:
{
  "_type": "question",
  "action_tracking_flag": 0,
  "context_group": {
    "_type": "group",
    "id": 0
  },
  "expiration": "2015-08-07T13:31:47",
  "expire_seconds": 0,
  "force_computer_id_flag": 1,
  "hidden_flag": 0,
  "id": 1,
  "management_rights_group": {
    "_type": "group",
    "id": 0
..trimmed for brevity..
```

Get saved question by names

Get two saved questions by name

Example Python Code

```
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'saved_question'
kwargs["name"] = [u'Installed Applications', u'Computer Name']

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

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

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

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

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

print out
```

Output from Python Code

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

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

print of response:
SavedQuestionList, len: 2

length of response (number of objects returned):
2

print the first object returned in JSON format:
{
  "_type": "saved_question",
  "action_tracking_flag": 0,
  "archive_enabled_flag": 0,
  "archive_owner": {
    "_type": "user"
  },
  "expire_seconds": 600,
  "hidden_flag": 0,
  "id": 64,
  "issue_seconds": 120,
  "issue_seconds_never_flag": 0,
  "keep_seconds": 0,
  "metadata": {
    "_type": "metadata",
..trimmed for brevity..
```

Get userrole by id

Get a user role by id.

Example Python Code

```
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'
kwargs["id"] = 1

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

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

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

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

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

print out
```

Output from Python Code

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

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

```
print of response:
UserRoleList, len: 1

length of response (number of objects returned):
1

print the first object returned in JSON format:
{
  "_type": "role",
  "description": "Administrators can perform all functions in the system, including creating other u
  "id": 1,
  "name": "Administrator",
  "permissions": {
    "_type": "permissions",
    "permission": [
      "admin",
      "sensor_read",
      "sensor_write",
      "question_read",
      "question_write",
      "action_read",
      "action_write",
..trimmed for brevity..
```

Get leader clients

Get all clients that are Leader status

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"
```

```
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
```

```
length of response (number of objects returned):
1

print the first object returned in JSON format:
{
  "_type": "client_status",
  "cache_row_id": 1,
  "computer_id": "3741604154",
  "full_version": "6.0.314.1195",
  "host_name": "JTANIUM1.localdomain",
  "ipaddress_client": "172.16.31.128",
  "ipaddress_server": "172.16.31.128",
  "last_registration": "2015-08-07T19:45:00",
  "port_number": 17473,
  "protocol_version": 314,
  "public_key_valid": 1,
  "receive_state": "Previous Only",
  "send_state": "Backward Only",
  "status": "Leader"
..trimmed for brevity..
```

Get setting by name

Get a system setting 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'setting'
kwargs["name"] = u'control_address'

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

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

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

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

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

print out
```

Output from Python Code

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

Type of response:  <class 'taniumpy.object_types.system_setting_list.SystemSettingList'>

print of response:
SystemSettingList, len: 1

length of response (number of objects returned):
1

print the first object returned in JSON format:
```

```
{
  "_type": "system_setting",
  "default_value": "512:17473:127.0.0.1",
  "hidden_flag": 0,
  "id": 58,
  "name": "control_address",
  "read_only_flag": 0,
  "setting_type": "Server",
  "value": "512:17473:127.0.0.1",
  "value_type": "Text"
}
```

Get user by name

Get a user by name

Example Python Code

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

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

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

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

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

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

print out
```

Output from Python Code

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

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

print of response:
UserList, len: 1

length of response (number of objects returned):
1

print the first object returned in JSON format:
{
  "_type": "user",
  "deleted_flag": 0,
  "group_id": 0,
  "id": 2,
  "last_login": "2015-08-07T19:45:07",
  "local_admin_flag": 1,
  "name": "Tanium User",
  "permissions": {
```

```
    "_type": "permissions",
    "permission": [
      "admin",
      "sensor_read",
      "sensor_write",
      "question_read",
..trimmed for brevity..
```

Get sensor by id

Get a sensor by id

Example Python Code

```
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'sensor'
kwargs["id"] = 1

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

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

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

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

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

print out
```

Output from Python Code

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

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

print of response:
SensorList, len: 1

length of response (number of objects returned):
1

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

```
    "query": [
      {
..trimmed for brevity..
```

Get sensor by mixed

Get multiple sensors by id, name, and hash

Example Python Code

```
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'sensor'
kwargs["hash"] = [u'322086833']
kwargs["name"] = [u'Computer Name']
kwargs["id"] = [1, 2]

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

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

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

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

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

print out
```

Output from Python Code

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

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

print of response:
SensorList, len: 4

length of response (number of objects returned):
4

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

```
..trimmed for brevity..
```

Get whitelisted url by id

Get a whitelisted url by id

Example Python Code

```
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'whitelisted_url'
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

```
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 kwargs["name"] = u'All Computers'
48
49 # call the handler with the get method, passing in kwargs for arguments
50 response = handler.get(**kwargs)
51
52 print ""
53 print "Type of response: ", type(response)
54
55 print ""
56 print "print of response:"
57 print response
58
59 print ""
60 print "length of response (number of objects returned): "
61 print len(response)
```

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

print out
```

Output from Python Code

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

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

print of response:
GroupList, len: 1

length of response (number of objects returned):
1

print the first object returned in JSON format:
{
  "_type": "group",
  "and_flag": 0,
  "deleted_flag": 0,
  "filters": {
    "_type": "filters",
    "filter": []
  },
  "id": 64,
  "name": "All Computers",
  "not_flag": 0,
  "sub_groups": {
    "_type": "groups",
    "group": []
  },
..trimmed for brevity..
```

Get sensor by hash

Get a sensor by hash

Example Python Code

```
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)
```

```
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])
```

```
if len(out.splitlines()) > 15:
    out = out.splitlines()[0:15]
    out.append('..trimmed for brevity..')
    out = '\n'.join(out)

print out
```

Output from Python Code

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

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

print of response:
SensorList, len: 1

length of response (number of objects returned):
1

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

Get package by name

Get a package by name

Example Python Code

```
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'package'
kwargs["name"] = u'Distribute Tanium Standard Utilities'

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

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

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

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

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

```
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)
12 lib_dir = os.path.join(pytan_root_dir, 'lib')
13 path_adds = [lib_dir]
14
15 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'sensor'
kwargs["name"] = [u'Computer Name', u'Action Statuses']

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

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

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

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

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

print out
```

Output from Python Code

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

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

print of response:
SensorList, len: 2

length of response (number of objects returned):
2

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

Get saved question by name

Get saved question by name

Example Python Code

```python
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'saved_question'
kwargs["name"] = u'Installed Applications'

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

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

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

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

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

print out
```

Output from Python Code

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

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

```

print of response:
SavedQuestionList, len: 1

length of response (number of objects returned):
1

print the first object returned in JSON format:
{
  "_type": "saved_question",
  "action_tracking_flag": 0,
  "archive_enabled_flag": 0,
  "archive_owner": {
    "_type": "user"
  },
  "expire_seconds": 600,
  "hidden_flag": 0,
  "id": 64,
  "issue_seconds": 120,
  "issue_seconds_never_flag": 0,
  "keep_seconds": 0,
  "metadata": {
    "_type": "metadata",
..trimmed for brevity..
```

Get user by id

Get a user by id

Example Python Code

```
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'user'
kwargs["id"] = 1

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

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

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

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

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

print out
```

Output from Python Code

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

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

print of response:
UserList, len: 1

```

```
length of response (number of objects returned):
1

print the first object returned in JSON format:
{
  "_type": "user",
  "deleted_flag": 0,
  "group_id": 0,
  "id": 1,
  "last_login": "2015-08-07T13:21:59",
  "local_admin_flag": -1,
  "name": "Jim Olsen",
  "permissions": {
    "_type": "permissions",
    "permission": [
      "admin",
      "sensor_read",
      "sensor_write",
      "question_read",
..trimmed for brevity..
```

Get sensor by name

Get a sensor by name

Example Python Code

```
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'sensor'
kwargs["name"] = u'Computer Name'

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

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

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

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

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

print out
```

Output from Python Code

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

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

print of response:
SensorList, len: 1

length of response (number of objects returned):
1

print the first object returned in JSON format:
```

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

Get saved action by name

Get a saved action by name

Example Python Code

```
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'saved_action'
kwargs["name"] = u'Distribute Tanium Standard Utilities'

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

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

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

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

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

print out
```

Output from Python Code

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

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

print of response:
SavedActionList, len: 1

length of response (number of objects returned):
1

print the first object returned in JSON format:
{
  "_type": "saved_action",
  "action_group_id": 0,
  "approved_flag": 1,
```

```
  "approver": {
    "_type": "user",
    "id": 1
  },
  "comment": "Distribute Tanium Standard Utilities",
  "creation_time": "2015-08-07T13:22:26",
  "distribute_seconds": 3200,
  "end_time": "Never",
  "expire_seconds": 3300,
  "id": 1,
  "issue_count": 0,
..trimmed for brevity..
```

Get all users

Get all users

Example Python Code

```
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'user'

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

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

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

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

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

print out
```

Output from Python Code

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

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

print of response:
UserList, len: 6

length of response (number of objects returned):
6

print the first object returned in JSON format:
{
  "_type": "user",
  "deleted_flag": 0,
  "group_id": 0,
  "id": 1,
  "last_login": "2015-08-07T13:21:59",
  "local_admin_flag": -1,
  "name": "Jim Olsen",
  "permissions": {
```

```
    "_type": "permissions",
    "permission": [
      "admin",
      "sensor_read",
      "sensor_write",
      "question_read",
..trimmed for brevity..
```

Get all saved actions

Get all saved actions

Example Python Code

```
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'saved_action'

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

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

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

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

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

print out
```

Output from Python Code

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

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

print of response:
SavedActionList, len: 4

length of response (number of objects returned):
4

print the first object returned in JSON format:
{
  "_type": "saved_action",
  "action_group_id": 0,
  "approved_flag": 1,
  "approver": {
    "_type": "user",
    "id": 1
  },
  "cache_row_id": 0,
  "comment": "Distribute Tanium Standard Utilities",
  "creation_time": "2015-08-07T13:22:26",
  "distribute_seconds": 3200,
  "end_time": "Never",
  "expire_seconds": 3300,
```

```
  "id": 1,
..trimmed for brevity..
```

Get all settings

Get all system settings

Example Python Code

```
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'setting'
```

```

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

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

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

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

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

print out
```

Output from Python Code

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

Type of response:  <class 'taniumpy.object_types.system_setting_list.SystemSettingList'>

print of response:
SystemSettingList, len: 91

length of response (number of objects returned):
91

print the first object returned in JSON format:
{
  "_type": "system_setting",
  "audit_data": {
    "_type": "audit_data",
    "creation_time": "2015-08-07T13:22:35",
    "last_modified_by": "Jim Olsen",
    "modification_time": "2015-08-07T13:22:35"
  },
  "cache_row_id": 0,
  "default_value": "0",
  "hidden_flag": 0,
  "id": 1,
  "name": "load_initial_content",
  "read_only_flag": 0,
  "setting_type": "Server",
..trimmed for brevity..
```

Get all saved questions

Get all saved questions

Example Python Code

```
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'saved_question'

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

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

```

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

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

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

print out
```

Output from Python Code

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

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

print of response:
SavedQuestionList, len: 107

length of response (number of objects returned):
107

print the first object returned in JSON format:
{
  "_type": "saved_question",
  "action_tracking_flag": 0,
  "archive_enabled_flag": 0,
  "archive_owner": {
    "_type": "user"
  },
  "cache_row_id": 0,
  "expire_seconds": 600,
  "hidden_flag": 0,
  "id": 1,
  "issue_seconds": 120,
  "issue_seconds_never_flag": 0,
  "keep_seconds": 0,
  "mod_time": "2015-08-07T13:22:22",
..trimmed for brevity..
```

Get all userroless

Get all user roles

Example Python Code

```
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)

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

print ""
```

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

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

print out
```

Output from Python Code

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

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

print of response:
UserRoleList, len: 9

length of response (number of objects returned):
9

print the first object returned in JSON format:
{
  "_type": "role",
  "description": "Administrators can perform all functions in the system, including creating other u
  "id": 1,
  "name": "Administrator",
  "permissions": {
    "_type": "permissions",
    "permission": [
      "admin",
      "sensor_read",
      "sensor_write",
      "question_read",
      "question_write",
      "action_read",
      "action_write",
..trimmed for brevity..
```

Get all questions

Get all questions

Example Python Code

```
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'question'

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

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

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

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

print ""
print "print the first object returned in JSON format:"
```

```
out = response.to_json(response[0])
if len(out.splitlines()) > 15:
    out = out.splitlines()[0:15]
    out.append('..trimmed for brevity..')
    out = '\n'.join(out)

print out
```

Output from Python Code

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

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

print of response:
QuestionList, len: 174

length of response (number of objects returned):
174

print the first object returned in JSON format:
{
  "_type": "question",
  "action_tracking_flag": 0,
  "cache_row_id": 1,
  "context_group": {
    "_type": "group",
    "id": 0
  },
  "expiration": "2015-08-07T13:32:29",
  "expire_seconds": 600,
  "hidden_flag": 0,
  "id": 104,
  "management_rights_group": {
    "_type": "group",
    "id": 0
..trimmed for brevity..
```

Get all groups

Get all groups

Example Python Code

```
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'

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

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

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

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

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

```

print out
```

Output from Python Code

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

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

print of response:
GroupList, len: 2

length of response (number of objects returned):
2

print the first object returned in JSON format:
{
  "_type": "group",
  "and_flag": 0,
  "deleted_flag": 0,
  "filters": {
    "_type": "filters",
    "filter": []
  },
  "id": 64,
  "name": "All Computers",
  "not_flag": 0,
  "sub_groups": {
    "_type": "groups",
    "group": []
  },
..trimmed for brevity..
```

Get all sensors

Get all sensors

Example Python Code

```
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'sensor'

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

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

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

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

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

print out
```

Output from Python Code

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

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

print of response:
SensorList, len: 419

length of response (number of objects returned):
419

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

Get all whitelisted urls

Get all whitelisted urls

Example Python Code

```
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'whitelisted_url'

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

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

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

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

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

print out
```

Output from Python Code

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

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

print of response:
```

```
WhiteListedUrlList, len: 46

length of response (number of objects returned):
46

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

Get all clients

Get all clients

Example Python Code

```
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'client'

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

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

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

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

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

print out
```

Output from Python Code

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

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

print of response:
SystemStatusList, len: 2

length of response (number of objects returned):
2

print the first object returned in JSON format:
{
  "_type": "client_status",
  "cache_row_id": 0,
  "computer_id": "3888017885",
  "full_version": "5.1.314.7724",
  "host_name": "Casus-Belli.local",
  "ipaddress_client": "172.16.31.1",
  "ipaddress_server": "172.16.31.1",
  "last_registration": "2015-08-07T19:44:58",
```

```
  "port_number": 17472,
  "protocol_version": 314,
  "public_key_valid": 1,
  "send_state": "Forward Only",
  "status": "Leader, Slow Link Behind"
}
```

Get all packages

Get all packages

Example Python Code

```
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["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",
```

```
..trimmed for brevity..
```

Get all actions

Get all actions

Example Python Code

```
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'action'

```

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

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

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

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

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

print out
```

Output from Python Code

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

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

print of response:
ActionList, len: 38

length of response (number of objects returned):
38

print the first object returned in JSON format:
{
  "_type": "action",
  "action_group": {
    "_type": "group",
    "id": 0,
    "name": "Default"
  },
  "approver": {
    "_type": "user",
    "id": 2,
    "name": "Tanium User"
  },
  "cache_row_id": 0,
  "comment": "Distribute Tanium Standard Utilities",
  "creation_time": "2015-08-07T13:26:19",
..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

```
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'action'
kwargs["name"] = u'Distribute Tanium Standard Utilities'


```

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

Output from Python Code

```
Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
Traceback (most recent call last):
  File "<string>", line 55, in <module>
  File "/Users/jolsen/gh/pytan/lib/pytan/utils.py", line 2699, in wrap
    ret = f(*args, **kwargs)
  File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 1245, in get
    raise pytan.exceptions.HandlerError(err(objtype, api_attrs))
HandlerError: Getting a action requires at least one filter: ['id']
```

Invalid get question by name

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

Example Python Code

```
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'question'
kwargs["name"] = u'dweedle'


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

Output from Python Code

```
Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
Traceback (most recent call last):
  File "<string>", line 55, in <module>
  File "/Users/jolsen/gh/pytan/lib/pytan/utils.py", line 2699, in wrap
    ret = f(*args, **kwargs)
  File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 1245, in get
    raise pytan.exceptions.HandlerError(err(objtype, api_attrs))
HandlerError: Getting a question requires at least one filter: ['id']
```

PyTan API Valid Deploy Action Examples

Deploy action simple

Deploy an action against all computers using human strings.

Example Python Code

```
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)
```

```
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
```

```
2015-08-07 19:45:28,448 DEBUG    pytan.handler.ActionPoller: ID 56: stopped_flag resolved to 0
2015-08-07 19:45:28,448 DEBUG    pytan.handler.ActionPoller: ID 56: status resolved to Open
2015-08-07 19:45:28,448 INFO     pytan.handler.ActionPoller: ID 56: Reached Threshold for Finished C

Type of response:  <type 'dict'>

Pretty print of response:
{'action_info': <taniumpy.object_types.result_info.ResultInfo object at 0x10c063810>,
 'action_object': <taniumpy.object_types.action.Action object at 0x10c063a50>,
 'action_result_map': {'failed': {'56:Expired.': [],
                                  '56:Failed.': [],
                                  '56:NotSucceeded.': [],
                                  '56:Stopped.': [],
                                  'total': 0},
                       'finished': {'56:Completed.': ['Casus-Belli.local',
                                                      'JTANIUM1.localdomain'],
                                    '56:Expired.': [],
                                    '56:Failed.': [],
                                    '56:NotSucceeded.': [],
                                    '56:Stopped.': [],
                                    '56:Succeeded.': [],
                                    '56:Verified.': [],
                                    'total': 2},
                       'running': {'56:Copying.': [],
                                   '56:Downloading.': [],
                                   '56:PendingVerification.': [],
                                   '56:Running.': [],
                                   '56:Waiting.': [],
                                   'total': 0},
                       'success': {'56:Completed.': ['Casus-Belli.local',
                                                     'JTANIUM1.localdomain'],
                                   '56:Verified.': [],
                                   'total': 2},
                       'unknown': {'total': 0}},
 'action_results': <taniumpy.object_types.result_set.ResultSet object at 0x11acb14d0>,
 'group_object': None,
 'package_object': <taniumpy.object_types.package_spec.PackageSpec object at 0x10c03d0d0>,
 'poller_object': <pytan.pollers.ActionPoller object at 0x10c063b10>,
 'poller_success': True,
 'saved_action_object': <taniumpy.object_types.saved_action.SavedAction object at 0x10bf69750>}

Print of action object:
Action, name: 'API Deploy Distribute Tanium Standard Utilities', id: 56

CSV Results of response:
Action Statuses,Computer Name
56:Completed.,Casus-Belli.local
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

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

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

print ""
print "Pretty print of response:"
```

```
print pprint.pformat(response)

print ""
print "Print of action object: "
print response['action_object']

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

# if results were returned (i.e. get_results=True was one of the kwargs passed in):
if response['action_results']:
    # call the write_csv() method to convert response to CSV and store it in out
    response['action_results'].write_csv(out, response['action_results'])

    print ""
    print "CSV Results of response: "
    print out.getvalue()
```

Output from Python Code

```
Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2015-08-07 19:45:28,497 DEBUG    pytan.handler.ActionPoller: ID 57: id resolved to 57
2015-08-07 19:45:28,497 DEBUG    pytan.handler.ActionPoller: ID 57: package_spec resolved to Package
2015-08-07 19:45:28,503 DEBUG    pytan.handler.ActionPoller: ID 57: target_group resolved to Group,
2015-08-07 19:45:28,504 DEBUG    pytan.handler.ActionPoller: ID 57: Result Map resolved to {'failed'
2015-08-07 19:45:28,504 DEBUG    pytan.handler.ActionPoller: ID 57: expiration_time resolved to 2015
2015-08-07 19:45:28,504 DEBUG    pytan.handler.ActionPoller: ID 57: status resolved to Open
2015-08-07 19:45:28,504 DEBUG    pytan.handler.ActionPoller: ID 57: stopped_flag resolved to 0
2015-08-07 19:45:28,504 DEBUG    pytan.handler.ActionPoller: ID 57: Object Info resolved to ID 57: P

Type of response:  <type 'dict'>

Pretty print of response:
{'action_info': <taniumpy.object_types.result_info.ResultInfo object at 0x10be950d0>,
 'action_object': <taniumpy.object_types.action.Action object at 0x10bf70fd0>,
 'action_result_map': None,
 'action_results': None,
 'group_object': None,
 'package_object': <taniumpy.object_types.package_spec.PackageSpec object at 0x11aae6050>,
 'poller_object': <pytan.pollers.ActionPoller object at 0x11aae6090>,
 'poller_success': None,
 'saved_action_object': <taniumpy.object_types.saved_action.SavedAction object at 0x10c063a50>}

Print of action object:
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

```
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

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

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

```

```
print ""
print "Print of action object: "
print response['action_object']

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

# if results were returned (i.e. get_results=True was one of the kwargs passed in):
if response['action_results']:
    # call the write_csv() method to convert response to CSV and store it in out
    response['action_results'].write_csv(out, response['action_results'])

    print ""
    print "CSV Results of response: "
    print out.getvalue()
```

Output from Python Code

```
Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2015-08-07 19:45:28,560 DEBUG    pytan.handler.ActionPoller: ID 58: id resolved to 58
2015-08-07 19:45:28,561 DEBUG    pytan.handler.ActionPoller: ID 58: package_spec resolved to Package
2015-08-07 19:45:28,567 DEBUG    pytan.handler.ActionPoller: ID 58: target_group resolved to Group,
2015-08-07 19:45:28,583 DEBUG    pytan.handler.ActionPoller: ID 58: Result Map resolved to {'failed'
2015-08-07 19:45:28,583 DEBUG    pytan.handler.ActionPoller: ID 58: expiration_time resolved to 2015
2015-08-07 19:45:28,583 DEBUG    pytan.handler.ActionPoller: ID 58: status resolved to Open
2015-08-07 19:45:28,584 DEBUG    pytan.handler.ActionPoller: ID 58: stopped_flag resolved to 0
2015-08-07 19:45:28,584 DEBUG    pytan.handler.ActionPoller: ID 58: Object Info resolved to ID 58: P
2015-08-07 19:45:28,584 DEBUG    pytan.handler.ActionPoller: ID 58: Adding Question to derive passed
2015-08-07 19:45:28,615 DEBUG    pytan.handler.QuestionPoller: ID 1302: id resolved to 1302
2015-08-07 19:45:28,615 DEBUG    pytan.handler.QuestionPoller: ID 1302: expiration resolved to 2015-
2015-08-07 19:45:28,615 DEBUG    pytan.handler.QuestionPoller: ID 1302: query_text resolved to Get n
2015-08-07 19:45:28,615 DEBUG    pytan.handler.QuestionPoller: ID 1302: id resolved to 1302
2015-08-07 19:45:28,615 DEBUG    pytan.handler.QuestionPoller: ID 1302: Object Info resolved to Ques
2015-08-07 19:45:28,618 DEBUG    pytan.handler.QuestionPoller: ID 1302: Progress: Tested: 0, Passed:
2015-08-07 19:45:28,618 DEBUG    pytan.handler.QuestionPoller: ID 1302: Timing: Started: 2015-08-07
2015-08-07 19:45:28,618 INFO     pytan.handler.QuestionPoller: ID 1302: Progress Changed 0% (0 of 2)
2015-08-07 19:45:33,623 DEBUG    pytan.handler.QuestionPoller: ID 1302: Progress: Tested: 1, Passed:
2015-08-07 19:45:33,623 DEBUG    pytan.handler.QuestionPoller: ID 1302: Timing: Started: 2015-08-07
2015-08-07 19:45:33,623 INFO     pytan.handler.QuestionPoller: ID 1302: Progress Changed 50% (1 of 2
2015-08-07 19:45:38,626 DEBUG    pytan.handler.QuestionPoller: ID 1302: Progress: Tested: 2, Passed:
2015-08-07 19:45:38,626 DEBUG    pytan.handler.QuestionPoller: ID 1302: Timing: Started: 2015-08-07
2015-08-07 19:45:38,626 INFO     pytan.handler.QuestionPoller: ID 1302: Progress Changed 100% (2 of
2015-08-07 19:45:38,626 INFO     pytan.handler.QuestionPoller: ID 1302: Reached Threshold of 99% (2
2015-08-07 19:45:38,626 DEBUG    pytan.handler.ActionPoller: ID 58: Passed Count resolved to 1
2015-08-07 19:45:38,638 DEBUG    pytan.handler.ActionPoller: ID 58: Progress: Seen Action: 0, Expect
2015-08-07 19:45:38,638 DEBUG    pytan.handler.ActionPoller: ID 58: Timing: Started: 2015-08-07 19:4
2015-08-07 19:45:38,638 INFO     pytan.handler.ActionPoller: ID 58: Progress Changed for Seen Count
2015-08-07 19:45:38,645 DEBUG    pytan.handler.ActionPoller: ID 58: stopped_flag resolved to 0
2015-08-07 19:45:38,645 DEBUG    pytan.handler.ActionPoller: ID 58: status resolved to Open
2015-08-07 19:45:43,660 DEBUG    pytan.handler.ActionPoller: ID 58: Progress: Seen Action: 0, Expect
2015-08-07 19:45:43,660 DEBUG    pytan.handler.ActionPoller: ID 58: Timing: Started: 2015-08-07 19:4
2015-08-07 19:45:43,667 DEBUG    pytan.handler.ActionPoller: ID 58: stopped_flag resolved to 0
2015-08-07 19:45:43,667 DEBUG    pytan.handler.ActionPoller: ID 58: status resolved to Open
2015-08-07 19:45:48,677 DEBUG    pytan.handler.ActionPoller: ID 58: Progress: Seen Action: 1, Expect
2015-08-07 19:45:48,677 DEBUG    pytan.handler.ActionPoller: ID 58: Timing: Started: 2015-08-07 19:4
2015-08-07 19:45:48,677 INFO     pytan.handler.ActionPoller: ID 58: Progress Changed for Seen Count
2015-08-07 19:45:48,684 DEBUG    pytan.handler.ActionPoller: ID 58: stopped_flag resolved to 0
```

```
2015-08-07 19:45:48,684 DEBUG    pytan.handler.ActionPoller: ID 58: status resolved to Open
2015-08-07 19:45:48,684 INFO     pytan.handler.ActionPoller: ID 58: Reached Threshold for Seen Count
2015-08-07 19:45:48,692 DEBUG    pytan.handler.ActionPoller: ID 58: failed: 0, finished: 1, running:
2015-08-07 19:45:48,692 DEBUG    pytan.handler.ActionPoller: ID 58: Timing: Started: 2015-08-07 19:4
2015-08-07 19:45:48,692 INFO     pytan.handler.ActionPoller: ID 58: Progress Changed for Finished Co
2015-08-07 19:45:48,698 DEBUG    pytan.handler.ActionPoller: ID 58: stopped_flag resolved to 0
2015-08-07 19:45:48,698 DEBUG    pytan.handler.ActionPoller: ID 58: status resolved to Open
2015-08-07 19:45:48,698 INFO     pytan.handler.ActionPoller: ID 58: Reached Threshold for Finished C

Type of response:  <type 'dict'>

Pretty print of response:
{'action_info': <taniumpy.object_types.result_info.ResultInfo object at 0x11aadbcd0>,
 'action_object': <taniumpy.object_types.action.Action object at 0x11aae6090>,
 'action_result_map': {'failed': {'58:Expired.': [],
                                  '58:Failed.': [],
                                  '58:NotSucceeded.': [],
                                  '58:Stopped.': [],
                                  'total': 0},
                       'finished': {'58:Completed.': ['JTANIUM1.localdomain'],
                                    '58:Expired.': [],
                                    '58:Failed.': [],
                                    '58:NotSucceeded.': [],
                                    '58:Stopped.': [],
                                    '58:Succeeded.': [],
                                    '58:Verified.': [],
                                    'total': 1},
                       'running': {'58:Copying.': [],
                                   '58:Downloading.': [],
                                   '58:PendingVerification.': [],
                                   '58:Running.': [],
                                   '58:Waiting.': [],
                                   'total': 0},
                       'success': {'58:Completed.': ['JTANIUM1.localdomain'],
                                   '58:Verified.': [],
                                   'total': 1},
                       'unknown': {'total': 0}},
 'action_results': <taniumpy.object_types.result_set.ResultSet object at 0x11ac77c90>,
 'group_object': <taniumpy.object_types.group.Group object at 0x10c03db90>,
 'package_object': <taniumpy.object_types.package_spec.PackageSpec object at 0x11aae6050>,
 'poller_object': <pytan.pollers.ActionPoller object at 0x10bea6950>,
 'poller_success': True,
 'saved_action_object': <taniumpy.object_types.saved_action.SavedAction object at 0x11acb14d0>}

Print of action object:
Action, name: 'API Deploy Distribute Tanium Standard Utilities', id: 58

CSV Results of response:
Action Statuses,Computer Name
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

```
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'Custom Tagging - Add Tags{$1=tag_should_be_added,$2=tag_should_be_ignore}'

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

print ""
```

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

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

print ""
print "Print of action object: "
print response['action_object']

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

# if results were returned (i.e. get_results=True was one of the kwargs passed in):
if response['action_results']:
    # call the write_csv() method to convert response to CSV and store it in out
    response['action_results'].write_csv(out, response['action_results'])

    print ""
    print "CSV Results of response: "
    print out.getvalue()
```

Output from Python Code

```
Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2015-08-07 19:45:48,755 DEBUG    pytan.handler.ActionPoller: ID 59: id resolved to 59
2015-08-07 19:45:48,755 DEBUG    pytan.handler.ActionPoller: ID 59: package_spec resolved to Package
2015-08-07 19:45:48,759 DEBUG    pytan.handler.ActionPoller: ID 59: target_group resolved to Group,
2015-08-07 19:45:48,773 DEBUG    pytan.handler.ActionPoller: ID 59: Result Map resolved to {'failed'
2015-08-07 19:45:48,773 DEBUG    pytan.handler.ActionPoller: ID 59: expiration_time resolved to 2015
2015-08-07 19:45:48,773 DEBUG    pytan.handler.ActionPoller: ID 59: status resolved to Open
2015-08-07 19:45:48,773 DEBUG    pytan.handler.ActionPoller: ID 59: stopped_flag resolved to 0
2015-08-07 19:45:48,773 DEBUG    pytan.handler.ActionPoller: ID 59: Object Info resolved to ID 59: P
2015-08-07 19:45:48,773 DEBUG    pytan.handler.ActionPoller: ID 59: Adding Question to derive passed
2015-08-07 19:45:48,802 DEBUG    pytan.handler.QuestionPoller: ID 1303: id resolved to 1303
2015-08-07 19:45:48,802 DEBUG    pytan.handler.QuestionPoller: ID 1303: expiration resolved to 2015-
2015-08-07 19:45:48,802 DEBUG    pytan.handler.QuestionPoller: ID 1303: query_text resolved to Get n
2015-08-07 19:45:48,802 DEBUG    pytan.handler.QuestionPoller: ID 1303: id resolved to 1303
2015-08-07 19:45:48,802 DEBUG    pytan.handler.QuestionPoller: ID 1303: Object Info resolved to Ques
2015-08-07 19:45:48,805 DEBUG    pytan.handler.QuestionPoller: ID 1303: Progress: Tested: 0, Passed:
2015-08-07 19:45:48,805 DEBUG    pytan.handler.QuestionPoller: ID 1303: Timing: Started: 2015-08-07
2015-08-07 19:45:48,805 INFO     pytan.handler.QuestionPoller: ID 1303: Progress Changed 0% (0 of 2)
2015-08-07 19:45:53,809 DEBUG    pytan.handler.QuestionPoller: ID 1303: Progress: Tested: 2, Passed:
2015-08-07 19:45:53,809 DEBUG    pytan.handler.QuestionPoller: ID 1303: Timing: Started: 2015-08-07
2015-08-07 19:45:53,809 INFO     pytan.handler.QuestionPoller: ID 1303: Progress Changed 100% (2 of
2015-08-07 19:45:53,809 INFO     pytan.handler.QuestionPoller: ID 1303: Reached Threshold of 99% (2
2015-08-07 19:45:53,809 DEBUG    pytan.handler.ActionPoller: ID 59: Passed Count resolved to 1
2015-08-07 19:45:53,819 DEBUG    pytan.handler.ActionPoller: ID 59: Progress: Seen Action: 0, Expect
2015-08-07 19:45:53,819 DEBUG    pytan.handler.ActionPoller: ID 59: Timing: Started: 2015-08-07 19:4
2015-08-07 19:45:53,819 INFO     pytan.handler.ActionPoller: ID 59: Progress Changed for Seen Count
2015-08-07 19:45:53,824 DEBUG    pytan.handler.ActionPoller: ID 59: stopped_flag resolved to 0
2015-08-07 19:45:53,824 DEBUG    pytan.handler.ActionPoller: ID 59: status resolved to Open
2015-08-07 19:45:58,837 DEBUG    pytan.handler.ActionPoller: ID 59: Progress: Seen Action: 1, Expect
2015-08-07 19:45:58,837 DEBUG    pytan.handler.ActionPoller: ID 59: Timing: Started: 2015-08-07 19:4
2015-08-07 19:45:58,837 INFO     pytan.handler.ActionPoller: ID 59: Progress Changed for Seen Count
2015-08-07 19:45:58,843 DEBUG    pytan.handler.ActionPoller: ID 59: stopped_flag resolved to 0
2015-08-07 19:45:58,844 DEBUG    pytan.handler.ActionPoller: ID 59: status resolved to Open
```

```
2015-08-07 19:45:58,844 INFO     pytan.handler.ActionPoller: ID 59: Reached Threshold for Seen Count
2015-08-07 19:45:58,852 DEBUG    pytan.handler.ActionPoller: ID 59: failed: 0, finished: 0, running:
2015-08-07 19:45:58,852 DEBUG    pytan.handler.ActionPoller: ID 59: Timing: Started: 2015-08-07 19:4
2015-08-07 19:45:58,852 INFO     pytan.handler.ActionPoller: ID 59: Progress Changed for Finished Co
2015-08-07 19:45:58,858 DEBUG    pytan.handler.ActionPoller: ID 59: stopped_flag resolved to 0
2015-08-07 19:45:58,858 DEBUG    pytan.handler.ActionPoller: ID 59: status resolved to Open
2015-08-07 19:46:03,873 DEBUG    pytan.handler.ActionPoller: ID 59: failed: 0, finished: 0, running:
2015-08-07 19:46:03,873 DEBUG    pytan.handler.ActionPoller: ID 59: Timing: Started: 2015-08-07 19:4
2015-08-07 19:46:03,879 DEBUG    pytan.handler.ActionPoller: ID 59: stopped_flag resolved to 0
2015-08-07 19:46:03,879 DEBUG    pytan.handler.ActionPoller: ID 59: status resolved to Open
2015-08-07 19:46:08,892 DEBUG    pytan.handler.ActionPoller: ID 59: failed: 0, finished: 1, running:
2015-08-07 19:46:08,892 DEBUG    pytan.handler.ActionPoller: ID 59: Timing: Started: 2015-08-07 19:4
2015-08-07 19:46:08,892 INFO     pytan.handler.ActionPoller: ID 59: Progress Changed for Finished Co
2015-08-07 19:46:08,898 DEBUG    pytan.handler.ActionPoller: ID 59: stopped_flag resolved to 0
2015-08-07 19:46:08,898 DEBUG    pytan.handler.ActionPoller: ID 59: status resolved to Open
2015-08-07 19:46:08,898 INFO     pytan.handler.ActionPoller: ID 59: Reached Threshold for Finished C

Type of response:  <type 'dict'>

Pretty print of response:
{'action_info': <taniumpy.object_types.result_info.ResultInfo object at 0x10c063090>,
 'action_object': <taniumpy.object_types.action.Action object at 0x10c17ecd0>,
 'action_result_map': {'failed': {'59:Expired.': [],
                                  '59:Failed.': [],
                                  '59:NotSucceeded.': [],
                                  '59:Stopped.': [],
                                  'total': 0},
                       'finished': {'59:Completed.': ['JTANIUM1.localdomain'],
                                    '59:Expired.': [],
                                    '59:Failed.': [],
                                    '59:NotSucceeded.': [],
                                    '59:Stopped.': [],
                                    '59:Succeeded.': [],
                                    '59:Verified.': [],
                                    'total': 1},
                       'running': {'59:Copying.': [],
                                   '59:Downloading.': ['JTANIUM1.localdomain'],
                                   '59:PendingVerification.': [],
                                   '59:Running.': [],
                                   '59:Waiting.': [],
                                   'total': 1},
                       'success': {'59:Completed.': ['JTANIUM1.localdomain'],
                                   '59:Verified.': [],
                                   'total': 1},
                       'unknown': {'total': 0}},
 'action_results': <taniumpy.object_types.result_set.ResultSet object at 0x10c17ec10>,
 'group_object': <taniumpy.object_types.group.Group object at 0x10c03d210>,
 'package_object': <taniumpy.object_types.package_spec.PackageSpec object at 0x10bf69dd0>,
 'poller_object': <pytan.pollers.ActionPoller object at 0x10c063250>,
 'poller_success': True,
 'saved_action_object': <taniumpy.object_types.saved_action.SavedAction object at 0x10c03ddd0>}

Print of action object:
Action, name: 'API Deploy Custom Tagging - Add Tags', id: 59

CSV Results of response:
Action Statuses,Computer Name
```

```
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

```
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['report_dir'] = tempfile.gettempdir()
kwargs["package"] = u'Distribute Tanium Standard Utilities'


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

Output from Python Code

```
Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2015-08-07 19:46:08,957 DEBUG    pytan.handler.QuestionPoller: ID 1304: id resolved to 1304
2015-08-07 19:46:08,957 DEBUG    pytan.handler.QuestionPoller: ID 1304: expiration resolved to 2015-
2015-08-07 19:46:08,957 DEBUG    pytan.handler.QuestionPoller: ID 1304: query_text resolved to Get C
2015-08-07 19:46:08,957 DEBUG    pytan.handler.QuestionPoller: ID 1304: id resolved to 1304
2015-08-07 19:46:08,957 DEBUG    pytan.handler.QuestionPoller: ID 1304: Object Info resolved to Ques
2015-08-07 19:46:08,962 DEBUG    pytan.handler.QuestionPoller: ID 1304: Progress: Tested: 0, Passed:
2015-08-07 19:46:08,962 DEBUG    pytan.handler.QuestionPoller: ID 1304: Timing: Started: 2015-08-07
2015-08-07 19:46:08,962 INFO     pytan.handler.QuestionPoller: ID 1304: Progress Changed 0% (0 of 2)
2015-08-07 19:46:13,969 DEBUG    pytan.handler.QuestionPoller: ID 1304: Progress: Tested: 2, Passed:
2015-08-07 19:46:13,969 DEBUG    pytan.handler.QuestionPoller: ID 1304: Timing: Started: 2015-08-07
2015-08-07 19:46:13,969 INFO     pytan.handler.QuestionPoller: ID 1304: Progress Changed 100% (2 of
2015-08-07 19:46:13,969 INFO     pytan.handler.QuestionPoller: ID 1304: Reached Threshold of 99% (2
2015-08-07 19:46:13,974 INFO     pytan.handler: Report file '/var/folders/dk/vjr1r_c53yx6k6gzp2bbt_c
Traceback (most recent call last):
  File "<string>", line 55, in <module>
  File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 404, in deploy_action
    **kwargs
  File "/Users/jolsen/gh/pytan/lib/pytan/utils.py", line 2699, in wrap
    ret = f(*args, **kwargs)
  File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 1662, in _deploy_action
    raise pytan.exceptions.RunFalse(m(report_path, len(result)))
RunFalse: 'Run' is not True!!
View and verify the contents of /var/folders/dk/vjr1r_c53yx6k6gzp2bbt_c40000gn/T/VERIFY_BEFORE_DEPLC
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

```
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['report_dir'] = tempfile.gettempdir()
kwargs["package_help"] = True


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

Output from Python Code

```
Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
Traceback (most recent call last):
  File "<string>", line 55, in <module>
  File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 374, in deploy_action
    raise pytan.exceptions.PytanHelp(pytan.help.help_package())
```

```
PytanHelp:
Package Help
============

Supplying package defines what package will be deployed as part of the
action.

A package string is a human string that describes, at a minimum, a
package. It can also optionally define a selector for the package,
and/or parameters for the package. A package must be provided as a string.

Examples for package
--------------------------------

Supplying a package:

    'Distribute Tanium Standard Utilities'

Supplying a package by id:

    'id:1'

Supplying a package by hash:

    'hash:123456789'

Supplying a package by name:

    'name:Distribute Tanium Standard Utilities'

Package Parameters
------------------

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

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

If a parameter is supplied and the package doesn't have a
corresponding key name, it will be ignored. If the package has
parameters and a parameter is NOT supplied then an exception
will be raised, printing out the JSON of the missing paramater
for the package in question.

Examples for package with parameters
------------------------------------

Supplying a package with a single parameter '$1':

    'Package With Params{$1=value1}'

Supplying a package with two parameters, '$1' and '$2':

```

```
    'Package With Params{$1=value1,$2=value2}'
```

Invalid deploy action package

Deploy an action using a non-existing package.

Example Python Code

```
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['report_dir'] = tempfile.gettempdir()
kwargs["run"] = True
```

```
kwargs["package"] = u'Invalid Package'


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

Output from Python Code

```
Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
Traceback (most recent call last):
  File "<string>", line 56, in <module>
  File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 404, in deploy_action
    **kwargs
  File "/Users/jolsen/gh/pytan/lib/pytan/utils.py", line 2699, in wrap
    ret = f(*args, **kwargs)
  File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 1611, in _deploy_action
    package_def = self._get_package_def(package_def)
  File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 1430, in _get_package_def
    d['package_obj'] = self.get('package', **def_search)[0]
  File "/Users/jolsen/gh/pytan/lib/pytan/utils.py", line 2699, in wrap
    ret = f(*args, **kwargs)
  File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 1255, in get
    return self._get_single(obj_map, **kwargs)
  File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 1390, in _get_single
    for x in self._single_find(obj_map, k, v, **kwargs):
  File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 1400, in _single_find
    obj_ret = self._find(api_obj_single, **kwargs)
  File "/Users/jolsen/gh/pytan/lib/pytan/utils.py", line 2699, in wrap
    ret = f(*args, **kwargs)
  File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 1327, in _find
    raise pytan.exceptions.HandlerError(err(search_str))
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

```
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['report_dir'] = tempfile.gettempdir()
kwargs["options_help"] = True


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

Output from Python Code

```
Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
Traceback (most recent call last):
  File "<string>", line 55, in <module>
  File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 380, in deploy_action
    raise pytan.exceptions.PytanHelp(pytan.help.help_options())
PytanHelp:
Options Help
============
```

```
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"
```

```
        Example for question: "value_type:value_type"

    'and'
        Help: Use 'and' for all of the filters supplied
        Usable on: group
        Example for sensor: "Sensor1, opt:and"
        Example for question: "and"

    'or'
        Help: Use 'or' for all of the filters supplied
        Usable on: group
        Example for sensor: "Sensor1, opt:or"
        Example for question: "or"

    'ignore_case'
        Help: Make the filter do a case insensitive match
        Usable on: filter
        Example for sensor: "Sensor1, opt:ignore_case"
        Example for question: "ignore_case"

    'match_case'
        Help: Make the filter do a case sensitive match
        Usable on: filter
        Example for sensor: "Sensor1, opt:match_case"
        Example for question: "match_case"

    'match_any_value'
        Help: Make the filter match any value
        Usable on: filter
        Example for sensor: "Sensor1, opt:match_any_value"
        Example for question: "match_any_value"

    'match_all_values'
        Help: Make the filter match all values
        Usable on: filter
        Example for sensor: "Sensor1, opt:match_all_values"
        Example for question: "match_all_values"

    'max_data_age'
        Help: Re-fetch cached values older than N seconds
        Usable on: filter
        VALUE description and type: seconds, <type 'int'>
        Example for sensor: "Sensor1, opt:max_data_age:seconds"
        Example for question: "max_data_age:seconds"

    'value_type'
        Help: Make the filter consider the value type as VALUE_TYPE
        Usable on: filter
        VALUE description and type: value_type, <type 'str'>
        Example for sensor: "Sensor1, opt:value_type:value_type"
        Example for question: "value_type:value_type"

    'and'
        Help: Use 'and' for all of the filters supplied
        Usable on: group
        Example for sensor: "Sensor1, opt:and"
        Example for question: "and"

```

```
    'or'
        Help: Use 'or' for all of the filters supplied
        Usable on: group
        Example for sensor: "Sensor1, opt:or"
        Example for question: "or"
```

Invalid deploy action empty package

Deploy an action using an empty package string.

Example Python Code

```
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['report_dir'] = tempfile.gettempdir()
kwargs["run"] = True
kwargs["package"] = u''


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

Output from Python Code

```
Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
Traceback (most recent call last):
  File "<string>", line 56, in <module>
  File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 398, in deploy_action
    package_def = pytan.utils.dehumanize_package(package)
  File "/Users/jolsen/gh/pytan/lib/pytan/utils.py", line 1508, in dehumanize_package
    raise pytan.exceptions.HumanParserError(err(package))
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

```
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['report_dir'] = tempfile.gettempdir()
kwargs["filters_help"] = True


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

Output from Python Code

```
Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
Traceback (most recent call last):
  File "<string>", line 55, in <module>
  File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 377, in deploy_action
    raise pytan.exceptions.PytanHelp(pytan.help.help_filters())
PytanHelp:
Filters Help
============

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

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

```

```
Valid Filters
-------------

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

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

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

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

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

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

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

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

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

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

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

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

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

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

```
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'
```

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

```
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'
```

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

    'greater than'
```

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

```
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'
```

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

```
543
544     'nre'
545         Help: Filter for non regular expression match for VALUE
546         Example: "Sensor1, that nre:VALUE"
547
548     'is'
549         Help: Filter for regular expression match for VALUE
550         Example: "Sensor1, that is:VALUE"
551
552     'regex'
553         Help: Filter for regular expression match for VALUE
554         Example: "Sensor1, that regex:VALUE"
555
556     'regex match'
557         Help: Filter for regular expression match for VALUE
558         Example: "Sensor1, that regex match:VALUE"
559
560     'regexmatch'
561         Help: Filter for regular expression match for VALUE
562         Example: "Sensor1, that regexmatch:VALUE"
563
564     're'
565         Help: Filter for regular expression match for VALUE
566         Example: "Sensor1, that re:VALUE"
```

Invalid deploy action missing parameters

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

Example Python Code

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

```
  "restrict": null,
  "validationExpressions": [
    {
      "expression": "\\S",
      "flags": "",
      "helpString": "You must enter a value",
      "model": "com.tanium.models::ValidationExpression",
      "parameterType": "com.tanium.models::ValidationExpression"
    }
  ],
  "value": ""
}
```

PyTan API Valid Create Object Examples

Create user

Create a user called API Test User

Example Python Code

```
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 delete method (to remove the package in case it exists)
delete_kwargs = {}
delete_kwargs["objtype"] = 'user'
delete_kwargs["name"] = 'API Test User'


# setup the arguments for the handler method
kwargs = {}
kwargs["username"] = u'API Test User'
kwargs["rolename"] = u'Administrator'
kwargs["properties"] = [[u'property1', u'value1']]

# delete the object in case it already exists
try:
    handler.delete(**delete_kwargs)
except Exception as e:
    print e

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


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

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

print ""
print "print the object returned in JSON format:"
print response.to_json(response)

# delete the object, we are done with it now
try:
    handler.delete(**delete_kwargs)
except Exception as e:
    print e
```

Output from Python Code

```
Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
No results found searching for user with {'name': 'API Test User'}!!
2015-08-07 19:46:14,085 INFO     pytan.handler: New user 'API Test User' created with ID 15, roles:

Type of response:  <class 'taniumpy.object_types.user.User'>

print of response:
```

```
User, name: 'API Test User', id: 15

print the object returned in JSON format:
{
  "_type": "user",
  "deleted_flag": 0,
  "group_id": 0,
  "id": 15,
  "last_login": "2001-01-01T00:00:00",
  "local_admin_flag": -1,
  "metadata": {
    "_type": "metadata",
    "item": [
      {
        "_type": "item",
        "admin_flag": 0,
        "name": "TConsole.User.Property.property1",
        "value": "value1"
      }
    ]
  },
  "name": "API Test User",
  "permissions": {
    "_type": "permissions",
    "permission": [
      "admin",
      "sensor_read",
      "sensor_write",
      "question_read",
      "question_write",
      "action_read",
      "action_write",
      "action_approval",
      "notification_write",
      "clients_read",
      "question_log_read",
      "content_admin"
    ]
  },
  "roles": {
    "_type": "roles",
    "role": [
      {
        "_type": "role",
        "description": "Administrators can perform all functions in the system, including creating c
        "id": 1,
        "name": "Administrator",
        "permissions": {
          "_type": "permissions",
          "permission": [
            "admin",
            "sensor_read",
            "sensor_write",
            "question_read",
            "question_write",
            "action_read",
            "action_write",
            "action_approval",
```

```
            "notification_write",
            "clients_read",
            "question_log_read",
            "content_admin"
          ]
        }
      }
    ]
  }
}
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

```
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 delete method (to remove the package in case it exists)
delete_kwargs = {}
delete_kwargs["objtype"] = 'package'
delete_kwargs["name"] = 'package49'


# setup the arguments for the handler method
kwargs = {}
kwargs["expire_seconds"] = 1500
kwargs["display_name"] = u'package49 API test'
kwargs["name"] = u'package49'
kwargs["parameters_json_file"] = u'../doc/example_of_all_package_parameters.json'
kwargs["verify_expire_seconds"] = 3600
kwargs["command"] = u'package49 $1 $2 $3 $4 $5 $6 $7 $8'
kwargs["file_urls"] = [u'3600::testing.vbs||https://content.tanium.com/files/initialcontent/bundles/
kwargs["verify_filter_options"] = [u'and']
kwargs["verify_filters"] = [u'Custom Tags, that contains:tag']
kwargs["command_timeout_seconds"] = 9999

# delete the object in case it already exists
try:
    handler.delete(**delete_kwargs)
except Exception as e:
    print e

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


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

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

print ""
print "print the object returned in JSON format:"
print response.to_json(response)

# delete the object, we are done with it now
try:
    handler.delete(**delete_kwargs)
except Exception as e:
    print e
```

Output from Python Code

```
Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
No results found searching for PackageSpec, name: 'package49'!!
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
```

```
  },
  "verify_group_id": 211
}
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

```
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 delete method (to remove the package in case it exists)
```

```
delete_kwargs = {}
delete_kwargs["objtype"] = 'group'
delete_kwargs["name"] = 'All Windows Computers API Test'


# setup the arguments for the handler method
kwargs = {}
kwargs["groupname"] = u'All Windows Computers API Test'
kwargs["filters"] = [u'Operating System, that contains:Windows']
kwargs["filter_options"] = [u'and']

# delete the object in case it already exists
try:
    handler.delete(**delete_kwargs)
except Exception as e:
    print e

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


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

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

print ""
print "print the object returned in JSON format:"
print response.to_json(response)

# delete the object, we are done with it now
try:
    handler.delete(**delete_kwargs)
except Exception as e:
    print e
```

Output from Python Code

```
Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
No results found searching for Group, name: 'All Windows Computers API Test'!!
2015-08-07 19:46:14,211 INFO     pytan.handler: New group 'All Windows Computers API Test' created w

Type of response:  <class 'taniumpy.object_types.group.Group'>

print of response:
Group, name: 'All Windows Computers API Test', id: 212

print the object returned in JSON format:
{
  "_type": "group",
  "and_flag": 1,
  "deleted_flag": 0,
  "filters": {
    "_type": "filters",
    "filter": [
```

```
      {
        "_type": "filter",
        "all_times_flag": 0,
        "all_values_flag": 0,
        "delimiter_index": 0,
        "ignore_case_flag": 1,
        "max_age_seconds": 0,
        "not_flag": 0,
        "operator": "RegexMatch",
        "sensor": {
          "_type": "sensor",
          "hash": 45421433
        },
        "substring_flag": 0,
        "substring_length": 0,
        "substring_start": 0,
        "utf8_flag": 0,
        "value": ".*Windows.*",
        "value_type": "String"
      }
    ]
  },
  "id": 212,
  "name": "All Windows Computers API Test",
  "not_flag": 0,
  "sub_groups": {
    "_type": "groups",
    "group": []
  },
  "text": " Operating System containing \"Windows\"",
  "type": 0
}
2015-08-07 19:46:14,219 INFO     pytan.handler: Deleted 'Group, id: 212'
```

Create whitelisted url

Create a whitelisted url

Example Python Code

```
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 delete method (to remove the package in case it exists)
delete_kwargs = {}
delete_kwargs["objtype"] = 'whitelisted_url'
delete_kwargs["url_regex"] = 'regex:http://test.com/.*API_Test.*URL'


# setup the arguments for the handler method
kwargs = {}
kwargs["url"] = u'http://test.com/.*API_Test.*URL'
kwargs["regex"] = True
kwargs["properties"] = [[u'property1', u'value1']]
kwargs["download_seconds"] = 3600

# delete the object in case it already exists
try:
    handler.delete(**delete_kwargs)
except Exception as e:
    print e

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


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

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

```

```
print ""
print "print the object returned in JSON format:"
print response.to_json(response)

# delete the object, we are done with it now
try:
    handler.delete(**delete_kwargs)
except Exception as e:
    print e
```

Output from Python Code

```
Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
No results found searching for whitelisted_url with {'url_regex': 'regex:http://test.com/.*API_Test.
2015-08-07 19:46:14,252 INFO     pytan.handler: New Whitelisted URL 'regex:http://test.com/.*API_Tes

Type of response:  <class 'taniumpy.object_types.white_listed_url.WhiteListedUrl'>

print of response:
WhiteListedUrl, id: 52

print the object returned in JSON format:
{
  "_type": "white_listed_url",
  "download_seconds": 3600,
  "id": 52,
  "metadata": {
    "_type": "metadata",
    "item": [
      {
        "_type": "item",
        "admin_flag": 0,
        "name": "TConsole.WhitelistedURL.property1",
        "value": "value1"
      }
    ]
  },
  "url_regex": "regex:http://test.com/.*API_Test.*URL"
}
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

```
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 = {}


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

Output from Python Code

```
Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
Traceback (most recent call last):
  File "<string>", line 53, in <module>
  File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 708, in create_sensor
    raise pytan.exceptions.HandlerError(m)
```

```
HandlerError: Sensor creation not supported via PyTan as of yet, too complex
Use create_sensor_from_json() instead!
```

PyTan API Valid Create Object From JSON Examples

Create package from json

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

Example Python Code

```
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 # 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

```
Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2015-08-07 19:46:14,304 INFO     pytan.handler: Deleted 'PackageSpec, id: 76'
2015-08-07 19:46:14,305 INFO     pytan.handler: Report file '/var/folders/dk/vjr1r_c53yx6k6gzp2bbt_c
2015-08-07 19:46:14,337 INFO     pytan.handler: New PackageSpec, name: 'Disable Java Auto Update API

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

print of response:
PackageSpecList, len: 1

print the object returned in JSON format:
{
  "_type": "package_specs",
  "package_spec": [
    {
      "_type": "package_spec",
      "available_time": "2015-08-07T13:22:40",
      "command": "cmd /c cscript //T:60 disable-java-auto-update.vbs",
      "command_timeout": 60,
      "creation_time": "2001-01-01T00:00:00",
      "deleted_flag": 0,
      "display_name": "Disable Java Auto Update",
      "expire_seconds": 660,
      "files": {
        "_type": "package_files",
        "file": [
          {
            "_type": "file",
            "bytes_downloaded": 0,
            "bytes_total": 0,
            "cache_status": "CACHED",
            "download_seconds": 0,
            "file_status": {
              "_type": "file_status",
              "status": [
                {
                  "_type": "status",
                  "bytes_downloaded": 0,
                  "bytes_total": 0,
                  "cache_status": "Processing",
                  "server_id": 1,
                  "server_name": "JTANIUM1.localdomain:17472",
                  "status": 0
                }
              ]
            },
            "hash": "9e36208ce643c767ad76ef2ad6a69141fbb5a59a607b8eb8065db09e3a153c0d",
            "id": 43,
            "name": "disable-java-auto-update.vbs",
            "size": 11377,
            "source": "https://content.tanium.com/files/published/InitialContent/2015-06-04_18-59-45
            "status": 0
          }
        ]
      },
      "hidden_flag": 0,
      "id": 83,
```

```
      "last_update": "2001-01-01T00:00:00",
      "metadata": {
        "_type": "metadata",
        "item": [
          {
            "_type": "item",
            "admin_flag": 0,
            "name": "defined",
            "value": "Tanium"
          },
          {
            "_type": "item",
            "admin_flag": 0,
            "name": "category",
            "value": "Tanium"
          }
        ]
      },
      "modification_time": "2001-01-01T00:00:00",
      "name": "Disable Java Auto Update API TEST",
      "skip_lock_flag": 0,
      "source_id": 0,
      "verify_expire_seconds": 600,
      "verify_group": {
        "_type": "group",
        "id": 0
      },
      "verify_group_id": 0
    }
  ]
}
```

Create user from json

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

Example Python Code

```
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

# set the attribute name and value we want to add to the original object (if any)
attr_name = "name"
attr_add = " API TEST"

# delete object before creating it?
delete = True

# setup the arguments for getting an object to export as json file
get_kwargs = {}
get_kwargs["objtype"] = u'user'
get_kwargs["id"] = 1


# get objects to use as an export to JSON file
orig_objs = handler.get(**get_kwargs)

# if attr_name and attr_add exists, modify the orig_objs to add attr_add to the attribute
# attr_name
if attr_name:
    for x in orig_objs:
        new_attr = getattr(x, attr_name)
        new_attr += attr_add
        setattr(x, attr_name, new_attr)
        if delete:
            # delete the object in case it already exists
            del_kwargs = {}
            del_kwargs[attr_name] = new_attr
            del_kwargs['objtype'] = u'user'
            try:
                handler.delete(**del_kwargs)
            except Exception as e:
```

```
                print e

# export orig_objs to a json file
json_file, results = handler.export_to_report_file(
    obj=orig_objs,
    export_format='json',
    report_dir=tempfile.gettempdir(),
)

# create the object from the exported JSON file
create_kwargs = {'objtype': u'user', 'json_file': json_file}
response = handler.create_from_json(**create_kwargs)


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

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

print ""
print "print the object returned in JSON format:"
print response.to_json(response)
```

Output from Python Code

```
Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2015-08-07 19:46:14,358 INFO     pytan.handler: Deleted "User, name: 'Jim Olsen API TEST', id: 12"
2015-08-07 19:46:14,359 INFO     pytan.handler: Report file '/var/folders/dk/vjr1r_c53yx6k6gzp2bbt_c
2015-08-07 19:46:14,380 INFO     pytan.handler: New User, name: 'Jim Olsen API TEST', id: 16 (ID: 16

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

print of response:
UserList, len: 1

print the object returned in JSON format:
{
  "_type": "users",
  "user": [
    {
      "_type": "user",
      "deleted_flag": 0,
      "group_id": 0,
      "id": 16,
      "last_login": "2001-01-01T00:00:00",
      "local_admin_flag": -1,
      "name": "Jim Olsen API TEST",
      "permissions": {
        "_type": "permissions",
        "permission": [
          "admin",
          "sensor_read",
          "sensor_write",
          "question_read",
          "question_write",
```

```
            "action_read",
            "action_write",
            "action_approval",
            "notification_write",
            "clients_read",
            "question_log_read",
            "content_admin"
          ]
        },
        "roles": {
          "_type": "roles",
          "role": [
            {
              "_type": "role",
              "description": "Administrators can perform all functions in the system, including creati
              "id": 1,
              "name": "Administrator",
              "permissions": {
                "_type": "permissions",
                "permission": [
                  "admin",
                  "sensor_read",
                  "sensor_write",
                  "question_read",
                  "question_write",
                  "action_read",
                  "action_write",
                  "action_approval",
                  "notification_write",
                  "clients_read",
                  "question_log_read",
                  "content_admin"
                ]
              }
            }
          ]
        }
      }
    ]
}
```

Create saved question from json

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

Example Python Code

```
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

# set the attribute name and value we want to add to the original object (if any)
attr_name = "name"
attr_add = " API TEST"

# delete object before creating it?
delete = True

# setup the arguments for getting an object to export as json file
get_kwargs = {}
get_kwargs["objtype"] = u'saved_question'
get_kwargs["id"] = 1


# get objects to use as an export to JSON file
orig_objs = handler.get(**get_kwargs)

# if attr_name and attr_add exists, modify the orig_objs to add attr_add to the attribute
# attr_name
if attr_name:
    for x in orig_objs:
        new_attr = getattr(x, attr_name)
```

```
        new_attr += attr_add
        setattr(x, attr_name, new_attr)
        if delete:
            # delete the object in case it already exists
            del_kwargs = {}
            del_kwargs[attr_name] = new_attr
            del_kwargs['objtype'] = u'saved_question'
            try:
                handler.delete(**del_kwargs)
            except Exception as e:
                print e

# export orig_objs to a json file
json_file, results = handler.export_to_report_file(
    obj=orig_objs,
    export_format='json',
    report_dir=tempfile.gettempdir(),
)

# create the object from the exported JSON file
create_kwargs = {'objtype': u'saved_question', 'json_file': json_file}
response = handler.create_from_json(**create_kwargs)


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

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

print ""
print "print the object returned in JSON format:"
print response.to_json(response)
```

Output from Python Code

```
Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2015-08-07 19:46:14,432 INFO     pytan.handler: Deleted 'SavedQuestion, id: 109'
2015-08-07 19:46:14,433 INFO     pytan.handler: Report file '/var/folders/dk/vjr1r_c53yx6k6gzp2bbt_c
2015-08-07 19:46:14,460 INFO     pytan.handler: New SavedQuestion, name: 'Has Tanium Standard Utilit

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

print of response:
SavedQuestionList, len: 1

print the object returned in JSON format:
{
  "_type": "saved_questions",
  "saved_question": [
    {
      "_type": "saved_question",
      "action_tracking_flag": 0,
      "archive_enabled_flag": 0,
      "archive_owner": {
        "_type": "user"
```

```
    },
    "expire_seconds": 600,
    "hidden_flag": 0,
    "id": 111,
    "issue_seconds": 120,
    "issue_seconds_never_flag": 0,
    "keep_seconds": 0,
    "mod_time": "2015-08-07T19:46:14",
    "mod_user": {
      "_type": "user",
      "name": "Tanium User"
    },
    "most_recent_question_id": 1256,
    "name": "Has Tanium Standard Utilities API TEST",
    "packages": {
      "_type": "package_specs",
      "package_spec": [
        {
          "_type": "package_spec",
          "id": 20,
          "name": "Distribute Tanium Standard Utilities"
        }
      ]
    },
    "public_flag": 1,
    "query_text": "Get Has Tanium Standard Utilities from all machines",
    "question": {
      "_type": "question",
      "action_tracking_flag": 0,
      "expiration": "2015-08-07T19:32:37",
      "expire_seconds": 0,
      "force_computer_id_flag": 0,
      "hidden_flag": 0,
      "id": 1256,
      "management_rights_group": {
        "_type": "group",
        "id": 0
      },
      "query_text": "Get Has Tanium Standard Utilities from all machines",
      "saved_question": {
        "_type": "saved_question",
        "id": 110
      },
      "selects": {
        "_type": "selects",
        "select": [
          {
            "_type": "select",
            "filter": {
              "_type": "filter",
              "all_times_flag": 0,
              "all_values_flag": 0,
              "delimiter_index": 0,
              "end_time": "2001-01-01T00:00:00",
              "ignore_case_flag": 1,
              "max_age_seconds": 0,
              "not_flag": 0,
              "operator": "Less",
```

```
      "start_time": "2001-01-01T00:00:00",
      "substring_flag": 0,
      "substring_length": 0,
      "substring_start": 0,
      "utf8_flag": 0,
      "value_type": "String"
    },
    "sensor": {
      "_type": "sensor",
      "category": "Tanium",
      "creation_time": "2015-08-07T13:22:09",
      "delimiter": ",",
      "description": "Returns whether a machine has the Tanium Standard Utilities\nExample
      "exclude_from_parse_flag": 1,
      "hash": 1782389954,
      "hidden_flag": 0,
      "id": 194,
      "ignore_case_flag": 1,
      "last_modified_by": "Jim Olsen",
      "max_age_seconds": 900,
      "modification_time": "2015-08-07T13:22:09",
      "name": "Has Tanium Standard Utilities",
      "queries": {
        "_type": "queries",
        "query": [
          {
            "_type": "query",
            "platform": "Windows",
            "script": "&#039;=========================================\n&#039; Has Tanium S
            "script_type": "VBScript"
          },
          {
            "_type": "query",
            "platform": "Linux",
            "script": "#!/bin/sh\n\n# THIS IS A STUB - NOT INTENDED AS FUNCTIONAL - NA\n#
            "script_type": "UnixShell"
          },
          {
            "_type": "query",
            "platform": "Mac",
            "script": "#!/bin/sh\n\n# THIS IS A STUB - NOT INTENDED AS FUNCTIONAL - NA\n#
            "script_type": "UnixShell"
          },
          {
            "_type": "query",
            "platform": "Solaris",
            "script": "#!/bin/sh\n\n# THIS IS A STUB - NOT INTENDED AS FUNCTIONAL - NA\n#
            "script_type": "UnixShell"
          },
          {
            "_type": "query",
            "platform": "AIX",
            "script": "#!/bin/sh\n\n# THIS IS A STUB - NOT INTENDED AS FUNCTIONAL - NA\n#
            "script_type": "UnixShell"
          }
        ]
      },
      "source_id": 0,
```

```
                "string_count": 16,
                "value_type": "String"
              }
            }
          ]
        },
        "skip_lock_flag": 0,
        "user": {
          "_type": "user",
          "id": 1,
          "name": "Jim Olsen"
        }
      },
      "row_count_flag": 0,
      "sort_column": 0,
      "user": {
        "_type": "user",
        "id": 2,
        "name": "Tanium User"
      }
    }
  ]
}
```

Create action from json

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

Example Python Code

```
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"
```

```
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",
```

```
        "command": "cmd /c cscript install-standard-utils.vbs \"Tools\\StdUtils\"",
        "id": 20,
        "name": "Distribute Tanium Standard Utilities"
      },
      "saved_action": {
        "_type": "saved_action",
        "id": 46
      },
      "skip_lock_flag": 0,
      "start_time": "2015-08-07T19:46:16",
      "status": "Open",
      "stopped_flag": 0,
      "target_group": {
        "_type": "group",
        "id": 37,
        "name": "Default"
      },
      "user": {
        "_type": "user",
        "group_id": 0,
        "id": 2,
        "last_login": "2015-08-07T19:46:14",
        "name": "Tanium User"
      }
    }
  ]
}
```

Create sensor from json

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

Example Python Code

```
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

# set the attribute name and value we want to add to the original object (if any)
attr_name = "name"
attr_add = " API TEST"

# delete object before creating it?
delete = True

# setup the arguments for getting an object to export as json file
get_kwargs = {}
get_kwargs["objtype"] = u'sensor'
get_kwargs["id"] = 381


# get objects to use as an export to JSON file
orig_objs = handler.get(**get_kwargs)

# if attr_name and attr_add exists, modify the orig_objs to add attr_add to the attribute
# attr_name
if attr_name:
    for x in orig_objs:
        new_attr = getattr(x, attr_name)
        new_attr += attr_add
        setattr(x, attr_name, new_attr)
        if delete:
            # delete the object in case it already exists
            del_kwargs = {}
            del_kwargs[attr_name] = new_attr
            del_kwargs['objtype'] = u'sensor'
            try:
                handler.delete(**del_kwargs)
            except Exception as e:
                print e

# export orig_objs to a json file
```

```
json_file, results = handler.export_to_report_file(
    obj=orig_objs,
    export_format='json',
    report_dir=tempfile.gettempdir(),
)

# create the object from the exported JSON file
create_kwargs = {'objtype': u'sensor', 'json_file': json_file}
response = handler.create_from_json(**create_kwargs)


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

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

print ""
print "print the object returned in JSON format:"
print response.to_json(response)
```

Output from Python Code

```
Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2015-08-07 19:46:14,551 INFO     pytan.handler: Deleted 'Sensor, id: 639'
2015-08-07 19:46:14,551 INFO     pytan.handler: Report file '/var/folders/dk/vjr1r_c53yx6k6gzp2bbt_c
2015-08-07 19:46:14,577 INFO     pytan.handler: New Sensor, name: 'Is Mac API TEST', id: 642 (ID: 64

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

print of response:
SensorList, len: 1

print the object returned in JSON format:
{
  "_type": "sensors",
  "sensor": [
    {
      "_type": "sensor",
      "category": "Operating System",
      "creation_time": "2015-08-07T19:46:14",
      "delimiter": ",",
      "description": "Returns whether the machine is a Mac.  True if so, False if not.\nExample: Tru
      "exclude_from_parse_flag": 0,
      "hash": 2387245230,
      "hidden_flag": 0,
      "id": 642,
      "ignore_case_flag": 1,
      "last_modified_by": "Tanium User",
      "max_age_seconds": 86400,
      "modification_time": "2015-08-07T19:46:14",
      "name": "Is Mac API TEST",
      "queries": {
        "_type": "queries",
        "query": [
          {
```

```
            "_type": "query",
            "platform": "Windows",
            "script": "&#039;=======================================\n&#039; Is Mac\n&#039;========
            "script_type": "VBScript"
          },
          {
            "_type": "query",
            "platform": "Linux",
            "script": "#!/bin/bash\necho False\n",
            "script_type": "UnixShell"
          },
          {
            "_type": "query",
            "platform": "Mac",
            "script": "#!/bin/bash\necho True\n",
            "script_type": "UnixShell"
          },
          {
            "_type": "query",
            "platform": "Solaris",
            "script": "#!/bin/sh\n\n# THIS IS A STUB - NOT INTENDED AS FUNCTIONAL - NA\n# \n# \n\nec
            "script_type": "UnixShell"
          },
          {
            "_type": "query",
            "platform": "AIX",
            "script": "#!/bin/sh\n\n# THIS IS A STUB - NOT INTENDED AS FUNCTIONAL - NA\n# \n# \n\nec
            "script_type": "UnixShell"
          }
        ]
      },
      "source_id": 0,
      "string_count": 0,
      "value_type": "String"
    }
  ]
}
```

Create question from json

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

Example Python Code

```
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

# set the attribute name and value we want to add to the original object (if any)
attr_name = ""
attr_add = ""

# delete object before creating it?
delete = False

# setup the arguments for getting an object to export as json file
get_kwargs = {}
get_kwargs["objtype"] = u'question'
get_kwargs["id"] = 1


# get objects to use as an export to JSON file
orig_objs = handler.get(**get_kwargs)

# if attr_name and attr_add exists, modify the orig_objs to add attr_add to the attribute
# attr_name
if attr_name:
    for x in orig_objs:
        new_attr = getattr(x, attr_name)
        new_attr += attr_add
        setattr(x, attr_name, new_attr)
        if delete:
            # delete the object in case it already exists
```

```
            del_kwargs = {}
            del_kwargs[attr_name] = new_attr
            del_kwargs['objtype'] = u'question'
            try:
                handler.delete(**del_kwargs)
            except Exception as e:
                print e

# export orig_objs to a json file
json_file, results = handler.export_to_report_file(
    obj=orig_objs,
    export_format='json',
    report_dir=tempfile.gettempdir(),
)

# create the object from the exported JSON file
create_kwargs = {'objtype': u'question', 'json_file': json_file}
response = handler.create_from_json(**create_kwargs)


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

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

print ""
print "print the object returned in JSON format:"
print response.to_json(response)
```

Output from Python Code

```
Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2015-08-07 19:46:14,608 INFO     pytan.handler: Report file '/var/folders/dk/vjr1r_c53yx6k6gzp2bbt_c
2015-08-07 19:46:14,640 INFO     pytan.handler: New Question, id: 1305 (ID: 1305) created successful

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

print of response:
QuestionList, len: 1

print the object returned in JSON format:
{
  "_type": "questions",
  "question": [
    {
      "_type": "question",
      "action_tracking_flag": 0,
      "context_group": {
        "_type": "group",
        "id": 0
      },
      "expiration": "2015-08-07T19:56:14",
      "expire_seconds": 0,
      "force_computer_id_flag": 1,
      "hidden_flag": 0,
```

```
      "id": 1305,
      "management_rights_group": {
        "_type": "group",
        "id": 0
      },
      "query_text": "Get Action Statuses matching \"Nil\" from all machines",
      "saved_question": {
        "_type": "saved_question",
        "id": 0
      },
      "selects": {
        "_type": "selects",
        "select": [
          {
            "_type": "select",
            "filter": {
              "_type": "filter",
              "all_times_flag": 0,
              "all_values_flag": 1,
              "delimiter_index": 0,
              "end_time": "2001-01-01T00:00:00",
              "ignore_case_flag": 1,
              "max_age_seconds": 0,
              "not_flag": 0,
              "operator": "RegexMatch",
              "start_time": "2001-01-01T00:00:00",
              "substring_flag": 0,
              "substring_length": 0,
              "substring_start": 0,
              "utf8_flag": 0,
              "value": "Nil",
              "value_type": "String"
            },
            "sensor": {
              "_type": "sensor",
              "category": "Reserved",
              "description": "The recorded state of each action a client has taken recently in the f
              "exclude_from_parse_flag": 1,
              "hash": 1792443391,
              "hidden_flag": 0,
              "id": 1,
              "ignore_case_flag": 1,
              "max_age_seconds": 3600,
              "name": "Action Statuses",
              "queries": {
                "_type": "queries",
                "query": [
                  {
                    "_type": "query",
                    "platform": "Windows",
                    "script": "Reserved",
                    "script_type": "WMIQuery"
                  }
                ]
              },
              "source_id": 0,
              "string_count": 238,
              "value_type": "String"
```

```
            }
          }
        ]
      },
      "skip_lock_flag": 0,
      "user": {
        "_type": "user",
        "id": 2,
        "name": "Tanium User"
      }
    }
  ]
}
```

Create whitelisted url from json

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

Example Python Code

```
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

# set the attribute name and value we want to add to the original object (if any)
attr_name = "url_regex"
attr_add = " API TEST"

# delete object before creating it?
delete = True

# setup the arguments for getting an object to export as json file
get_kwargs = {}
get_kwargs["objtype"] = u'whitelisted_url'
get_kwargs["url_regex"] = u'test1'


# get objects to use as an export to JSON file
orig_objs = handler.get(**get_kwargs)

# if attr_name and attr_add exists, modify the orig_objs to add attr_add to the attribute
# attr_name
if attr_name:
    for x in orig_objs:
        new_attr = getattr(x, attr_name)
        new_attr += attr_add
        setattr(x, attr_name, new_attr)
        if delete:
            # delete the object in case it already exists
            del_kwargs = {}
            del_kwargs[attr_name] = new_attr
            del_kwargs['objtype'] = u'whitelisted_url'
            try:
                handler.delete(**del_kwargs)
            except Exception as e:
                print e

# export orig_objs to a json file
json_file, results = handler.export_to_report_file(
    obj=orig_objs,
    export_format='json',
    report_dir=tempfile.gettempdir(),
)

# create the object from the exported JSON file
create_kwargs = {'objtype': u'whitelisted_url', 'json_file': json_file}
response = handler.create_from_json(**create_kwargs)


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

```

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

print ""
print "print the object returned in JSON format:"
print response.to_json(response)
```

Output from Python Code

```
Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2015-08-07 19:46:14,685 INFO     pytan.handler: Deleted 'WhiteListedUrl, id: 27'
2015-08-07 19:46:14,686 INFO     pytan.handler: Report file '/var/folders/dk/vjr1r_c53yx6k6gzp2bbt_c
2015-08-07 19:46:14,695 INFO     pytan.handler: New WhiteListedUrl, id: 53 (ID: 53) created successf

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

print of response:
WhiteListedUrlList, len: 1

print the object returned in JSON format:
{
  "_type": "white_listed_urls",
  "white_listed_url": [
    {
      "_type": "white_listed_url",
      "download_seconds": 86400,
      "id": 53,
      "url_regex": "test1 API TEST"
    }
  ]
}
```

Create group from json

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

Example Python Code

```
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]
```

```
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'
```

```
            try:
                handler.delete(**del_kwargs)
            except Exception as e:
                print e

# export orig_objs to a json file
json_file, results = handler.export_to_report_file(
    obj=orig_objs,
    export_format='json',
    report_dir=tempfile.gettempdir(),
)

# create the object from the exported JSON file
create_kwargs = {'objtype': u'group', 'json_file': json_file}
response = handler.create_from_json(**create_kwargs)


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

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

print ""
print "print the object returned in JSON format:"
print response.to_json(response)
```

Output from Python Code

```
Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2015-08-07 19:46:14,746 INFO     pytan.handler: Deleted 'Group, id: 157'
2015-08-07 19:46:14,747 INFO     pytan.handler: Report file '/var/folders/dk/vjr1r_c53yx6k6gzp2bbt_c
2015-08-07 19:46:14,769 INFO     pytan.handler: New Group, name: 'All Computers API TEST', id: 213 (

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

print of response:
GroupList, len: 1

print the object returned in JSON format:
{
  "_type": "groups",
  "group": [
    {
      "_type": "group",
      "and_flag": 0,
      "deleted_flag": 0,
      "filters": {
        "_type": "filters",
        "filter": []
      },
      "id": 213,
      "name": "All Computers API TEST",
      "not_flag": 0,
      "sub_groups": {
        "_type": "groups",
```

```
        "group": []
      },
      "type": 0
    }
  ]
}
```

PyTan API Invalid Create Object From JSON Examples

Invalid create saved action from json

Create a saved action from json (not supported!)

Example Python Code

```
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 getting an object to export as json file
get_kwargs = {}
get_kwargs["objtype"] = u'saved_action'
get_kwargs["name"] = u'Distribute Tanium Standard Utilities'

# get objects to use as an export to JSON file
orig_objs = handler.get(**get_kwargs)

# export orig_objs to a json file
json_file, results = handler.export_to_report_file(
    obj=orig_objs,
    export_format='json',
    report_dir=tempfile.gettempdir(),
)

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

# create the object from the exported JSON file
create_kwargs = {'objtype': u'saved_action', 'json_file': json_file}
try:
    response = handler.create_from_json(**create_kwargs)
except Exception as e:
    traceback.print_exc(file=sys.stdout)
```

Output from Python Code

```
Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2015-08-07 19:46:14,794 INFO     pytan.handler: Report file '/var/folders/dk/vjr1r_c53yx6k6gzp2bbt_c
Traceback (most recent call last):
  File "<string>", line 67, in <module>
  File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 546, in create_from_json
    raise pytan.exceptions.HandlerError(m(objtype, json_createable))
HandlerError: saved_action is not a json createable object! Supported objects: user, whitelisted_url
```

Invalid create client from json

Create a client from json (not supported!)

Example Python Code

```
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 getting an object to export as json file
get_kwargs = {}
get_kwargs["objtype"] = u'client'
get_kwargs["status"] = u'Leader'

# get objects to use as an export to JSON file
orig_objs = handler.get(**get_kwargs)

# export orig_objs to a json file
json_file, results = handler.export_to_report_file(
    obj=orig_objs,
    export_format='json',
    report_dir=tempfile.gettempdir(),
)

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

# create the object from the exported JSON file
create_kwargs = {'objtype': u'client', 'json_file': json_file}
try:
    response = handler.create_from_json(**create_kwargs)
except Exception as e:
```

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

Output from Python Code

```
Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2015-08-07 19:46:14,805 INFO     pytan.handler: Report file '/var/folders/dk/vjr1r_c53yx6k6gzp2bbt_c
Traceback (most recent call last):
  File "<string>", line 67, in <module>
  File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 546, in create_from_json
    raise pytan.exceptions.HandlerError(m(objtype, json_createable))
HandlerError: client is not a json createable object! Supported objects: user, whitelisted_url, save
```

Invalid create userrole from json

Create a user role from json (not supported!)

Example Python Code

```
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 getting an object to export as json file
get_kwargs = {}
get_kwargs["objtype"] = u'userrole'
get_kwargs["name"] = u'Administrator'

# get objects to use as an export to JSON file
orig_objs = handler.get(**get_kwargs)

# export orig_objs to a json file
json_file, results = handler.export_to_report_file(
    obj=orig_objs,
    export_format='json',
    report_dir=tempfile.gettempdir(),
)

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

# create the object from the exported JSON file
create_kwargs = {'objtype': u'userrole', 'json_file': json_file}
try:
    response = handler.create_from_json(**create_kwargs)
except Exception as e:
    traceback.print_exc(file=sys.stdout)
```

Output from Python Code

```
Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2015-08-07 19:46:14,815 INFO     pytan.handler: Report file '/var/folders/dk/vjr1r_c53yx6k6gzp2bbt_c
Traceback (most recent call last):
  File "<string>", line 67, in <module>
  File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 546, in create_from_json
    raise pytan.exceptions.HandlerError(m(objtype, json_createable))
HandlerError: userrole is not a json createable object! Supported objects: user, whitelisted_url, sa
```

Invalid create setting from json

Create a setting from json (not supported!)

Example Python Code

```
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 getting an object to export as json file
get_kwargs = {}
get_kwargs["objtype"] = u'setting'
get_kwargs["id"] = 1

# get objects to use as an export to JSON file
orig_objs = handler.get(**get_kwargs)

# export orig_objs to a json file
json_file, results = handler.export_to_report_file(
    obj=orig_objs,
    export_format='json',
    report_dir=tempfile.gettempdir(),
)

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

# create the object from the exported JSON file
```

```
create_kwargs = {'objtype': u'setting', 'json_file': json_file}
try:
    response = handler.create_from_json(**create_kwargs)
except Exception as e:
    traceback.print_exc(file=sys.stdout)
```

Output from Python Code

```
Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2015-08-07 19:46:14,843 INFO     pytan.handler: Report file '/var/folders/dk/vjr1r_c53yx6k6gzp2bbt_c
Traceback (most recent call last):
  File "<string>", line 67, in <module>
  File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 546, in create_from_json
    raise pytan.exceptions.HandlerError(m(objtype, json_createable))
HandlerError: setting is not a json createable object! Supported objects: user, whitelisted_url, sav
```

PyTan API Valid Export ResultSet Examples

Export resultset csv default options

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

Example Python Code

```
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'

# ask the question that will provide the resultset that we want to use
ask_kwargs = {
    'qtype': 'manual',
    'sensors': [
        "Computer Name", "IP Route Details", "IP Address",
        'Folder Name Search with RegEx Match{dirname=Program Files,regex=.*Shared.*}',
    ],
}
response = handler.ask(**ask_kwargs)

# export the object to a string
# (we could just as easily export to a file using export_to_report_file)
export_kwargs['obj'] = response['question_results']
export_str = handler.export_obj(**export_kwargs)


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

print out
```

Output from Python Code

```
Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2015-08-07 19:46:14,930 DEBUG    pytan.handler.QuestionPoller: ID 1306: id resolved to 1306
2015-08-07 19:46:14,930 DEBUG    pytan.handler.QuestionPoller: ID 1306: expiration resolved to 2015-
2015-08-07 19:46:14,930 DEBUG    pytan.handler.QuestionPoller: ID 1306: query_text resolved to Get C
2015-08-07 19:46:14,930 DEBUG    pytan.handler.QuestionPoller: ID 1306: id resolved to 1306
2015-08-07 19:46:14,930 DEBUG    pytan.handler.QuestionPoller: ID 1306: Object Info resolved to Ques
2015-08-07 19:46:14,933 DEBUG    pytan.handler.QuestionPoller: ID 1306: Progress: Tested: 0, Passed:
2015-08-07 19:46:14,933 DEBUG    pytan.handler.QuestionPoller: ID 1306: Timing: Started: 2015-08-07
2015-08-07 19:46:14,933 INFO     pytan.handler.QuestionPoller: ID 1306: Progress Changed 0% (0 of 2)
2015-08-07 19:46:19,940 DEBUG    pytan.handler.QuestionPoller: ID 1306: Progress: Tested: 0, Passed:
2015-08-07 19:46:19,940 DEBUG    pytan.handler.QuestionPoller: ID 1306: Timing: Started: 2015-08-07
2015-08-07 19:46:24,944 DEBUG    pytan.handler.QuestionPoller: ID 1306: Progress: Tested: 0, Passed:
2015-08-07 19:46:24,944 DEBUG    pytan.handler.QuestionPoller: ID 1306: Timing: Started: 2015-08-07
```

```
2015-08-07 19:46:29,951 DEBUG    pytan.handler.QuestionPoller: ID 1306: Progress: Tested: 0, Passed:
2015-08-07 19:46:29,951 DEBUG    pytan.handler.QuestionPoller: ID 1306: Timing: Started: 2015-08-07
2015-08-07 19:46:34,955 DEBUG    pytan.handler.QuestionPoller: ID 1306: Progress: Tested: 0, Passed:
2015-08-07 19:46:34,955 DEBUG    pytan.handler.QuestionPoller: ID 1306: Timing: Started: 2015-08-07
2015-08-07 19:46:39,959 DEBUG    pytan.handler.QuestionPoller: ID 1306: Progress: Tested: 1, Passed:
2015-08-07 19:46:39,959 DEBUG    pytan.handler.QuestionPoller: ID 1306: Timing: Started: 2015-08-07
2015-08-07 19:46:39,959 INFO     pytan.handler.QuestionPoller: ID 1306: Progress Changed 50% (1 of 2
2015-08-07 19:46:44,964 DEBUG    pytan.handler.QuestionPoller: ID 1306: Progress: Tested: 1, Passed:
2015-08-07 19:46:44,965 DEBUG    pytan.handler.QuestionPoller: ID 1306: Timing: Started: 2015-08-07
2015-08-07 19:46:49,971 DEBUG    pytan.handler.QuestionPoller: ID 1306: Progress: Tested: 2, Passed:
2015-08-07 19:46:49,971 DEBUG    pytan.handler.QuestionPoller: ID 1306: Timing: Started: 2015-08-07
2015-08-07 19:46:49,971 INFO     pytan.handler.QuestionPoller: ID 1306: Progress Changed 100% (2 of
2015-08-07 19:46:49,971 INFO     pytan.handler.QuestionPoller: ID 1306: Reached Threshold of 99% (2

print the export_str returned from export_obj():
Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2015-08-07 19:46:14,843 INFO     pytan.handler: Report file '/var/folders/dk/vjr1r_c53yx6k6gzp2bbt_c
Traceback (most recent call last):
  File "<string>", line 67, in <module>
  File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 546, in create_from_json
    raise pytan.exceptions.HandlerError(m(objtype, json_createable))
HandlerError: setting is not a json createable object! Supported objects: user, whitelisted_url, sav
```

Export resultset csv expand false

Export a ResultSet from asking a question as CSV with false for expand_grouped_columns

Example Python Code

```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["expand_grouped_columns"] = False

# ask the question that will provide the resultset that we want to use
ask_kwargs = {
    'qtype': 'manual',
    'sensors': [
        "Computer Name", "IP Route Details", "IP Address",
        'Folder Name Search with RegEx Match{dirname=Program Files,regex=.*Shared.*}',
    ],
}
response = handler.ask(**ask_kwargs)

# export the object to a string
# (we could just as easily export to a file using export_to_report_file)
export_kwargs['obj'] = response['question_results']
export_str = handler.export_obj(**export_kwargs)


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

print out
```

Output from Python Code

```
Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2015-08-07 19:46:50,080 DEBUG    pytan.handler.QuestionPoller: ID 1307: id resolved to 1307
2015-08-07 19:46:50,080 DEBUG    pytan.handler.QuestionPoller: ID 1307: expiration resolved to 2015-
2015-08-07 19:46:50,080 DEBUG    pytan.handler.QuestionPoller: ID 1307: query_text resolved to Get C
2015-08-07 19:46:50,080 DEBUG    pytan.handler.QuestionPoller: ID 1307: id resolved to 1307
2015-08-07 19:46:50,080 DEBUG    pytan.handler.QuestionPoller: ID 1307: Object Info resolved to Ques
2015-08-07 19:46:50,083 DEBUG    pytan.handler.QuestionPoller: ID 1307: Progress: Tested: 0, Passed:
2015-08-07 19:46:50,083 DEBUG    pytan.handler.QuestionPoller: ID 1307: Timing: Started: 2015-08-07
```

```
2015-08-07 19:46:50,083 INFO     pytan.handler.QuestionPoller: ID 1307: Progress Changed 0% (0 of 2)
2015-08-07 19:46:55,088 DEBUG    pytan.handler.QuestionPoller: ID 1307: Progress: Tested: 1, Passed:
2015-08-07 19:46:55,088 DEBUG    pytan.handler.QuestionPoller: ID 1307: Timing: Started: 2015-08-07
2015-08-07 19:46:55,088 INFO     pytan.handler.QuestionPoller: ID 1307: Progress Changed 50% (1 of 2
2015-08-07 19:47:00,092 DEBUG    pytan.handler.QuestionPoller: ID 1307: Progress: Tested: 1, Passed:
2015-08-07 19:47:00,092 DEBUG    pytan.handler.QuestionPoller: ID 1307: Timing: Started: 2015-08-07
2015-08-07 19:47:05,096 DEBUG    pytan.handler.QuestionPoller: ID 1307: Progress: Tested: 1, Passed:
2015-08-07 19:47:05,096 DEBUG    pytan.handler.QuestionPoller: ID 1307: Timing: Started: 2015-08-07
2015-08-07 19:47:10,102 DEBUG    pytan.handler.QuestionPoller: ID 1307: Progress: Tested: 2, Passed:
2015-08-07 19:47:10,102 DEBUG    pytan.handler.QuestionPoller: ID 1307: Timing: Started: 2015-08-07
2015-08-07 19:47:10,102 INFO     pytan.handler.QuestionPoller: ID 1307: Progress Changed 100% (2 of
2015-08-07 19:47:10,102 INFO     pytan.handler.QuestionPoller: ID 1307: Reached Threshold of 99% (2

print the export_str returned from export_obj():
Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2015-08-07 19:46:14,930 DEBUG    pytan.handler.QuestionPoller: ID 1306: id resolved to 1306
2015-08-07 19:46:14,930 DEBUG    pytan.handler.QuestionPoller: ID 1306: expiration resolved to 2015-
2015-08-07 19:46:14,930 DEBUG    pytan.handler.QuestionPoller: ID 1306: query_text resolved to Get C
2015-08-07 19:46:14,930 DEBUG    pytan.handler.QuestionPoller: ID 1306: id resolved to 1306
2015-08-07 19:46:14,930 DEBUG    pytan.handler.QuestionPoller: ID 1306: Object Info resolved to Ques
2015-08-07 19:46:14,933 DEBUG    pytan.handler.QuestionPoller: ID 1306: Progress: Tested: 0, Passed:
2015-08-07 19:46:14,933 DEBUG    pytan.handler.QuestionPoller: ID 1306: Timing: Started: 2015-08-07
2015-08-07 19:46:14,933 INFO     pytan.handler.QuestionPoller: ID 1306: Progress Changed 0% (0 of 2)
2015-08-07 19:46:19,940 DEBUG    pytan.handler.QuestionPoller: ID 1306: Progress: Tested: 0, Passed:
2015-08-07 19:46:19,940 DEBUG    pytan.handler.QuestionPoller: ID 1306: Timing: Started: 2015-08-07
2015-08-07 19:46:24,944 DEBUG    pytan.handler.QuestionPoller: ID 1306: Progress: Tested: 0, Passed:
2015-08-07 19:46:24,944 DEBUG    pytan.handler.QuestionPoller: ID 1306: Timing: Started: 2015-08-07
2015-08-07 19:46:29,951 DEBUG    pytan.handler.QuestionPoller: ID 1306: Progress: Tested: 0, Passed:
2015-08-07 19:46:29,951 DEBUG    pytan.handler.QuestionPoller: ID 1306: Timing: Started: 2015-08-07
..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
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)

```

```
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

```
Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2015-08-07 19:47:10,227 DEBUG    pytan.handler.QuestionPoller: ID 1309: id resolved to 1309
2015-08-07 19:47:10,227 DEBUG    pytan.handler.QuestionPoller: ID 1309: expiration resolved to 2015-
2015-08-07 19:47:10,227 DEBUG    pytan.handler.QuestionPoller: ID 1309: query_text resolved to Get C
2015-08-07 19:47:10,227 DEBUG    pytan.handler.QuestionPoller: ID 1309: id resolved to 1309
2015-08-07 19:47:10,227 DEBUG    pytan.handler.QuestionPoller: ID 1309: Object Info resolved to Ques
2015-08-07 19:47:10,231 DEBUG    pytan.handler.QuestionPoller: ID 1309: Progress: Tested: 0, Passed:
2015-08-07 19:47:10,231 DEBUG    pytan.handler.QuestionPoller: ID 1309: Timing: Started: 2015-08-07
2015-08-07 19:47:10,231 INFO     pytan.handler.QuestionPoller: ID 1309: Progress Changed 0% (0 of 2)
2015-08-07 19:47:15,235 DEBUG    pytan.handler.QuestionPoller: ID 1309: Progress: Tested: 0, Passed:
2015-08-07 19:47:15,235 DEBUG    pytan.handler.QuestionPoller: ID 1309: Timing: Started: 2015-08-07
2015-08-07 19:47:20,239 DEBUG    pytan.handler.QuestionPoller: ID 1309: Progress: Tested: 1, Passed:
2015-08-07 19:47:20,239 DEBUG    pytan.handler.QuestionPoller: ID 1309: Timing: Started: 2015-08-07
2015-08-07 19:47:20,239 INFO     pytan.handler.QuestionPoller: ID 1309: Progress Changed 50% (1 of 2
2015-08-07 19:47:25,243 DEBUG    pytan.handler.QuestionPoller: ID 1309: Progress: Tested: 1, Passed:
2015-08-07 19:47:25,243 DEBUG    pytan.handler.QuestionPoller: ID 1309: Timing: Started: 2015-08-07
2015-08-07 19:47:30,250 DEBUG    pytan.handler.QuestionPoller: ID 1309: Progress: Tested: 1, Passed:
2015-08-07 19:47:30,250 DEBUG    pytan.handler.QuestionPoller: ID 1309: Timing: Started: 2015-08-07
2015-08-07 19:47:35,255 DEBUG    pytan.handler.QuestionPoller: ID 1309: Progress: Tested: 1, Passed:
2015-08-07 19:47:35,255 DEBUG    pytan.handler.QuestionPoller: ID 1309: Timing: Started: 2015-08-07
2015-08-07 19:47:40,259 DEBUG    pytan.handler.QuestionPoller: ID 1309: Progress: Tested: 2, Passed:
2015-08-07 19:47:40,259 DEBUG    pytan.handler.QuestionPoller: ID 1309: Timing: Started: 2015-08-07
2015-08-07 19:47:40,259 INFO     pytan.handler.QuestionPoller: ID 1309: Progress Changed 100% (2 of
2015-08-07 19:47:40,259 INFO     pytan.handler.QuestionPoller: ID 1309: Reached Threshold of 99% (2

print the export_str returned from export_obj():
Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2015-08-07 19:46:50,080 DEBUG    pytan.handler.QuestionPoller: ID 1307: id resolved to 1307
2015-08-07 19:46:50,080 DEBUG    pytan.handler.QuestionPoller: ID 1307: expiration resolved to 2015-
2015-08-07 19:46:50,080 DEBUG    pytan.handler.QuestionPoller: ID 1307: query_text resolved to Get C
2015-08-07 19:46:50,080 DEBUG    pytan.handler.QuestionPoller: ID 1307: id resolved to 1307
2015-08-07 19:46:50,080 DEBUG    pytan.handler.QuestionPoller: ID 1307: Object Info resolved to Ques
2015-08-07 19:46:50,083 DEBUG    pytan.handler.QuestionPoller: ID 1307: Progress: Tested: 0, Passed:
2015-08-07 19:46:50,083 DEBUG    pytan.handler.QuestionPoller: ID 1307: Timing: Started: 2015-08-07
2015-08-07 19:46:50,083 INFO     pytan.handler.QuestionPoller: ID 1307: Progress Changed 0% (0 of 2)
2015-08-07 19:46:55,088 DEBUG    pytan.handler.QuestionPoller: ID 1307: Progress: Tested: 1, Passed:
2015-08-07 19:46:55,088 DEBUG    pytan.handler.QuestionPoller: ID 1307: Timing: Started: 2015-08-07
2015-08-07 19:46:55,088 INFO     pytan.handler.QuestionPoller: ID 1307: Progress Changed 50% (1 of 2
2015-08-07 19:47:00,092 DEBUG    pytan.handler.QuestionPoller: ID 1307: Progress: Tested: 1, Passed:
2015-08-07 19:47:00,092 DEBUG    pytan.handler.QuestionPoller: ID 1307: Timing: Started: 2015-08-07
2015-08-07 19:47:05,096 DEBUG    pytan.handler.QuestionPoller: ID 1307: Progress: Tested: 1, Passed:
..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

```
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["header_sort"] = True
export_kwargs["export_format"] = u'csv'
export_kwargs["header_add_type"] = True
export_kwargs["expand_grouped_columns"] = True
export_kwargs["header_add_sensor"] = True

# ask the question that will provide the resultset that we want to use
ask_kwargs = {
    'qtype': 'manual',
    'sensors': [
        "Computer Name", "IP Route Details", "IP Address",
        'Folder Name Search with RegEx Match{dirname=Program Files,regex=.*Shared.*}',
    ],
}
response = handler.ask(**ask_kwargs)

# export the object to a string
# (we could just as easily export to a file using export_to_report_file)
```

```
export_kwargs['obj'] = response['question_results']
export_str = handler.export_obj(**export_kwargs)


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

print out
```

Output from Python Code

```
Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2015-08-07 19:47:40,405 DEBUG    pytan.handler.QuestionPoller: ID 1310: id resolved to 1310
2015-08-07 19:47:40,405 DEBUG    pytan.handler.QuestionPoller: ID 1310: expiration resolved to 2015-
2015-08-07 19:47:40,405 DEBUG    pytan.handler.QuestionPoller: ID 1310: query_text resolved to Get C
2015-08-07 19:47:40,405 DEBUG    pytan.handler.QuestionPoller: ID 1310: id resolved to 1310
2015-08-07 19:47:40,405 DEBUG    pytan.handler.QuestionPoller: ID 1310: Object Info resolved to Ques
2015-08-07 19:47:40,408 DEBUG    pytan.handler.QuestionPoller: ID 1310: Progress: Tested: 0, Passed:
2015-08-07 19:47:40,408 DEBUG    pytan.handler.QuestionPoller: ID 1310: Timing: Started: 2015-08-07
2015-08-07 19:47:40,408 INFO     pytan.handler.QuestionPoller: ID 1310: Progress Changed 0% (0 of 2)
2015-08-07 19:47:45,417 DEBUG    pytan.handler.QuestionPoller: ID 1310: Progress: Tested: 0, Passed:
2015-08-07 19:47:45,417 DEBUG    pytan.handler.QuestionPoller: ID 1310: Timing: Started: 2015-08-07
2015-08-07 19:47:50,421 DEBUG    pytan.handler.QuestionPoller: ID 1310: Progress: Tested: 0, Passed:
2015-08-07 19:47:50,421 DEBUG    pytan.handler.QuestionPoller: ID 1310: Timing: Started: 2015-08-07
2015-08-07 19:47:55,425 DEBUG    pytan.handler.QuestionPoller: ID 1310: Progress: Tested: 0, Passed:
2015-08-07 19:47:55,425 DEBUG    pytan.handler.QuestionPoller: ID 1310: Timing: Started: 2015-08-07
2015-08-07 19:48:00,431 DEBUG    pytan.handler.QuestionPoller: ID 1310: Progress: Tested: 0, Passed:
2015-08-07 19:48:00,431 DEBUG    pytan.handler.QuestionPoller: ID 1310: Timing: Started: 2015-08-07
2015-08-07 19:48:05,435 DEBUG    pytan.handler.QuestionPoller: ID 1310: Progress: Tested: 0, Passed:
2015-08-07 19:48:05,435 DEBUG    pytan.handler.QuestionPoller: ID 1310: Timing: Started: 2015-08-07
2015-08-07 19:48:10,440 DEBUG    pytan.handler.QuestionPoller: ID 1310: Progress: Tested: 0, Passed:
2015-08-07 19:48:10,440 DEBUG    pytan.handler.QuestionPoller: ID 1310: Timing: Started: 2015-08-07
2015-08-07 19:48:15,444 DEBUG    pytan.handler.QuestionPoller: ID 1310: Progress: Tested: 0, Passed:
2015-08-07 19:48:15,444 DEBUG    pytan.handler.QuestionPoller: ID 1310: Timing: Started: 2015-08-07
2015-08-07 19:48:20,449 DEBUG    pytan.handler.QuestionPoller: ID 1310: Progress: Tested: 0, Passed:
2015-08-07 19:48:20,449 DEBUG    pytan.handler.QuestionPoller: ID 1310: Timing: Started: 2015-08-07
2015-08-07 19:48:25,453 DEBUG    pytan.handler.QuestionPoller: ID 1310: Progress: Tested: 0, Passed:
2015-08-07 19:48:25,453 DEBUG    pytan.handler.QuestionPoller: ID 1310: Timing: Started: 2015-08-07
2015-08-07 19:48:30,459 DEBUG    pytan.handler.QuestionPoller: ID 1310: Progress: Tested: 0, Passed:
2015-08-07 19:48:30,459 DEBUG    pytan.handler.QuestionPoller: ID 1310: Timing: Started: 2015-08-07
2015-08-07 19:48:35,467 DEBUG    pytan.handler.QuestionPoller: ID 1310: Progress: Tested: 0, Passed:
2015-08-07 19:48:35,467 DEBUG    pytan.handler.QuestionPoller: ID 1310: Timing: Started: 2015-08-07
2015-08-07 19:48:40,473 DEBUG    pytan.handler.QuestionPoller: ID 1310: Progress: Tested: 0, Passed:
2015-08-07 19:48:40,473 DEBUG    pytan.handler.QuestionPoller: ID 1310: Timing: Started: 2015-08-07
2015-08-07 19:48:45,481 DEBUG    pytan.handler.QuestionPoller: ID 1310: Progress: Tested: 0, Passed:
2015-08-07 19:48:45,481 DEBUG    pytan.handler.QuestionPoller: ID 1310: Timing: Started: 2015-08-07
2015-08-07 19:48:50,489 DEBUG    pytan.handler.QuestionPoller: ID 1310: Progress: Tested: 0, Passed:
2015-08-07 19:48:50,489 DEBUG    pytan.handler.QuestionPoller: ID 1310: Timing: Started: 2015-08-07
2015-08-07 19:48:55,493 DEBUG    pytan.handler.QuestionPoller: ID 1310: Progress: Tested: 1, Passed:
2015-08-07 19:48:55,493 DEBUG    pytan.handler.QuestionPoller: ID 1310: Timing: Started: 2015-08-07
2015-08-07 19:48:55,493 INFO     pytan.handler.QuestionPoller: ID 1310: Progress Changed 50% (1 of 2
2015-08-07 19:49:00,497 DEBUG    pytan.handler.QuestionPoller: ID 1310: Progress: Tested: 1, Passed:
2015-08-07 19:49:00,497 DEBUG    pytan.handler.QuestionPoller: ID 1310: Timing: Started: 2015-08-07
```

```
2015-08-07 19:49:05,502 DEBUG    pytan.handler.QuestionPoller: ID 1310: Progress: Tested: 1, Passed:
2015-08-07 19:49:05,502 DEBUG    pytan.handler.QuestionPoller: ID 1310: Timing: Started: 2015-08-07
2015-08-07 19:49:10,508 DEBUG    pytan.handler.QuestionPoller: ID 1310: Progress: Tested: 2, Passed:
2015-08-07 19:49:10,508 DEBUG    pytan.handler.QuestionPoller: ID 1310: Timing: Started: 2015-08-07
2015-08-07 19:49:10,508 INFO     pytan.handler.QuestionPoller: ID 1310: Progress Changed 100% (2 of
2015-08-07 19:49:10,508 INFO     pytan.handler.QuestionPoller: ID 1310: Reached Threshold of 99% (2

print the export_str returned from export_obj():
Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2015-08-07 19:47:10,227 DEBUG    pytan.handler.QuestionPoller: ID 1309: id resolved to 1309
2015-08-07 19:47:10,227 DEBUG    pytan.handler.QuestionPoller: ID 1309: expiration resolved to 2015-
2015-08-07 19:47:10,227 DEBUG    pytan.handler.QuestionPoller: ID 1309: query_text resolved to Get C
2015-08-07 19:47:10,227 DEBUG    pytan.handler.QuestionPoller: ID 1309: id resolved to 1309
2015-08-07 19:47:10,227 DEBUG    pytan.handler.QuestionPoller: ID 1309: Object Info resolved to Ques
2015-08-07 19:47:10,231 DEBUG    pytan.handler.QuestionPoller: ID 1309: Progress: Tested: 0, Passed:
2015-08-07 19:47:10,231 DEBUG    pytan.handler.QuestionPoller: ID 1309: Timing: Started: 2015-08-07
2015-08-07 19:47:10,231 INFO     pytan.handler.QuestionPoller: ID 1309: Progress Changed 0% (0 of 2)
2015-08-07 19:47:15,235 DEBUG    pytan.handler.QuestionPoller: ID 1309: Progress: Tested: 0, Passed:
2015-08-07 19:47:15,235 DEBUG    pytan.handler.QuestionPoller: ID 1309: Timing: Started: 2015-08-07
2015-08-07 19:47:20,239 DEBUG    pytan.handler.QuestionPoller: ID 1309: Progress: Tested: 1, Passed:
2015-08-07 19:47:20,239 DEBUG    pytan.handler.QuestionPoller: ID 1309: Timing: Started: 2015-08-07
2015-08-07 19:47:20,239 INFO     pytan.handler.QuestionPoller: ID 1309: Progress Changed 50% (1 of 2
2015-08-07 19:47:25,243 DEBUG    pytan.handler.QuestionPoller: ID 1309: Progress: Tested: 1, Passed:
..trimmed for brevity..
```

Export resultset json

Export a ResultSet from asking a question as JSON with the default 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"
```

```
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:
```

```
2015-08-07 19:49:10,713 DEBUG    pytan.handler.QuestionPoller: ID 1311: Timing: Started: 2015-08-07
2015-08-07 19:49:10,713 INFO     pytan.handler.QuestionPoller: ID 1311: Progress Changed 0% (0 of 2)
2015-08-07 19:49:15,721 DEBUG    pytan.handler.QuestionPoller: ID 1311: Progress: Tested: 0, Passed:
2015-08-07 19:49:15,721 DEBUG    pytan.handler.QuestionPoller: ID 1311: Timing: Started: 2015-08-07
2015-08-07 19:49:20,725 DEBUG    pytan.handler.QuestionPoller: ID 1311: Progress: Tested: 0, Passed:
2015-08-07 19:49:20,726 DEBUG    pytan.handler.QuestionPoller: ID 1311: Timing: Started: 2015-08-07
2015-08-07 19:49:25,730 DEBUG    pytan.handler.QuestionPoller: ID 1311: Progress: Tested: 0, Passed:
2015-08-07 19:49:25,730 DEBUG    pytan.handler.QuestionPoller: ID 1311: Timing: Started: 2015-08-07
2015-08-07 19:49:30,739 DEBUG    pytan.handler.QuestionPoller: ID 1311: Progress: Tested: 1, Passed:
2015-08-07 19:49:30,739 DEBUG    pytan.handler.QuestionPoller: ID 1311: Timing: Started: 2015-08-07
2015-08-07 19:49:30,739 INFO     pytan.handler.QuestionPoller: ID 1311: Progress Changed 50% (1 of 2
2015-08-07 19:49:35,743 DEBUG    pytan.handler.QuestionPoller: ID 1311: Progress: Tested: 1, Passed:
2015-08-07 19:49:35,744 DEBUG    pytan.handler.QuestionPoller: ID 1311: Timing: Started: 2015-08-07
2015-08-07 19:49:40,751 DEBUG    pytan.handler.QuestionPoller: ID 1311: Progress: Tested: 2, Passed:
2015-08-07 19:49:40,751 DEBUG    pytan.handler.QuestionPoller: ID 1311: Timing: Started: 2015-08-07
2015-08-07 19:49:40,751 INFO     pytan.handler.QuestionPoller: ID 1311: Progress Changed 100% (2 of
2015-08-07 19:49:40,751 INFO     pytan.handler.QuestionPoller: ID 1311: Reached Threshold of 99% (2

print the export_str returned from export_obj():
Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2015-08-07 19:47:40,405 DEBUG    pytan.handler.QuestionPoller: ID 1310: id resolved to 1310
2015-08-07 19:47:40,405 DEBUG    pytan.handler.QuestionPoller: ID 1310: expiration resolved to 2015-
2015-08-07 19:47:40,405 DEBUG    pytan.handler.QuestionPoller: ID 1310: query_text resolved to Get C
2015-08-07 19:47:40,405 DEBUG    pytan.handler.QuestionPoller: ID 1310: id resolved to 1310
2015-08-07 19:47:40,405 DEBUG    pytan.handler.QuestionPoller: ID 1310: Object Info resolved to Ques
2015-08-07 19:47:40,408 DEBUG    pytan.handler.QuestionPoller: ID 1310: Progress: Tested: 0, Passed:
2015-08-07 19:47:40,408 DEBUG    pytan.handler.QuestionPoller: ID 1310: Timing: Started: 2015-08-07
2015-08-07 19:47:40,408 INFO     pytan.handler.QuestionPoller: ID 1310: Progress Changed 0% (0 of 2)
2015-08-07 19:47:45,417 DEBUG    pytan.handler.QuestionPoller: ID 1310: Progress: Tested: 0, Passed:
2015-08-07 19:47:45,417 DEBUG    pytan.handler.QuestionPoller: ID 1310: Timing: Started: 2015-08-07
2015-08-07 19:47:50,421 DEBUG    pytan.handler.QuestionPoller: ID 1310: Progress: Tested: 0, Passed:
2015-08-07 19:47:50,421 DEBUG    pytan.handler.QuestionPoller: ID 1310: Timing: Started: 2015-08-07
2015-08-07 19:47:55,425 DEBUG    pytan.handler.QuestionPoller: ID 1310: Progress: Tested: 0, Passed:
2015-08-07 19:47:55,425 DEBUG    pytan.handler.QuestionPoller: ID 1310: Timing: Started: 2015-08-07
..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)

# 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]
```

```
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
```

```
print out
```

Output from Python Code

```
Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2015-08-07 19:49:40,850 DEBUG    pytan.handler.QuestionPoller: ID 1312: id resolved to 1312
2015-08-07 19:49:40,850 DEBUG    pytan.handler.QuestionPoller: ID 1312: expiration resolved to 2015-
2015-08-07 19:49:40,850 DEBUG    pytan.handler.QuestionPoller: ID 1312: query_text resolved to Get C
2015-08-07 19:49:40,850 DEBUG    pytan.handler.QuestionPoller: ID 1312: id resolved to 1312
2015-08-07 19:49:40,850 DEBUG    pytan.handler.QuestionPoller: ID 1312: Object Info resolved to Ques
2015-08-07 19:49:40,853 DEBUG    pytan.handler.QuestionPoller: ID 1312: Progress: Tested: 0, Passed:
2015-08-07 19:49:40,853 DEBUG    pytan.handler.QuestionPoller: ID 1312: Timing: Started: 2015-08-07
2015-08-07 19:49:40,854 INFO     pytan.handler.QuestionPoller: ID 1312: Progress Changed 0% (0 of 2)
2015-08-07 19:49:45,859 DEBUG    pytan.handler.QuestionPoller: ID 1312: Progress: Tested: 0, Passed:
2015-08-07 19:49:45,859 DEBUG    pytan.handler.QuestionPoller: ID 1312: Timing: Started: 2015-08-07
2015-08-07 19:49:50,863 DEBUG    pytan.handler.QuestionPoller: ID 1312: Progress: Tested: 0, Passed:
2015-08-07 19:49:50,863 DEBUG    pytan.handler.QuestionPoller: ID 1312: Timing: Started: 2015-08-07
2015-08-07 19:49:55,870 DEBUG    pytan.handler.QuestionPoller: ID 1312: Progress: Tested: 0, Passed:
2015-08-07 19:49:55,870 DEBUG    pytan.handler.QuestionPoller: ID 1312: Timing: Started: 2015-08-07
2015-08-07 19:50:00,877 DEBUG    pytan.handler.QuestionPoller: ID 1312: Progress: Tested: 0, Passed:
2015-08-07 19:50:00,877 DEBUG    pytan.handler.QuestionPoller: ID 1312: Timing: Started: 2015-08-07
2015-08-07 19:50:05,881 DEBUG    pytan.handler.QuestionPoller: ID 1312: Progress: Tested: 0, Passed:
2015-08-07 19:50:05,881 DEBUG    pytan.handler.QuestionPoller: ID 1312: Timing: Started: 2015-08-07
2015-08-07 19:50:10,886 DEBUG    pytan.handler.QuestionPoller: ID 1312: Progress: Tested: 0, Passed:
2015-08-07 19:50:10,886 DEBUG    pytan.handler.QuestionPoller: ID 1312: Timing: Started: 2015-08-07
2015-08-07 19:50:15,891 DEBUG    pytan.handler.QuestionPoller: ID 1312: Progress: Tested: 0, Passed:
2015-08-07 19:50:15,891 DEBUG    pytan.handler.QuestionPoller: ID 1312: Timing: Started: 2015-08-07
2015-08-07 19:50:20,896 DEBUG    pytan.handler.QuestionPoller: ID 1312: Progress: Tested: 0, Passed:
2015-08-07 19:50:20,896 DEBUG    pytan.handler.QuestionPoller: ID 1312: Timing: Started: 2015-08-07
2015-08-07 19:50:25,901 DEBUG    pytan.handler.QuestionPoller: ID 1312: Progress: Tested: 0, Passed:
2015-08-07 19:50:25,901 DEBUG    pytan.handler.QuestionPoller: ID 1312: Timing: Started: 2015-08-07
2015-08-07 19:50:30,906 DEBUG    pytan.handler.QuestionPoller: ID 1312: Progress: Tested: 0, Passed:
2015-08-07 19:50:30,906 DEBUG    pytan.handler.QuestionPoller: ID 1312: Timing: Started: 2015-08-07
2015-08-07 19:50:35,910 DEBUG    pytan.handler.QuestionPoller: ID 1312: Progress: Tested: 0, Passed:
2015-08-07 19:50:35,910 DEBUG    pytan.handler.QuestionPoller: ID 1312: Timing: Started: 2015-08-07
2015-08-07 19:50:40,915 DEBUG    pytan.handler.QuestionPoller: ID 1312: Progress: Tested: 0, Passed:
2015-08-07 19:50:40,915 DEBUG    pytan.handler.QuestionPoller: ID 1312: Timing: Started: 2015-08-07
2015-08-07 19:50:45,919 DEBUG    pytan.handler.QuestionPoller: ID 1312: Progress: Tested: 0, Passed:
2015-08-07 19:50:45,919 DEBUG    pytan.handler.QuestionPoller: ID 1312: Timing: Started: 2015-08-07
2015-08-07 19:50:50,923 DEBUG    pytan.handler.QuestionPoller: ID 1312: Progress: Tested: 0, Passed:
2015-08-07 19:50:50,923 DEBUG    pytan.handler.QuestionPoller: ID 1312: Timing: Started: 2015-08-07
2015-08-07 19:50:55,928 DEBUG    pytan.handler.QuestionPoller: ID 1312: Progress: Tested: 0, Passed:
2015-08-07 19:50:55,928 DEBUG    pytan.handler.QuestionPoller: ID 1312: Timing: Started: 2015-08-07
2015-08-07 19:51:00,934 DEBUG    pytan.handler.QuestionPoller: ID 1312: Progress: Tested: 0, Passed:
2015-08-07 19:51:00,935 DEBUG    pytan.handler.QuestionPoller: ID 1312: Timing: Started: 2015-08-07
2015-08-07 19:51:05,939 DEBUG    pytan.handler.QuestionPoller: ID 1312: Progress: Tested: 0, Passed:
2015-08-07 19:51:05,939 DEBUG    pytan.handler.QuestionPoller: ID 1312: Timing: Started: 2015-08-07
2015-08-07 19:51:10,947 DEBUG    pytan.handler.QuestionPoller: ID 1312: Progress: Tested: 2, Passed:
2015-08-07 19:51:10,947 DEBUG    pytan.handler.QuestionPoller: ID 1312: Timing: Started: 2015-08-07
2015-08-07 19:51:10,947 INFO     pytan.handler.QuestionPoller: ID 1312: Progress Changed 100% (2 of
2015-08-07 19:51:10,947 INFO     pytan.handler.QuestionPoller: ID 1312: Reached Threshold of 99% (2

print the export_str returned from export_obj():
Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2015-08-07 19:49:10,709 DEBUG    pytan.handler.QuestionPoller: ID 1311: id resolved to 1311
2015-08-07 19:49:10,709 DEBUG    pytan.handler.QuestionPoller: ID 1311: expiration resolved to 2015-
2015-08-07 19:49:10,709 DEBUG    pytan.handler.QuestionPoller: ID 1311: query_text resolved to Get C
```

```
2015-08-07 19:49:10,709 DEBUG    pytan.handler.QuestionPoller: ID 1311: id resolved to 1311
2015-08-07 19:49:10,709 DEBUG    pytan.handler.QuestionPoller: ID 1311: Object Info resolved to Ques
2015-08-07 19:49:10,713 DEBUG    pytan.handler.QuestionPoller: ID 1311: Progress: Tested: 0, Passed:
2015-08-07 19:49:10,713 DEBUG    pytan.handler.QuestionPoller: ID 1311: Timing: Started: 2015-08-07
2015-08-07 19:49:10,713 INFO     pytan.handler.QuestionPoller: ID 1311: Progress Changed 0% (0 of 2)
2015-08-07 19:49:15,721 DEBUG    pytan.handler.QuestionPoller: ID 1311: Progress: Tested: 0, Passed:
2015-08-07 19:49:15,721 DEBUG    pytan.handler.QuestionPoller: ID 1311: Timing: Started: 2015-08-07
2015-08-07 19:49:20,725 DEBUG    pytan.handler.QuestionPoller: ID 1311: Progress: Tested: 0, Passed:
2015-08-07 19:49:20,726 DEBUG    pytan.handler.QuestionPoller: ID 1311: Timing: Started: 2015-08-07
2015-08-07 19:49:25,730 DEBUG    pytan.handler.QuestionPoller: ID 1311: Progress: Tested: 0, Passed:
2015-08-07 19:49:25,730 DEBUG    pytan.handler.QuestionPoller: ID 1311: Timing: Started: 2015-08-07
..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
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"] = True

# ask the question that will provide the resultset that we want to use
ask_kwargs = {
    'qtype': 'manual',
    'sensors': [
        "Computer Name", "IP Route Details", "IP Address",
        'Folder Name Search with RegEx Match{dirname=Program Files,regex=.*Shared.*}',
    ],
}
response = handler.ask(**ask_kwargs)

# export the object to a string
# (we could just as easily export to a file using export_to_report_file)
export_kwargs['obj'] = response['question_results']
export_str = handler.export_obj(**export_kwargs)


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

print out
```

Output from Python Code

```
Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2015-08-07 19:51:11,061 DEBUG    pytan.handler.QuestionPoller: ID 1313: id resolved to 1313
2015-08-07 19:51:11,061 DEBUG    pytan.handler.QuestionPoller: ID 1313: expiration resolved to 2015-
2015-08-07 19:51:11,061 DEBUG    pytan.handler.QuestionPoller: ID 1313: query_text resolved to Get C
2015-08-07 19:51:11,061 DEBUG    pytan.handler.QuestionPoller: ID 1313: id resolved to 1313
2015-08-07 19:51:11,061 DEBUG    pytan.handler.QuestionPoller: ID 1313: Object Info resolved to Ques
2015-08-07 19:51:11,066 DEBUG    pytan.handler.QuestionPoller: ID 1313: Progress: Tested: 0, Passed:
2015-08-07 19:51:11,066 DEBUG    pytan.handler.QuestionPoller: ID 1313: Timing: Started: 2015-08-07
2015-08-07 19:51:11,066 INFO     pytan.handler.QuestionPoller: ID 1313: Progress Changed 0% (0 of 2)
2015-08-07 19:51:16,074 DEBUG    pytan.handler.QuestionPoller: ID 1313: Progress: Tested: 0, Passed:
2015-08-07 19:51:16,074 DEBUG    pytan.handler.QuestionPoller: ID 1313: Timing: Started: 2015-08-07
2015-08-07 19:51:21,079 DEBUG    pytan.handler.QuestionPoller: ID 1313: Progress: Tested: 0, Passed:
2015-08-07 19:51:21,079 DEBUG    pytan.handler.QuestionPoller: ID 1313: Timing: Started: 2015-08-07
2015-08-07 19:51:26,083 DEBUG    pytan.handler.QuestionPoller: ID 1313: Progress: Tested: 0, Passed:
2015-08-07 19:51:26,083 DEBUG    pytan.handler.QuestionPoller: ID 1313: Timing: Started: 2015-08-07
2015-08-07 19:51:31,089 DEBUG    pytan.handler.QuestionPoller: ID 1313: Progress: Tested: 0, Passed:
2015-08-07 19:51:31,089 DEBUG    pytan.handler.QuestionPoller: ID 1313: Timing: Started: 2015-08-07
2015-08-07 19:51:36,093 DEBUG    pytan.handler.QuestionPoller: ID 1313: Progress: Tested: 0, Passed:
```

```
2015-08-07 19:51:36,093 DEBUG    pytan.handler.QuestionPoller: ID 1313: Timing: Started: 2015-08-07
2015-08-07 19:51:41,099 DEBUG    pytan.handler.QuestionPoller: ID 1313: Progress: Tested: 0, Passed:
2015-08-07 19:51:41,099 DEBUG    pytan.handler.QuestionPoller: ID 1313: Timing: Started: 2015-08-07
2015-08-07 19:51:46,107 DEBUG    pytan.handler.QuestionPoller: ID 1313: Progress: Tested: 0, Passed:
2015-08-07 19:51:46,107 DEBUG    pytan.handler.QuestionPoller: ID 1313: Timing: Started: 2015-08-07
2015-08-07 19:51:51,112 DEBUG    pytan.handler.QuestionPoller: ID 1313: Progress: Tested: 0, Passed:
2015-08-07 19:51:51,112 DEBUG    pytan.handler.QuestionPoller: ID 1313: Timing: Started: 2015-08-07
2015-08-07 19:51:56,117 DEBUG    pytan.handler.QuestionPoller: ID 1313: Progress: Tested: 0, Passed:
2015-08-07 19:51:56,117 DEBUG    pytan.handler.QuestionPoller: ID 1313: Timing: Started: 2015-08-07
2015-08-07 19:52:01,121 DEBUG    pytan.handler.QuestionPoller: ID 1313: Progress: Tested: 0, Passed:
2015-08-07 19:52:01,122 DEBUG    pytan.handler.QuestionPoller: ID 1313: Timing: Started: 2015-08-07
2015-08-07 19:52:06,125 DEBUG    pytan.handler.QuestionPoller: ID 1313: Progress: Tested: 0, Passed:
2015-08-07 19:52:06,126 DEBUG    pytan.handler.QuestionPoller: ID 1313: Timing: Started: 2015-08-07
2015-08-07 19:52:11,132 DEBUG    pytan.handler.QuestionPoller: ID 1313: Progress: Tested: 0, Passed:
2015-08-07 19:52:11,133 DEBUG    pytan.handler.QuestionPoller: ID 1313: Timing: Started: 2015-08-07
2015-08-07 19:52:16,136 DEBUG    pytan.handler.QuestionPoller: ID 1313: Progress: Tested: 0, Passed:
2015-08-07 19:52:16,136 DEBUG    pytan.handler.QuestionPoller: ID 1313: Timing: Started: 2015-08-07
2015-08-07 19:52:21,142 DEBUG    pytan.handler.QuestionPoller: ID 1313: Progress: Tested: 2, Passed:
2015-08-07 19:52:21,142 DEBUG    pytan.handler.QuestionPoller: ID 1313: Timing: Started: 2015-08-07
2015-08-07 19:52:21,142 INFO     pytan.handler.QuestionPoller: ID 1313: Progress Changed 100% (2 of
2015-08-07 19:52:21,142 INFO     pytan.handler.QuestionPoller: ID 1313: Reached Threshold of 99% (2

print the export_str returned from export_obj():
Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2015-08-07 19:49:40,850 DEBUG    pytan.handler.QuestionPoller: ID 1312: id resolved to 1312
2015-08-07 19:49:40,850 DEBUG    pytan.handler.QuestionPoller: ID 1312: expiration resolved to 2015-
2015-08-07 19:49:40,850 DEBUG    pytan.handler.QuestionPoller: ID 1312: query_text resolved to Get C
2015-08-07 19:49:40,850 DEBUG    pytan.handler.QuestionPoller: ID 1312: id resolved to 1312
2015-08-07 19:49:40,850 DEBUG    pytan.handler.QuestionPoller: ID 1312: Object Info resolved to Ques
2015-08-07 19:49:40,853 DEBUG    pytan.handler.QuestionPoller: ID 1312: Progress: Tested: 0, Passed:
2015-08-07 19:49:40,853 DEBUG    pytan.handler.QuestionPoller: ID 1312: Timing: Started: 2015-08-07
2015-08-07 19:49:40,854 INFO     pytan.handler.QuestionPoller: ID 1312: Progress Changed 0% (0 of 2)
2015-08-07 19:49:45,859 DEBUG    pytan.handler.QuestionPoller: ID 1312: Progress: Tested: 0, Passed:
2015-08-07 19:49:45,859 DEBUG    pytan.handler.QuestionPoller: ID 1312: Timing: Started: 2015-08-07
2015-08-07 19:49:50,863 DEBUG    pytan.handler.QuestionPoller: ID 1312: Progress: Tested: 0, Passed:
2015-08-07 19:49:50,863 DEBUG    pytan.handler.QuestionPoller: ID 1312: Timing: Started: 2015-08-07
2015-08-07 19:49:55,870 DEBUG    pytan.handler.QuestionPoller: ID 1312: Progress: Tested: 0, Passed:
2015-08-07 19:49:55,870 DEBUG    pytan.handler.QuestionPoller: ID 1312: Timing: Started: 2015-08-07
..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

```
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"] = False

# ask the question that will provide the resultset that we want to use
ask_kwargs = {
    'qtype': 'manual',
    'sensors': [
        "Computer Name", "IP Route Details", "IP Address",
        'Folder Name Search with RegEx Match{dirname=Program Files,regex=.*Shared.*}',
    ],
}
response = handler.ask(**ask_kwargs)

# export the object to a string
# (we could just as easily export to a file using export_to_report_file)
export_kwargs['obj'] = response['question_results']
export_str = handler.export_obj(**export_kwargs)


print ""
print "print the export_str returned from export_obj():"
if len(out.splitlines()) > 15:
```

```
    out = out.splitlines()[0:15]
    out.append('..trimmed for brevity..')
    out = '\n'.join(out)

print out
```

Output from Python Code

```
Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2015-08-07 19:52:21,243 DEBUG    pytan.handler.QuestionPoller: ID 1315: id resolved to 1315
2015-08-07 19:52:21,243 DEBUG    pytan.handler.QuestionPoller: ID 1315: expiration resolved to 2015-
2015-08-07 19:52:21,243 DEBUG    pytan.handler.QuestionPoller: ID 1315: query_text resolved to Get C
2015-08-07 19:52:21,243 DEBUG    pytan.handler.QuestionPoller: ID 1315: id resolved to 1315
2015-08-07 19:52:21,243 DEBUG    pytan.handler.QuestionPoller: ID 1315: Object Info resolved to Ques
2015-08-07 19:52:21,246 DEBUG    pytan.handler.QuestionPoller: ID 1315: Progress: Tested: 0, Passed:
2015-08-07 19:52:21,246 DEBUG    pytan.handler.QuestionPoller: ID 1315: Timing: Started: 2015-08-07
2015-08-07 19:52:21,246 INFO     pytan.handler.QuestionPoller: ID 1315: Progress Changed 0% (0 of 2)
2015-08-07 19:52:26,251 DEBUG    pytan.handler.QuestionPoller: ID 1315: Progress: Tested: 0, Passed:
2015-08-07 19:52:26,251 DEBUG    pytan.handler.QuestionPoller: ID 1315: Timing: Started: 2015-08-07
2015-08-07 19:52:31,255 DEBUG    pytan.handler.QuestionPoller: ID 1315: Progress: Tested: 0, Passed:
2015-08-07 19:52:31,255 DEBUG    pytan.handler.QuestionPoller: ID 1315: Timing: Started: 2015-08-07
2015-08-07 19:52:36,264 DEBUG    pytan.handler.QuestionPoller: ID 1315: Progress: Tested: 1, Passed:
2015-08-07 19:52:36,264 DEBUG    pytan.handler.QuestionPoller: ID 1315: Timing: Started: 2015-08-07
2015-08-07 19:52:36,264 INFO     pytan.handler.QuestionPoller: ID 1315: Progress Changed 50% (1 of 2
2015-08-07 19:52:41,271 DEBUG    pytan.handler.QuestionPoller: ID 1315: Progress: Tested: 1, Passed:
2015-08-07 19:52:41,271 DEBUG    pytan.handler.QuestionPoller: ID 1315: Timing: Started: 2015-08-07
2015-08-07 19:52:46,275 DEBUG    pytan.handler.QuestionPoller: ID 1315: Progress: Tested: 1, Passed:
2015-08-07 19:52:46,275 DEBUG    pytan.handler.QuestionPoller: ID 1315: Timing: Started: 2015-08-07
2015-08-07 19:52:51,282 DEBUG    pytan.handler.QuestionPoller: ID 1315: Progress: Tested: 2, Passed:
2015-08-07 19:52:51,283 DEBUG    pytan.handler.QuestionPoller: ID 1315: Timing: Started: 2015-08-07
2015-08-07 19:52:51,283 INFO     pytan.handler.QuestionPoller: ID 1315: Progress Changed 100% (2 of
2015-08-07 19:52:51,283 INFO     pytan.handler.QuestionPoller: ID 1315: Reached Threshold of 99% (2

print the export_str returned from export_obj():
Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2015-08-07 19:51:11,061 DEBUG    pytan.handler.QuestionPoller: ID 1313: id resolved to 1313
2015-08-07 19:51:11,061 DEBUG    pytan.handler.QuestionPoller: ID 1313: expiration resolved to 2015-
2015-08-07 19:51:11,061 DEBUG    pytan.handler.QuestionPoller: ID 1313: query_text resolved to Get C
2015-08-07 19:51:11,061 DEBUG    pytan.handler.QuestionPoller: ID 1313: id resolved to 1313
2015-08-07 19:51:11,061 DEBUG    pytan.handler.QuestionPoller: ID 1313: Object Info resolved to Ques
2015-08-07 19:51:11,066 DEBUG    pytan.handler.QuestionPoller: ID 1313: Progress: Tested: 0, Passed:
2015-08-07 19:51:11,066 DEBUG    pytan.handler.QuestionPoller: ID 1313: Timing: Started: 2015-08-07
2015-08-07 19:51:11,066 INFO     pytan.handler.QuestionPoller: ID 1313: Progress Changed 0% (0 of 2)
2015-08-07 19:51:16,074 DEBUG    pytan.handler.QuestionPoller: ID 1313: Progress: Tested: 0, Passed:
2015-08-07 19:51:16,074 DEBUG    pytan.handler.QuestionPoller: ID 1313: Timing: Started: 2015-08-07
2015-08-07 19:51:21,079 DEBUG    pytan.handler.QuestionPoller: ID 1313: Progress: Tested: 0, Passed:
2015-08-07 19:51:21,079 DEBUG    pytan.handler.QuestionPoller: ID 1313: Timing: Started: 2015-08-07
2015-08-07 19:51:26,083 DEBUG    pytan.handler.QuestionPoller: ID 1313: Progress: Tested: 0, Passed:
2015-08-07 19:51:26,083 DEBUG    pytan.handler.QuestionPoller: ID 1313: Timing: Started: 2015-08-07
..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

```
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': [
        "Computer Name", "IP Route Details", "IP Address",
        'Folder Name Search with RegEx Match{dirname=Program Files,regex=.*Shared.*}',
    ],
}
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

```
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_type"] = False

# ask the question that will provide the resultset that we want to use
ask_kwargs = {
    'qtype': 'manual',
    'sensors': [
```

```
        "Computer Name", "IP Route Details", "IP Address",
        'Folder Name Search with RegEx Match{dirname=Program Files,regex=.*Shared.*}',
    ],
}
response = handler.ask(**ask_kwargs)

# export the object to a string
# (we could just as easily export to a file using export_to_report_file)
export_kwargs['obj'] = response['question_results']
export_str = handler.export_obj(**export_kwargs)


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

print out
```

Output from Python Code

```
Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2015-08-07 19:53:11,519 DEBUG    pytan.handler.QuestionPoller: ID 1318: id resolved to 1318
2015-08-07 19:53:11,519 DEBUG    pytan.handler.QuestionPoller: ID 1318: expiration resolved to 2015-
2015-08-07 19:53:11,519 DEBUG    pytan.handler.QuestionPoller: ID 1318: query_text resolved to Get C
2015-08-07 19:53:11,519 DEBUG    pytan.handler.QuestionPoller: ID 1318: id resolved to 1318
2015-08-07 19:53:11,519 DEBUG    pytan.handler.QuestionPoller: ID 1318: Object Info resolved to Ques
2015-08-07 19:53:11,522 DEBUG    pytan.handler.QuestionPoller: ID 1318: Progress: Tested: 0, Passed:
2015-08-07 19:53:11,522 DEBUG    pytan.handler.QuestionPoller: ID 1318: Timing: Started: 2015-08-07
2015-08-07 19:53:11,522 INFO     pytan.handler.QuestionPoller: ID 1318: Progress Changed 0% (0 of 2)
2015-08-07 19:53:16,530 DEBUG    pytan.handler.QuestionPoller: ID 1318: Progress: Tested: 0, Passed:
2015-08-07 19:53:16,530 DEBUG    pytan.handler.QuestionPoller: ID 1318: Timing: Started: 2015-08-07
2015-08-07 19:53:21,538 DEBUG    pytan.handler.QuestionPoller: ID 1318: Progress: Tested: 0, Passed:
2015-08-07 19:53:21,538 DEBUG    pytan.handler.QuestionPoller: ID 1318: Timing: Started: 2015-08-07
2015-08-07 19:53:26,543 DEBUG    pytan.handler.QuestionPoller: ID 1318: Progress: Tested: 0, Passed:
2015-08-07 19:53:26,543 DEBUG    pytan.handler.QuestionPoller: ID 1318: Timing: Started: 2015-08-07
2015-08-07 19:53:31,548 DEBUG    pytan.handler.QuestionPoller: ID 1318: Progress: Tested: 0, Passed:
2015-08-07 19:53:31,548 DEBUG    pytan.handler.QuestionPoller: ID 1318: Timing: Started: 2015-08-07
2015-08-07 19:53:36,552 DEBUG    pytan.handler.QuestionPoller: ID 1318: Progress: Tested: 0, Passed:
2015-08-07 19:53:36,552 DEBUG    pytan.handler.QuestionPoller: ID 1318: Timing: Started: 2015-08-07
2015-08-07 19:53:41,559 DEBUG    pytan.handler.QuestionPoller: ID 1318: Progress: Tested: 0, Passed:
2015-08-07 19:53:41,559 DEBUG    pytan.handler.QuestionPoller: ID 1318: Timing: Started: 2015-08-07
2015-08-07 19:53:46,566 DEBUG    pytan.handler.QuestionPoller: ID 1318: Progress: Tested: 0, Passed:
2015-08-07 19:53:46,566 DEBUG    pytan.handler.QuestionPoller: ID 1318: Timing: Started: 2015-08-07
2015-08-07 19:53:51,570 DEBUG    pytan.handler.QuestionPoller: ID 1318: Progress: Tested: 0, Passed:
2015-08-07 19:53:51,570 DEBUG    pytan.handler.QuestionPoller: ID 1318: Timing: Started: 2015-08-07
2015-08-07 19:53:56,577 DEBUG    pytan.handler.QuestionPoller: ID 1318: Progress: Tested: 0, Passed:
2015-08-07 19:53:56,577 DEBUG    pytan.handler.QuestionPoller: ID 1318: Timing: Started: 2015-08-07
2015-08-07 19:54:01,583 DEBUG    pytan.handler.QuestionPoller: ID 1318: Progress: Tested: 0, Passed:
2015-08-07 19:54:01,583 DEBUG    pytan.handler.QuestionPoller: ID 1318: Timing: Started: 2015-08-07
2015-08-07 19:54:06,590 DEBUG    pytan.handler.QuestionPoller: ID 1318: Progress: Tested: 0, Passed:
2015-08-07 19:54:06,591 DEBUG    pytan.handler.QuestionPoller: ID 1318: Timing: Started: 2015-08-07
2015-08-07 19:54:11,600 DEBUG    pytan.handler.QuestionPoller: ID 1318: Progress: Tested: 0, Passed:
2015-08-07 19:54:11,600 DEBUG    pytan.handler.QuestionPoller: ID 1318: Timing: Started: 2015-08-07
2015-08-07 19:54:16,608 DEBUG    pytan.handler.QuestionPoller: ID 1318: Progress: Tested: 0, Passed:
```

```
2015-08-07 19:54:16,608 DEBUG    pytan.handler.QuestionPoller: ID 1318: Timing: Started: 2015-08-07
2015-08-07 19:54:21,612 DEBUG    pytan.handler.QuestionPoller: ID 1318: Progress: Tested: 0, Passed:
2015-08-07 19:54:21,612 DEBUG    pytan.handler.QuestionPoller: ID 1318: Timing: Started: 2015-08-07
2015-08-07 19:54:26,621 DEBUG    pytan.handler.QuestionPoller: ID 1318: Progress: Tested: 0, Passed:
2015-08-07 19:54:26,621 DEBUG    pytan.handler.QuestionPoller: ID 1318: Timing: Started: 2015-08-07
2015-08-07 19:54:31,629 DEBUG    pytan.handler.QuestionPoller: ID 1318: Progress: Tested: 0, Passed:
2015-08-07 19:54:31,629 DEBUG    pytan.handler.QuestionPoller: ID 1318: Timing: Started: 2015-08-07
2015-08-07 19:54:36,635 DEBUG    pytan.handler.QuestionPoller: ID 1318: Progress: Tested: 2, Passed:
2015-08-07 19:54:36,635 DEBUG    pytan.handler.QuestionPoller: ID 1318: Timing: Started: 2015-08-07
2015-08-07 19:54:36,635 INFO     pytan.handler.QuestionPoller: ID 1318: Progress Changed 100% (2 of
2015-08-07 19:54:36,635 INFO     pytan.handler.QuestionPoller: ID 1318: Reached Threshold of 99% (2

print the export_str returned from export_obj():
Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2015-08-07 19:52:51,388 DEBUG    pytan.handler.QuestionPoller: ID 1316: id resolved to 1316
2015-08-07 19:52:51,388 DEBUG    pytan.handler.QuestionPoller: ID 1316: expiration resolved to 2015-
2015-08-07 19:52:51,388 DEBUG    pytan.handler.QuestionPoller: ID 1316: query_text resolved to Get C
2015-08-07 19:52:51,388 DEBUG    pytan.handler.QuestionPoller: ID 1316: id resolved to 1316
2015-08-07 19:52:51,388 DEBUG    pytan.handler.QuestionPoller: ID 1316: Object Info resolved to Ques
2015-08-07 19:52:51,392 DEBUG    pytan.handler.QuestionPoller: ID 1316: Progress: Tested: 0, Passed:
2015-08-07 19:52:51,392 DEBUG    pytan.handler.QuestionPoller: ID 1316: Timing: Started: 2015-08-07
2015-08-07 19:52:51,392 INFO     pytan.handler.QuestionPoller: ID 1316: Progress Changed 0% (0 of 2)
2015-08-07 19:52:56,396 DEBUG    pytan.handler.QuestionPoller: ID 1316: Progress: Tested: 0, Passed:
2015-08-07 19:52:56,396 DEBUG    pytan.handler.QuestionPoller: ID 1316: Timing: Started: 2015-08-07
2015-08-07 19:53:01,404 DEBUG    pytan.handler.QuestionPoller: ID 1316: Progress: Tested: 0, Passed:
2015-08-07 19:53:01,404 DEBUG    pytan.handler.QuestionPoller: ID 1316: Timing: Started: 2015-08-07
2015-08-07 19:53:06,413 DEBUG    pytan.handler.QuestionPoller: ID 1316: Progress: Tested: 1, Passed:
2015-08-07 19:53:06,413 DEBUG    pytan.handler.QuestionPoller: ID 1316: Timing: Started: 2015-08-07
..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)
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_type"] = True

# ask the question that will provide the resultset that we want to use
ask_kwargs = {
    'qtype': 'manual',
    'sensors': [
        "Computer Name", "IP Route Details", "IP Address",
        'Folder Name Search with RegEx Match{dirname=Program Files,regex=.*Shared.*}',
    ],
}
response = handler.ask(**ask_kwargs)

# export the object to a string
# (we could just as easily export to a file using export_to_report_file)
export_kwargs['obj'] = response['question_results']
export_str = handler.export_obj(**export_kwargs)


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

print out
```

Output from Python Code

```
Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2015-08-07 19:54:36,751 DEBUG    pytan.handler.QuestionPoller: ID 1319: id resolved to 1319
2015-08-07 19:54:36,751 DEBUG    pytan.handler.QuestionPoller: ID 1319: expiration resolved to 2015-
2015-08-07 19:54:36,751 DEBUG    pytan.handler.QuestionPoller: ID 1319: query_text resolved to Get C
2015-08-07 19:54:36,751 DEBUG    pytan.handler.QuestionPoller: ID 1319: id resolved to 1319
2015-08-07 19:54:36,751 DEBUG    pytan.handler.QuestionPoller: ID 1319: Object Info resolved to Ques
2015-08-07 19:54:36,754 DEBUG    pytan.handler.QuestionPoller: ID 1319: Progress: Tested: 0, Passed:
2015-08-07 19:54:36,754 DEBUG    pytan.handler.QuestionPoller: ID 1319: Timing: Started: 2015-08-07
2015-08-07 19:54:36,754 INFO     pytan.handler.QuestionPoller: ID 1319: Progress Changed 0% (0 of 2)
2015-08-07 19:54:41,761 DEBUG    pytan.handler.QuestionPoller: ID 1319: Progress: Tested: 0, Passed:
2015-08-07 19:54:41,761 DEBUG    pytan.handler.QuestionPoller: ID 1319: Timing: Started: 2015-08-07
2015-08-07 19:54:46,767 DEBUG    pytan.handler.QuestionPoller: ID 1319: Progress: Tested: 0, Passed:
2015-08-07 19:54:46,767 DEBUG    pytan.handler.QuestionPoller: ID 1319: Timing: Started: 2015-08-07
2015-08-07 19:54:51,771 DEBUG    pytan.handler.QuestionPoller: ID 1319: Progress: Tested: 0, Passed:
2015-08-07 19:54:51,771 DEBUG    pytan.handler.QuestionPoller: ID 1319: Timing: Started: 2015-08-07
2015-08-07 19:54:56,775 DEBUG    pytan.handler.QuestionPoller: ID 1319: Progress: Tested: 2, Passed:
2015-08-07 19:54:56,775 DEBUG    pytan.handler.QuestionPoller: ID 1319: Timing: Started: 2015-08-07
2015-08-07 19:54:56,775 INFO     pytan.handler.QuestionPoller: ID 1319: Progress Changed 100% (2 of
2015-08-07 19:54:56,775 INFO     pytan.handler.QuestionPoller: ID 1319: Reached Threshold of 99% (2

print the export_str returned from export_obj():
Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2015-08-07 19:53:11,519 DEBUG    pytan.handler.QuestionPoller: ID 1318: id resolved to 1318
2015-08-07 19:53:11,519 DEBUG    pytan.handler.QuestionPoller: ID 1318: expiration resolved to 2015-
2015-08-07 19:53:11,519 DEBUG    pytan.handler.QuestionPoller: ID 1318: query_text resolved to Get C
2015-08-07 19:53:11,519 DEBUG    pytan.handler.QuestionPoller: ID 1318: id resolved to 1318
2015-08-07 19:53:11,519 DEBUG    pytan.handler.QuestionPoller: ID 1318: Object Info resolved to Ques
2015-08-07 19:53:11,522 DEBUG    pytan.handler.QuestionPoller: ID 1318: Progress: Tested: 0, Passed:
2015-08-07 19:53:11,522 DEBUG    pytan.handler.QuestionPoller: ID 1318: Timing: Started: 2015-08-07
2015-08-07 19:53:11,522 INFO     pytan.handler.QuestionPoller: ID 1318: Progress Changed 0% (0 of 2)
2015-08-07 19:53:16,530 DEBUG    pytan.handler.QuestionPoller: ID 1318: Progress: Tested: 0, Passed:
2015-08-07 19:53:16,530 DEBUG    pytan.handler.QuestionPoller: ID 1318: Timing: Started: 2015-08-07
2015-08-07 19:53:21,538 DEBUG    pytan.handler.QuestionPoller: ID 1318: Progress: Tested: 0, Passed:
2015-08-07 19:53:21,538 DEBUG    pytan.handler.QuestionPoller: ID 1318: Timing: Started: 2015-08-07
2015-08-07 19:53:26,543 DEBUG    pytan.handler.QuestionPoller: ID 1318: Progress: Tested: 0, Passed:
2015-08-07 19:53:26,543 DEBUG    pytan.handler.QuestionPoller: ID 1318: Timing: Started: 2015-08-07
..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

```
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"] = False

# ask the question that will provide the resultset that we want to use
ask_kwargs = {
    'qtype': 'manual',
    'sensors': [
        "Computer Name", "IP Route Details", "IP Address",
        'Folder Name Search with RegEx Match{dirname=Program Files,regex=.*Shared.*}',
    ],
}
response = handler.ask(**ask_kwargs)

# export the object to a string
# (we could just as easily export to a file using export_to_report_file)
export_kwargs['obj'] = response['question_results']
export_str = handler.export_obj(**export_kwargs)


print ""
print "print the export_str returned from export_obj():"
if len(out.splitlines()) > 15:
    out = out.splitlines()[0:15]
    out.append('..trimmed for brevity..')
```

```
    out = '\n'.join(out)

print out
```

Output from Python Code

```
Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2015-08-07 19:54:56,919 DEBUG    pytan.handler.QuestionPoller: ID 1320: id resolved to 1320
2015-08-07 19:54:56,919 DEBUG    pytan.handler.QuestionPoller: ID 1320: expiration resolved to 2015-
2015-08-07 19:54:56,919 DEBUG    pytan.handler.QuestionPoller: ID 1320: query_text resolved to Get C
2015-08-07 19:54:56,919 DEBUG    pytan.handler.QuestionPoller: ID 1320: id resolved to 1320
2015-08-07 19:54:56,919 DEBUG    pytan.handler.QuestionPoller: ID 1320: Object Info resolved to Ques
2015-08-07 19:54:56,923 DEBUG    pytan.handler.QuestionPoller: ID 1320: Progress: Tested: 0, Passed:
2015-08-07 19:54:56,923 DEBUG    pytan.handler.QuestionPoller: ID 1320: Timing: Started: 2015-08-07
2015-08-07 19:54:56,923 INFO     pytan.handler.QuestionPoller: ID 1320: Progress Changed 0% (0 of 2)
2015-08-07 19:55:01,930 DEBUG    pytan.handler.QuestionPoller: ID 1320: Progress: Tested: 0, Passed:
2015-08-07 19:55:01,930 DEBUG    pytan.handler.QuestionPoller: ID 1320: Timing: Started: 2015-08-07
2015-08-07 19:55:06,936 DEBUG    pytan.handler.QuestionPoller: ID 1320: Progress: Tested: 0, Passed:
2015-08-07 19:55:06,936 DEBUG    pytan.handler.QuestionPoller: ID 1320: Timing: Started: 2015-08-07
2015-08-07 19:55:11,944 DEBUG    pytan.handler.QuestionPoller: ID 1320: Progress: Tested: 0, Passed:
2015-08-07 19:55:11,944 DEBUG    pytan.handler.QuestionPoller: ID 1320: Timing: Started: 2015-08-07
2015-08-07 19:55:16,949 DEBUG    pytan.handler.QuestionPoller: ID 1320: Progress: Tested: 0, Passed:
2015-08-07 19:55:16,950 DEBUG    pytan.handler.QuestionPoller: ID 1320: Timing: Started: 2015-08-07
2015-08-07 19:55:21,957 DEBUG    pytan.handler.QuestionPoller: ID 1320: Progress: Tested: 1, Passed:
2015-08-07 19:55:21,957 DEBUG    pytan.handler.QuestionPoller: ID 1320: Timing: Started: 2015-08-07
2015-08-07 19:55:21,957 INFO     pytan.handler.QuestionPoller: ID 1320: Progress Changed 50% (1 of 2
2015-08-07 19:55:26,962 DEBUG    pytan.handler.QuestionPoller: ID 1320: Progress: Tested: 2, Passed:
2015-08-07 19:55:26,962 DEBUG    pytan.handler.QuestionPoller: ID 1320: Timing: Started: 2015-08-07
2015-08-07 19:55:26,962 INFO     pytan.handler.QuestionPoller: ID 1320: Progress Changed 100% (2 of
2015-08-07 19:55:26,962 INFO     pytan.handler.QuestionPoller: ID 1320: Reached Threshold of 99% (2

print the export_str returned from export_obj():
Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2015-08-07 19:54:36,751 DEBUG    pytan.handler.QuestionPoller: ID 1319: id resolved to 1319
2015-08-07 19:54:36,751 DEBUG    pytan.handler.QuestionPoller: ID 1319: expiration resolved to 2015-
2015-08-07 19:54:36,751 DEBUG    pytan.handler.QuestionPoller: ID 1319: query_text resolved to Get C
2015-08-07 19:54:36,751 DEBUG    pytan.handler.QuestionPoller: ID 1319: id resolved to 1319
2015-08-07 19:54:36,751 DEBUG    pytan.handler.QuestionPoller: ID 1319: Object Info resolved to Ques
2015-08-07 19:54:36,754 DEBUG    pytan.handler.QuestionPoller: ID 1319: Progress: Tested: 0, Passed:
2015-08-07 19:54:36,754 DEBUG    pytan.handler.QuestionPoller: ID 1319: Timing: Started: 2015-08-07
2015-08-07 19:54:36,754 INFO     pytan.handler.QuestionPoller: ID 1319: Progress Changed 0% (0 of 2)
2015-08-07 19:54:41,761 DEBUG    pytan.handler.QuestionPoller: ID 1319: Progress: Tested: 0, Passed:
2015-08-07 19:54:41,761 DEBUG    pytan.handler.QuestionPoller: ID 1319: Timing: Started: 2015-08-07
2015-08-07 19:54:46,767 DEBUG    pytan.handler.QuestionPoller: ID 1319: Progress: Tested: 0, Passed:
2015-08-07 19:54:46,767 DEBUG    pytan.handler.QuestionPoller: ID 1319: Timing: Started: 2015-08-07
2015-08-07 19:54:51,771 DEBUG    pytan.handler.QuestionPoller: ID 1319: Progress: Tested: 0, Passed:
2015-08-07 19:54:51,771 DEBUG    pytan.handler.QuestionPoller: ID 1319: Timing: Started: 2015-08-07
..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

```
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': [
        "Computer Name", "IP Route Details", "IP Address",
        'Folder Name Search with RegEx Match{dirname=Program Files,regex=.*Shared.*}',
    ],
}
response = handler.ask(**ask_kwargs)

```

```
# export the object to a string
# (we could just as easily export to a file using export_to_report_file)
export_kwargs['obj'] = response['question_results']
export_str = handler.export_obj(**export_kwargs)


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

print out
```

Output from Python Code

```
Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2015-08-07 19:55:27,070 DEBUG    pytan.handler.QuestionPoller: ID 1321: id resolved to 1321
2015-08-07 19:55:27,070 DEBUG    pytan.handler.QuestionPoller: ID 1321: expiration resolved to 2015-
2015-08-07 19:55:27,070 DEBUG    pytan.handler.QuestionPoller: ID 1321: query_text resolved to Get C
2015-08-07 19:55:27,070 DEBUG    pytan.handler.QuestionPoller: ID 1321: id resolved to 1321
2015-08-07 19:55:27,070 DEBUG    pytan.handler.QuestionPoller: ID 1321: Object Info resolved to Ques
2015-08-07 19:55:27,074 DEBUG    pytan.handler.QuestionPoller: ID 1321: Progress: Tested: 0, Passed:
2015-08-07 19:55:27,074 DEBUG    pytan.handler.QuestionPoller: ID 1321: Timing: Started: 2015-08-07
2015-08-07 19:55:27,074 INFO     pytan.handler.QuestionPoller: ID 1321: Progress Changed 0% (0 of 2)
2015-08-07 19:55:32,077 DEBUG    pytan.handler.QuestionPoller: ID 1321: Progress: Tested: 0, Passed:
2015-08-07 19:55:32,077 DEBUG    pytan.handler.QuestionPoller: ID 1321: Timing: Started: 2015-08-07
2015-08-07 19:55:37,083 DEBUG    pytan.handler.QuestionPoller: ID 1321: Progress: Tested: 0, Passed:
2015-08-07 19:55:37,083 DEBUG    pytan.handler.QuestionPoller: ID 1321: Timing: Started: 2015-08-07
2015-08-07 19:55:42,089 DEBUG    pytan.handler.QuestionPoller: ID 1321: Progress: Tested: 0, Passed:
2015-08-07 19:55:42,089 DEBUG    pytan.handler.QuestionPoller: ID 1321: Timing: Started: 2015-08-07
2015-08-07 19:55:47,097 DEBUG    pytan.handler.QuestionPoller: ID 1321: Progress: Tested: 0, Passed:
2015-08-07 19:55:47,097 DEBUG    pytan.handler.QuestionPoller: ID 1321: Timing: Started: 2015-08-07
2015-08-07 19:55:52,105 DEBUG    pytan.handler.QuestionPoller: ID 1321: Progress: Tested: 0, Passed:
2015-08-07 19:55:52,105 DEBUG    pytan.handler.QuestionPoller: ID 1321: Timing: Started: 2015-08-07
2015-08-07 19:55:57,112 DEBUG    pytan.handler.QuestionPoller: ID 1321: Progress: Tested: 0, Passed:
2015-08-07 19:55:57,112 DEBUG    pytan.handler.QuestionPoller: ID 1321: Timing: Started: 2015-08-07
2015-08-07 19:56:02,119 DEBUG    pytan.handler.QuestionPoller: ID 1321: Progress: Tested: 0, Passed:
2015-08-07 19:56:02,119 DEBUG    pytan.handler.QuestionPoller: ID 1321: Timing: Started: 2015-08-07
2015-08-07 19:56:07,129 DEBUG    pytan.handler.QuestionPoller: ID 1321: Progress: Tested: 0, Passed:
2015-08-07 19:56:07,129 DEBUG    pytan.handler.QuestionPoller: ID 1321: Timing: Started: 2015-08-07
2015-08-07 19:56:12,138 DEBUG    pytan.handler.QuestionPoller: ID 1321: Progress: Tested: 0, Passed:
2015-08-07 19:56:12,138 DEBUG    pytan.handler.QuestionPoller: ID 1321: Timing: Started: 2015-08-07
2015-08-07 19:56:17,145 DEBUG    pytan.handler.QuestionPoller: ID 1321: Progress: Tested: 0, Passed:
2015-08-07 19:56:17,145 DEBUG    pytan.handler.QuestionPoller: ID 1321: Timing: Started: 2015-08-07
2015-08-07 19:56:22,152 DEBUG    pytan.handler.QuestionPoller: ID 1321: Progress: Tested: 0, Passed:
2015-08-07 19:56:22,152 DEBUG    pytan.handler.QuestionPoller: ID 1321: Timing: Started: 2015-08-07
2015-08-07 19:56:27,160 DEBUG    pytan.handler.QuestionPoller: ID 1321: Progress: Tested: 0, Passed:
2015-08-07 19:56:27,160 DEBUG    pytan.handler.QuestionPoller: ID 1321: Timing: Started: 2015-08-07
2015-08-07 19:56:32,167 DEBUG    pytan.handler.QuestionPoller: ID 1321: Progress: Tested: 2, Passed:
2015-08-07 19:56:32,167 DEBUG    pytan.handler.QuestionPoller: ID 1321: Timing: Started: 2015-08-07
2015-08-07 19:56:32,167 INFO     pytan.handler.QuestionPoller: ID 1321: Progress Changed 100% (2 of
2015-08-07 19:56:32,167 INFO     pytan.handler.QuestionPoller: ID 1321: Reached Threshold of 99% (2

print the export_str returned from export_obj():
Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
```

```
2015-08-07 19:54:56,919 DEBUG    pytan.handler.QuestionPoller: ID 1320: id resolved to 1320
2015-08-07 19:54:56,919 DEBUG    pytan.handler.QuestionPoller: ID 1320: expiration resolved to 2015-
2015-08-07 19:54:56,919 DEBUG    pytan.handler.QuestionPoller: ID 1320: query_text resolved to Get C
2015-08-07 19:54:56,919 DEBUG    pytan.handler.QuestionPoller: ID 1320: id resolved to 1320
2015-08-07 19:54:56,919 DEBUG    pytan.handler.QuestionPoller: ID 1320: Object Info resolved to Ques
2015-08-07 19:54:56,923 DEBUG    pytan.handler.QuestionPoller: ID 1320: Progress: Tested: 0, Passed:
2015-08-07 19:54:56,923 DEBUG    pytan.handler.QuestionPoller: ID 1320: Timing: Started: 2015-08-07
2015-08-07 19:54:56,923 INFO     pytan.handler.QuestionPoller: ID 1320: Progress Changed 0% (0 of 2)
2015-08-07 19:55:01,930 DEBUG    pytan.handler.QuestionPoller: ID 1320: Progress: Tested: 0, Passed:
2015-08-07 19:55:01,930 DEBUG    pytan.handler.QuestionPoller: ID 1320: Timing: Started: 2015-08-07
2015-08-07 19:55:06,936 DEBUG    pytan.handler.QuestionPoller: ID 1320: Progress: Tested: 0, Passed:
2015-08-07 19:55:06,936 DEBUG    pytan.handler.QuestionPoller: ID 1320: Timing: Started: 2015-08-07
2015-08-07 19:55:11,944 DEBUG    pytan.handler.QuestionPoller: ID 1320: Progress: Tested: 0, Passed:
2015-08-07 19:55:11,944 DEBUG    pytan.handler.QuestionPoller: ID 1320: Timing: Started: 2015-08-07
..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
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"] = [[]]

# ask the question that will provide the resultset that we want to use
ask_kwargs = {
    'qtype': 'manual',
    'sensors': [
        "Computer Name"
    ],
}
response = handler.ask(**ask_kwargs)
export_kwargs['obj'] = response['question_results']

# export the object to a string
# this should throw an exception: pytan.exceptions.HandlerError
import traceback

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

Output from Python Code

```
Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2015-08-07 19:56:32,279 DEBUG    pytan.handler.QuestionPoller: ID 1323: id resolved to 1323
2015-08-07 19:56:32,279 DEBUG    pytan.handler.QuestionPoller: ID 1323: expiration resolved to 2015-
2015-08-07 19:56:32,279 DEBUG    pytan.handler.QuestionPoller: ID 1323: query_text resolved to Get C
2015-08-07 19:56:32,279 DEBUG    pytan.handler.QuestionPoller: ID 1323: id resolved to 1323
2015-08-07 19:56:32,279 DEBUG    pytan.handler.QuestionPoller: ID 1323: Object Info resolved to Ques
2015-08-07 19:56:32,282 DEBUG    pytan.handler.QuestionPoller: ID 1323: Progress: Tested: 0, Passed:
2015-08-07 19:56:32,282 DEBUG    pytan.handler.QuestionPoller: ID 1323: Timing: Started: 2015-08-07
2015-08-07 19:56:32,282 INFO     pytan.handler.QuestionPoller: ID 1323: Progress Changed 0% (0 of 2)
2015-08-07 19:56:37,290 DEBUG    pytan.handler.QuestionPoller: ID 1323: Progress: Tested: 2, Passed:
2015-08-07 19:56:37,290 DEBUG    pytan.handler.QuestionPoller: ID 1323: Timing: Started: 2015-08-07
2015-08-07 19:56:37,290 INFO     pytan.handler.QuestionPoller: ID 1323: Progress Changed 100% (2 of
2015-08-07 19:56:37,290 INFO     pytan.handler.QuestionPoller: ID 1323: Reached Threshold of 99% (2
Traceback (most recent call last):
  File "<string>", line 65, in <module>
  File "/Users/jolsen/gh/pytan/lib/pytan/utils.py", line 2699, in wrap
    ret = f(*args, **kwargs)
  File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 1084, in export_obj
    pytan.utils.check_dictkey(**check_args)
```

```
  File "/Users/jolsen/gh/pytan/lib/pytan/utils.py", line 2692, in check_dictkey
    raise pytan.exceptions.HandlerError(err(key, valid_list_types, list_types))
HandlerError: 'header_sort' must be a list of [<type 'str'>, <type 'unicode'>], you supplied [<type
```

Invalid export resultset csv bad sort type

Export a ResultSet from asking a question using a bad header_sort

Example Python Code

```
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'bad'

# ask the question that will provide the resultset that we want to use
ask_kwargs = {
    'qtype': 'manual',
    'sensors': [
        "Computer Name"
    ],
}
response = handler.ask(**ask_kwargs)
export_kwargs['obj'] = response['question_results']

# export the object to a string
# this should throw an exception: pytan.exceptions.HandlerError
import traceback

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

Output from Python Code

```
Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2015-08-07 19:56:37,342 DEBUG    pytan.handler.QuestionPoller: ID 1324: id resolved to 1324
2015-08-07 19:56:37,342 DEBUG    pytan.handler.QuestionPoller: ID 1324: expiration resolved to 2015-
2015-08-07 19:56:37,342 DEBUG    pytan.handler.QuestionPoller: ID 1324: query_text resolved to Get C
2015-08-07 19:56:37,342 DEBUG    pytan.handler.QuestionPoller: ID 1324: id resolved to 1324
2015-08-07 19:56:37,342 DEBUG    pytan.handler.QuestionPoller: ID 1324: Object Info resolved to Ques
2015-08-07 19:56:37,345 DEBUG    pytan.handler.QuestionPoller: ID 1324: Progress: Tested: 0, Passed:
2015-08-07 19:56:37,345 DEBUG    pytan.handler.QuestionPoller: ID 1324: Timing: Started: 2015-08-07
2015-08-07 19:56:37,345 INFO     pytan.handler.QuestionPoller: ID 1324: Progress Changed 0% (0 of 2)
2015-08-07 19:56:42,353 DEBUG    pytan.handler.QuestionPoller: ID 1324: Progress: Tested: 0, Passed:
2015-08-07 19:56:42,353 DEBUG    pytan.handler.QuestionPoller: ID 1324: Timing: Started: 2015-08-07
2015-08-07 19:56:47,361 DEBUG    pytan.handler.QuestionPoller: ID 1324: Progress: Tested: 1, Passed:
2015-08-07 19:56:47,361 DEBUG    pytan.handler.QuestionPoller: ID 1324: Timing: Started: 2015-08-07
2015-08-07 19:56:47,361 INFO     pytan.handler.QuestionPoller: ID 1324: Progress Changed 50% (1 of 2
2015-08-07 19:56:52,368 DEBUG    pytan.handler.QuestionPoller: ID 1324: Progress: Tested: 2, Passed:
2015-08-07 19:56:52,368 DEBUG    pytan.handler.QuestionPoller: ID 1324: Timing: Started: 2015-08-07
2015-08-07 19:56:52,368 INFO     pytan.handler.QuestionPoller: ID 1324: Progress Changed 100% (2 of
2015-08-07 19:56:52,368 INFO     pytan.handler.QuestionPoller: ID 1324: Reached Threshold of 99% (2
Traceback (most recent call last):
  File "<string>", line 65, in <module>
  File "/Users/jolsen/gh/pytan/lib/pytan/utils.py", line 2699, in wrap
    ret = f(*args, **kwargs)
  File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 1084, in export_obj
    pytan.utils.check_dictkey(**check_args)
  File "/Users/jolsen/gh/pytan/lib/pytan/utils.py", line 2685, in check_dictkey
    raise pytan.exceptions.HandlerError(err(key, valid_types, k_type))
HandlerError: 'header_sort' must be one of [<type 'bool'>, <type 'list'>, <type 'tuple'>], you suppl
```

Invalid export resultset csv bad expand type

Export a ResultSet from asking a question using a bad expand_grouped_columns

Example Python Code

```
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["expand_grouped_columns"] = u'bad'

# ask the question that will provide the resultset that we want to use
ask_kwargs = {
    'qtype': 'manual',
    'sensors': [
        "Computer Name"
    ],
}
response = handler.ask(**ask_kwargs)
export_kwargs['obj'] = response['question_results']
```

```

# export the object to a string
# this should throw an exception: pytan.exceptions.HandlerError
import traceback

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

Output from Python Code

```
Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2015-08-07 19:56:52,416 DEBUG    pytan.handler.QuestionPoller: ID 1325: id resolved to 1325
2015-08-07 19:56:52,416 DEBUG    pytan.handler.QuestionPoller: ID 1325: expiration resolved to 2015-
2015-08-07 19:56:52,416 DEBUG    pytan.handler.QuestionPoller: ID 1325: query_text resolved to Get C
2015-08-07 19:56:52,416 DEBUG    pytan.handler.QuestionPoller: ID 1325: id resolved to 1325
2015-08-07 19:56:52,416 DEBUG    pytan.handler.QuestionPoller: ID 1325: Object Info resolved to Ques
2015-08-07 19:56:52,420 DEBUG    pytan.handler.QuestionPoller: ID 1325: Progress: Tested: 0, Passed:
2015-08-07 19:56:52,420 DEBUG    pytan.handler.QuestionPoller: ID 1325: Timing: Started: 2015-08-07
2015-08-07 19:56:52,420 INFO     pytan.handler.QuestionPoller: ID 1325: Progress Changed 0% (0 of 2)
2015-08-07 19:56:57,424 DEBUG    pytan.handler.QuestionPoller: ID 1325: Progress: Tested: 1, Passed:
2015-08-07 19:56:57,424 DEBUG    pytan.handler.QuestionPoller: ID 1325: Timing: Started: 2015-08-07
2015-08-07 19:56:57,424 INFO     pytan.handler.QuestionPoller: ID 1325: Progress Changed 50% (1 of 2
2015-08-07 19:57:02,429 DEBUG    pytan.handler.QuestionPoller: ID 1325: Progress: Tested: 2, Passed:
2015-08-07 19:57:02,429 DEBUG    pytan.handler.QuestionPoller: ID 1325: Timing: Started: 2015-08-07
2015-08-07 19:57:02,429 INFO     pytan.handler.QuestionPoller: ID 1325: Progress Changed 100% (2 of
2015-08-07 19:57:02,429 INFO     pytan.handler.QuestionPoller: ID 1325: Reached Threshold of 99% (2
Traceback (most recent call last):
  File "<string>", line 65, in <module>
  File "/Users/jolsen/gh/pytan/lib/pytan/utils.py", line 2699, in wrap
    ret = f(*args, **kwargs)
  File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 1084, in export_obj
    pytan.utils.check_dictkey(**check_args)
  File "/Users/jolsen/gh/pytan/lib/pytan/utils.py", line 2685, in check_dictkey
    raise pytan.exceptions.HandlerError(err(key, valid_types, k_type))
HandlerError: 'expand_grouped_columns' must be one of [<type 'bool'>], you supplied <type 'unicode'>
```

Invalid export resultset csv bad sensors sub type

Export a ResultSet from asking a question using a bad sensors

Example Python Code

```
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["sensors"] = [[]]
export_kwargs["header_add_sensor"] = True

# ask the question that will provide the resultset that we want to use
ask_kwargs = {
    'qtype': 'manual',
    'sensors': [
        "Computer Name"
    ],
}
response = handler.ask(**ask_kwargs)
export_kwargs['obj'] = response['question_results']

# export the object to a string
# this should throw an exception: pytan.exceptions.HandlerError
import traceback

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

Output from Python Code

```
Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2015-08-07 19:57:02,479 DEBUG    pytan.handler.QuestionPoller: ID 1326: id resolved to 1326
2015-08-07 19:57:02,479 DEBUG    pytan.handler.QuestionPoller: ID 1326: expiration resolved to 2015-
2015-08-07 19:57:02,479 DEBUG    pytan.handler.QuestionPoller: ID 1326: query_text resolved to Get C
2015-08-07 19:57:02,479 DEBUG    pytan.handler.QuestionPoller: ID 1326: id resolved to 1326
2015-08-07 19:57:02,479 DEBUG    pytan.handler.QuestionPoller: ID 1326: Object Info resolved to Ques
2015-08-07 19:57:02,482 DEBUG    pytan.handler.QuestionPoller: ID 1326: Progress: Tested: 0, Passed:
2015-08-07 19:57:02,483 DEBUG    pytan.handler.QuestionPoller: ID 1326: Timing: Started: 2015-08-07
2015-08-07 19:57:02,483 INFO     pytan.handler.QuestionPoller: ID 1326: Progress Changed 0% (0 of 2)
2015-08-07 19:57:07,486 DEBUG    pytan.handler.QuestionPoller: ID 1326: Progress: Tested: 0, Passed:
2015-08-07 19:57:07,486 DEBUG    pytan.handler.QuestionPoller: ID 1326: Timing: Started: 2015-08-07
2015-08-07 19:57:12,490 DEBUG    pytan.handler.QuestionPoller: ID 1326: Progress: Tested: 2, Passed:
2015-08-07 19:57:12,490 DEBUG    pytan.handler.QuestionPoller: ID 1326: Timing: Started: 2015-08-07
2015-08-07 19:57:12,490 INFO     pytan.handler.QuestionPoller: ID 1326: Progress Changed 100% (2 of
2015-08-07 19:57:12,490 INFO     pytan.handler.QuestionPoller: ID 1326: Reached Threshold of 99% (2
Traceback (most recent call last):
  File "<string>", line 66, in <module>
  File "/Users/jolsen/gh/pytan/lib/pytan/utils.py", line 2699, in wrap
    ret = f(*args, **kwargs)
  File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 1084, in export_obj
    pytan.utils.check_dictkey(**check_args)
  File "/Users/jolsen/gh/pytan/lib/pytan/utils.py", line 2692, in check_dictkey
    raise pytan.exceptions.HandlerError(err(key, valid_list_types, list_types))
HandlerError: 'sensors' must be a list of [<class 'taniumpy.object_types.sensor.Sensor'>], you suppl
```

Invalid export resultset bad format

Export a ResultSet from asking a question using a bad export_format

Example Python Code

```
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'bad'

# ask the question that will provide the resultset that we want to use
ask_kwargs = {
    'qtype': 'manual',
    'sensors': [
        "Computer Name"
    ],
}
response = handler.ask(**ask_kwargs)
export_kwargs['obj'] = response['question_results']

# export the object to a string
# this should throw an exception: pytan.exceptions.HandlerError
import traceback

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

Output from Python Code

```
Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2015-08-07 19:57:12,541 DEBUG    pytan.handler.QuestionPoller: ID 1327: id resolved to 1327
2015-08-07 19:57:12,541 DEBUG    pytan.handler.QuestionPoller: ID 1327: expiration resolved to 2015-
2015-08-07 19:57:12,541 DEBUG    pytan.handler.QuestionPoller: ID 1327: query_text resolved to Get C
2015-08-07 19:57:12,541 DEBUG    pytan.handler.QuestionPoller: ID 1327: id resolved to 1327
2015-08-07 19:57:12,541 DEBUG    pytan.handler.QuestionPoller: ID 1327: Object Info resolved to Ques
2015-08-07 19:57:12,545 DEBUG    pytan.handler.QuestionPoller: ID 1327: Progress: Tested: 0, Passed:
2015-08-07 19:57:12,545 DEBUG    pytan.handler.QuestionPoller: ID 1327: Timing: Started: 2015-08-07
2015-08-07 19:57:12,545 INFO     pytan.handler.QuestionPoller: ID 1327: Progress Changed 0% (0 of 2)
2015-08-07 19:57:17,551 DEBUG    pytan.handler.QuestionPoller: ID 1327: Progress: Tested: 0, Passed:
2015-08-07 19:57:17,551 DEBUG    pytan.handler.QuestionPoller: ID 1327: Timing: Started: 2015-08-07
2015-08-07 19:57:22,556 DEBUG    pytan.handler.QuestionPoller: ID 1327: Progress: Tested: 2, Passed:
```

```
2015-08-07 19:57:22,556 DEBUG    pytan.handler.QuestionPoller: ID 1327: Timing: Started: 2015-08-07
2015-08-07 19:57:22,556 INFO     pytan.handler.QuestionPoller: ID 1327: Progress Changed 100% (2 of
2015-08-07 19:57:22,556 INFO     pytan.handler.QuestionPoller: ID 1327: Reached Threshold of 99% (2
Traceback (most recent call last):
  File "<string>", line 64, in <module>
  File "/Users/jolsen/gh/pytan/lib/pytan/utils.py", line 2699, in wrap
    ret = f(*args, **kwargs)
  File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 1078, in export_obj
    raise pytan.exceptions.HandlerError(err)
HandlerError: u'bad' not a supported export format for ResultSet, must be one of: json, csv
```

PyTan API Valid Export BaseType Examples

Export basetype csv default options

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

Example Python Code

```python
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'

# get the objects that will provide the basetype that we want to use
get_kwargs = {
    'name': [
        "Computer Name", "IP Route Details", "IP Address",
        'Folder Name Search with RegEx Match',
    ],
    'objtype': 'sensor',
}
response = handler.get(**get_kwargs)

# export the object to a string
# (we could just as easily export to a file using export_to_report_file)
export_kwargs['obj'] = response
export_str = handler.export_obj(**export_kwargs)


print ""
print "print the export_str returned from export_obj():"

out = export_str
if len(out.splitlines()) > 15:
    out = out.splitlines()[0:15]
    out.append('..trimmed for brevity..')
    out = '\n'.join(out)

print out
```

Output from Python Code

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

print the export_str returned from export_obj():
category,creation_time,delimiter,description,exclude_from_parse_flag,hash,hidden_flag,id,ignore_case
Reserved,,,"The assigned name of the client machine.
Example: workstation-1.company.com",0,3409330187,0,3,1,,86400,,,,,Computer Name,,Windows,select CSNa
Network,2015-08-07T13:22:12,|,"Returns IPv4 network routes, filtered to exclude noise. With Flags, M
Example:  172.16.0.0|192.168.1.1|255.255.0.0|UG|100|eth0",1,435227963,0,552,1,Jim Olsen,60,0,defined
Set objWMIService = GetObject(&quot;winmgmts:&quot; _
    &amp; &quot;{impersonationLevel=impersonate}!\\&quot; &amp; strComputer &amp; &quot;\root\cimv2&

Set collip = objWMIService.ExecQuery(&quot;select * from win32_networkadapterconfiguration where IPE
dim ipaddrs()
ipcount = 0
for each ipItem in collip
    for each ipaddr in ipItem.IPAddress
        ipcount = ipcount + 1
```

```
    next
..trimmed for brevity..
```

Export basetype json type false

Export a BaseType from getting objects as JSON with false for include_type

Example Python Code

```
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["include_type"] = False

# get the objects that will provide the basetype that we want to use
get_kwargs = {
    'name': [
        "Computer Name", "IP Route Details", "IP Address",
        'Folder Name Search with RegEx Match',
    ],
    'objtype': 'sensor',
}
response = handler.get(**get_kwargs)

# export the object to a string
# (we could just as easily export to a file using export_to_report_file)
export_kwargs['obj'] = response
export_str = handler.export_obj(**export_kwargs)


print ""
print "print the export_str returned from export_obj():"

out = export_str
if len(out.splitlines()) > 15:
    out = out.splitlines()[0:15]
    out.append('..trimmed for brevity..')
    out = '\n'.join(out)

print out
```

Output from Python Code

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

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

Export basetype json explode false

Export a BaseType from getting objects as JSON with false for explode_json_string_values

Example Python Code

```
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"] = False

# get the objects that will provide the basetype that we want to use
get_kwargs = {
    'name': [
        "Computer Name", "IP Route Details", "IP Address",
        'Folder Name Search with RegEx Match',
    ],
    'objtype': 'sensor',
}
response = handler.get(**get_kwargs)
```

```
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

```
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
```

```
# 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",
        'Folder Name Search with RegEx Match',
    ],
    'objtype': 'sensor',
}
response = handler.get(**get_kwargs)

# export the object to a string
# (we could just as easily export to a file using export_to_report_file)
export_kwargs['obj'] = response
export_str = handler.export_obj(**export_kwargs)


print ""
print "print the export_str returned from export_obj():"
```

```

out = export_str
if len(out.splitlines()) > 15:
    out = out.splitlines()[0:15]
    out.append('..trimmed for brevity..')
    out = '\n'.join(out)

print out
```

Output from Python Code

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

print the export_str returned from export_obj():
{
  "_type": "sensors",
  "sensor": [
    {
      "_type": "sensor",
      "category": "Reserved",
      "description": "The assigned name of the client machine.\nExample: workstation-1.company.com",
      "exclude_from_parse_flag": 0,
      "hash": 3409330187,
      "hidden_flag": 0,
      "id": 3,
      "ignore_case_flag": 1,
      "max_age_seconds": 86400,
      "name": "Computer Name",
      "queries": {
..trimmed for brevity..
```

Export basetype xml default options

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

Example Python Code

```
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)
```

```
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

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

print the export_str returned from export_obj():
<sensors><cache_info /><sensor><category>Reserved</category><preview_sensor_flag /><hash>3409330187<
Example: workstation-1.company.com</description><string_hints /><subcolumns /><metadata /><parameter
Example:  172.16.0.0|192.168.1.1|255.255.0.0|UG|100|eth0</description><string_hints /><subcolumns><s
Set objWMIService = GetObject(&amp;quot;winmgmts:&amp;quot; _
    &amp;amp; &amp;quot;{impersonationLevel=impersonate}!\\&amp;quot; &amp;amp; strComputer &amp;amp
Set collip = objWMIService.ExecQuery(&amp;quot;select * from win32_networkadapterconfiguration where
dim ipaddrs()
ipcount = 0
for each ipItem in collip
    for each ipaddr in ipItem.IPAddress
        ipcount = ipcount + 1
    next
next
redim ipaddrs(ipcount)
..trimmed for brevity..
```

Export basetype xml minimal false

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

Example Python Code

```
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
```

```
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
```

```

Set collip = objWMIService.ExecQuery(&amp;quot;select * from win32_networkadapterconfiguration where
dim ipaddrs()
ipcount = 0
for each ipItem in collip
    for each ipaddr in ipItem.IPAddress
        ipcount = ipcount + 1
    next
next
redim ipaddrs(ipcount)
..trimmed for brevity..
```

Export basetype xml minimal true

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

Example Python Code

```
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"] = True

# get the objects that will provide the basetype that we want to use
get_kwargs = {
    'name': [
        "Computer Name", "IP Route Details", "IP Address",
        'Folder Name Search with RegEx Match',
    ],
    'objtype': 'sensor',
}
response = handler.get(**get_kwargs)

# export the object to a string
# (we could just as easily export to a file using export_to_report_file)
export_kwargs['obj'] = response
export_str = handler.export_obj(**export_kwargs)


print ""
print "print the export_str returned from export_obj():"

out = export_str
if len(out.splitlines()) > 15:
    out = out.splitlines()[0:15]
    out.append('..trimmed for brevity..')
    out = '\n'.join(out)

print out
```

Output from Python Code

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

print the export_str returned from export_obj():
<sensors><sensor><category>Reserved</category><hash>3409330187</hash><name>Computer Name</name><hidd
Example: workstation-1.company.com</description><queries><query><platform>Windows</platform><script_
Example:  172.16.0.0|192.168.1.1|255.255.0.0|UG|100|eth0</description><subcolumns><subcolumn><index>
Set objWMIService = GetObject(&amp;quot;winmgmts:&amp;quot; _
    &amp;amp; &amp;quot;{impersonationLevel=impersonate}!\\&amp;quot; &amp;amp; strComputer &amp;amp

Set collip = objWMIService.ExecQuery(&amp;quot;select * from win32_networkadapterconfiguration where
dim ipaddrs()
ipcount = 0
for each ipItem in collip
    for each ipaddr in ipItem.IPAddress
        ipcount = ipcount + 1
    next
next
```

```
redim ipaddrs(ipcount)
..trimmed for brevity..
```

Export basetype csv with explode false

Export a BaseType from getting objects as CSV with false for explode_json_string_values

Example Python Code

```
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["explode_json_string_values"] = False

# get the objects that will provide the basetype that we want to use
get_kwargs = {
    'name': [
        "Computer Name", "IP Route Details", "IP Address",
        'Folder Name Search with RegEx Match',
    ],
    'objtype': 'sensor',
}
response = handler.get(**get_kwargs)

# export the object to a string
# (we could just as easily export to a file using export_to_report_file)
export_kwargs['obj'] = response
export_str = handler.export_obj(**export_kwargs)


print ""
print "print the export_str returned from export_obj():"

out = export_str
if len(out.splitlines()) > 15:
    out = out.splitlines()[0:15]
    out.append('..trimmed for brevity..')
    out = '\n'.join(out)

print out
```

Output from Python Code

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

print the export_str returned from export_obj():
category,creation_time,delimiter,description,exclude_from_parse_flag,hash,hidden_flag,id,ignore_case
Reserved,,,"The assigned name of the client machine.
Example: workstation-1.company.com",0,3409330187,0,3,1,,86400,,,,,Computer Name,,Windows,select CSNa
Network,2015-08-07T13:22:12,|,"Returns IPv4 network routes, filtered to exclude noise. With Flags, M
Example:  172.16.0.0|192.168.1.1|255.255.0.0|UG|100|eth0",1,435227963,0,552,1,Jim Olsen,60,0,defined
Set objWMIService = GetObject(&amp;quot;winmgmts:&amp;quot; _
    &amp; &amp;quot;{impersonationLevel=impersonate}!\\&amp;quot; &amp; strComputer &amp; &amp;quot;\root\cimv2&

Set collip = objWMIService.ExecQuery(&amp;quot;select * from win32_networkadapterconfiguration where IPE
dim ipaddrs()
ipcount = 0
for each ipItem in collip
    for each ipaddr in ipItem.IPAddress
        ipcount = ipcount + 1
    next
..trimmed for brevity..
```

Export basetype csv with explode true

Export a BaseType from getting objects as CSV with true for explode_json_string_values

Example Python Code

```
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["explode_json_string_values"] = True

# get the objects that will provide the basetype that we want to use
get_kwargs = {
    'name': [
        "Computer Name", "IP Route Details", "IP Address",
        'Folder Name Search with RegEx Match',
    ],
    'objtype': 'sensor',
}
response = handler.get(**get_kwargs)
```

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

Export basetype csv with sort empty list

Export a BaseType from getting objects as CSV with an empty list for header_sort

Example Python Code

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

```
# 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",
        'Folder Name Search with RegEx Match',
    ],
    'objtype': 'sensor',
}
response = handler.get(**get_kwargs)

# export the object to a string
# (we could just as easily export to a file using export_to_report_file)
export_kwargs['obj'] = response
export_str = handler.export_obj(**export_kwargs)


print ""
print "print the export_str returned from export_obj():"
```

```

out = export_str
if len(out.splitlines()) > 15:
    out = out.splitlines()[0:15]
    out.append('..trimmed for brevity..')
    out = '\n'.join(out)

print out
```

Output from Python Code

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

print the export_str returned from export_obj():
category,creation_time,delimiter,description,exclude_from_parse_flag,hash,hidden_flag,id,ignore_case
Reserved,,,"The assigned name of the client machine.
Example: workstation-1.company.com",0,3409330187,0,3,1,,86400,,,,,Computer Name,,Windows,select CSNa
Network,2015-08-07T13:22:12,|,"Returns IPv4 network routes, filtered to exclude noise. With Flags, M
Example:  172.16.0.0|192.168.1.1|255.255.0.0|UG|100|eth0",1,435227963,0,552,1,Jim Olsen,60,0,defined
Set objWMIService = GetObject(&amp;quot;winmgmts:&amp;quot; _
    &amp; &amp;quot;{impersonationLevel=impersonate}!\\&amp;quot; &amp; strComputer &amp; &amp;quot;\root\cimv2&

Set collip = objWMIService.ExecQuery(&amp;quot;select * from win32_networkadapterconfiguration where IPE
dim ipaddrs()
ipcount = 0
for each ipItem in collip
    for each ipaddr in ipItem.IPAddress
        ipcount = ipcount + 1
    next
..trimmed for brevity..
```

Export basetype csv with sort true

Export a BaseType from getting objects as CSV with true for header_sort

Example Python Code

```
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)
```

```
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

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

print the export_str returned from export_obj():
category,creation_time,delimiter,description,exclude_from_parse_flag,hash,hidden_flag,id,ignore_case
Reserved,,,"The assigned name of the client machine.
Example: workstation-1.company.com",0,3409330187,0,3,1,,86400,,,,,Computer Name,,Windows,select CSNa
Network,2015-08-07T13:22:12,|,"Returns IPv4 network routes, filtered to exclude noise. With Flags, M
Example:  172.16.0.0|192.168.1.1|255.255.0.0|UG|100|eth0",1,435227963,0,552,1,Jim Olsen,60,0,defined
Set objWMIService = GetObject(&quot;winmgmts:&quot; _
    &amp; &quot;{impersonationLevel=impersonate}!\\&quot; &amp; strComputer &amp; &quot;\root\cimv2&

Set collip = objWMIService.ExecQuery(&quot;select * from win32_networkadapterconfiguration where IPE
dim ipaddrs()
ipcount = 0
for each ipItem in collip
    for each ipaddr in ipItem.IPAddress
        ipcount = ipcount + 1
    next
..trimmed for brevity..
```

Export basetype csv with sort list

Export a BaseType from getting objects as CSV with name and description for header_sort

Example Python Code

```python
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
```

```
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,
```

```
Set objWMIService = GetObject(&quot;winmgmts:&quot; _
    &amp; &quot;{impersonationLevel=impersonate}!\\&quot; &amp; strComputer &amp; &quot;\root\cimv2&

Set collip = objWMIService.ExecQuery(&quot;select * from win32_networkadapterconfiguration where IPE
dim ipaddrs()
ipcount = 0
for each ipItem in collip
    for each ipaddr in ipItem.IPAddress
        ipcount = ipcount + 1
    next
..trimmed for brevity..
```

Export basetype json default options

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

Example Python Code

```python
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'

# get the objects that will provide the basetype that we want to use
get_kwargs = {
    'name': [
        "Computer Name", "IP Route Details", "IP Address",
        'Folder Name Search with RegEx Match',
    ],
    'objtype': 'sensor',
}
response = handler.get(**get_kwargs)

# export the object to a string
# (we could just as easily export to a file using export_to_report_file)
export_kwargs['obj'] = response
export_str = handler.export_obj(**export_kwargs)


print ""
print "print the export_str returned from export_obj():"

out = export_str
if len(out.splitlines()) > 15:
    out = out.splitlines()[0:15]
    out.append('..trimmed for brevity..')
    out = '\n'.join(out)

print out
```

Output from Python Code

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

print the export_str returned from export_obj():
{
  "_type": "sensors",
  "sensor": [
    {
      "_type": "sensor",
      "category": "Reserved",
      "description": "The assigned name of the client machine.\nExample: workstation-1.company.com",
      "exclude_from_parse_flag": 0,
      "hash": 3409330187,
      "hidden_flag": 0,
      "id": 3,
      "ignore_case_flag": 1,
      "max_age_seconds": 86400,
      "name": "Computer Name",
      "queries": {
```

```
..trimmed for brevity..
```

Export basetype json type true

Export a BaseType from getting objects as JSON with true for include_type

Example Python Code

```
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["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

```
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["explode_json_string_values"] = u'bad'

# get the objects that will provide the basetype that we want to use
```

```
get_kwargs = {
    'name': [
        "Computer Name", "IP Route Details", "IP Address",
        'Folder Name Search with RegEx Match',
    ],
    'objtype': 'sensor',
}
response = handler.get(**get_kwargs)
export_kwargs['obj'] = response

# export the object to a string
# this should throw an exception: pytan.exceptions.HandlerError
import traceback

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

Output from Python Code

```
Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
Traceback (most recent call last):
  File "<string>", line 66, in <module>
  File "/Users/jolsen/gh/pytan/lib/pytan/utils.py", line 2699, in wrap
    ret = f(*args, **kwargs)
  File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 1084, in export_obj
    pytan.utils.check_dictkey(**check_args)
  File "/Users/jolsen/gh/pytan/lib/pytan/utils.py", line 2685, in check_dictkey
    raise pytan.exceptions.HandlerError(err(key, valid_types, k_type))
HandlerError: 'explode_json_string_values' must be one of [<type 'bool'>], you supplied <type 'unicc
```

Invalid export basetype csv bad sort sub type

Export a BaseType from getting objects using a bad header_sort

Example Python Code

```
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",
        'Folder Name Search with RegEx Match',
    ],
    'objtype': 'sensor',
}
response = handler.get(**get_kwargs)
export_kwargs['obj'] = response

# export the object to a string
# this should throw an exception: pytan.exceptions.HandlerError
import traceback

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

Output from Python Code

```
Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
Traceback (most recent call last):
  File "<string>", line 66, in <module>
```

```
  File "/Users/jolsen/gh/pytan/lib/pytan/utils.py", line 2699, in wrap
    ret = f(*args, **kwargs)
  File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 1084, in export_obj
    pytan.utils.check_dictkey(**check_args)
  File "/Users/jolsen/gh/pytan/lib/pytan/utils.py", line 2692, in check_dictkey
    raise pytan.exceptions.HandlerError(err(key, valid_list_types, list_types))
HandlerError: 'header_sort' must be a list of [<type 'str'>, <type 'unicode'>], you supplied [<type
```

Invalid export basetype csv bad sort type

Export a BaseType from getting objects using a bad header_sort

Example Python Code

```
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'bad'

# get the objects that will provide the basetype that we want to use
get_kwargs = {
    'name': [
        "Computer Name", "IP Route Details", "IP Address",
        'Folder Name Search with RegEx Match',
    ],
    'objtype': 'sensor',
}
response = handler.get(**get_kwargs)
export_kwargs['obj'] = response

# export the object to a string
# this should throw an exception: pytan.exceptions.HandlerError
import traceback

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

Output from Python Code

```
Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
Traceback (most recent call last):
  File "<string>", line 66, in <module>
  File "/Users/jolsen/gh/pytan/lib/pytan/utils.py", line 2699, in wrap
    ret = f(*args, **kwargs)
  File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 1084, in export_obj
    pytan.utils.check_dictkey(**check_args)
  File "/Users/jolsen/gh/pytan/lib/pytan/utils.py", line 2685, in check_dictkey
    raise pytan.exceptions.HandlerError(err(key, valid_types, k_type))
HandlerError: 'header_sort' must be one of [<type 'bool'>, <type 'list'>, <type 'tuple'>], you suppl
```

Invalid export basetype xml bad minimal type

Export a BaseType from getting objects using a bad minimal

Example Python Code

```
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)
export_kwargs['obj'] = response

# export the object to a string
# this should throw an exception: pytan.exceptions.HandlerError
import traceback

try:
    handler.export_obj(**export_kwargs)
except Exception as e:
```

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

Output from Python Code

```
Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
Traceback (most recent call last):
  File "<string>", line 66, in <module>
  File "/Users/jolsen/gh/pytan/lib/pytan/utils.py", line 2699, in wrap
    ret = f(*args, **kwargs)
  File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 1084, in export_obj
    pytan.utils.check_dictkey(**check_args)
  File "/Users/jolsen/gh/pytan/lib/pytan/utils.py", line 2685, in check_dictkey
    raise pytan.exceptions.HandlerError(err(key, valid_types, k_type))
HandlerError: 'minimal' must be one of [<type 'bool'>], you supplied <type 'unicode'>!
```

Invalid export basetype json bad include type

Export a BaseType from getting objects using a bad include_type

Example Python Code

```
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["include_type"] = u'bad'

# get the objects that will provide the basetype that we want to use
get_kwargs = {
    'name': [
        "Computer Name", "IP Route Details", "IP Address",
        'Folder Name Search with RegEx Match',
    ],
    'objtype': 'sensor',
}
response = handler.get(**get_kwargs)
export_kwargs['obj'] = response

# export the object to a string
# this should throw an exception: pytan.exceptions.HandlerError
import traceback

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

Output from Python Code

```
Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
Traceback (most recent call last):
  File "<string>", line 66, in <module>
  File "/Users/jolsen/gh/pytan/lib/pytan/utils.py", line 2699, in wrap
    ret = f(*args, **kwargs)
  File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 1084, in export_obj
    pytan.utils.check_dictkey(**check_args)
  File "/Users/jolsen/gh/pytan/lib/pytan/utils.py", line 2685, in check_dictkey
    raise pytan.exceptions.HandlerError(err(key, valid_types, k_type))
HandlerError: 'include_type' must be one of [<type 'bool'>], you supplied <type 'unicode'>!
```

Invalid export basetype json bad explode type

Export a BaseType from getting objects using a bad explode_json_string_values

Example Python Code

```
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",
        'Folder Name Search with RegEx Match',
    ],
    'objtype': 'sensor',
}
response = handler.get(**get_kwargs)
export_kwargs['obj'] = response
```

```

# export the object to a string
# this should throw an exception: pytan.exceptions.HandlerError
import traceback

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

Output from Python Code

```
Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
Traceback (most recent call last):
  File "<string>", line 66, in <module>
  File "/Users/jolsen/gh/pytan/lib/pytan/utils.py", line 2699, in wrap
    ret = f(*args, **kwargs)
  File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 1084, in export_obj
    pytan.utils.check_dictkey(**check_args)
  File "/Users/jolsen/gh/pytan/lib/pytan/utils.py", line 2685, in check_dictkey
    raise pytan.exceptions.HandlerError(err(key, valid_types, k_type))
HandlerError: 'explode_json_string_values' must be one of [<type 'bool'>], you supplied <type 'unicc
```

Invalid export basetype bad format

Export a BaseType from getting objects using a bad export_format

Example Python Code

```
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'bad'

# get the objects that will provide the basetype that we want to use
get_kwargs = {
    'name': [
        "Computer Name", "IP Route Details", "IP Address",
        'Folder Name Search with RegEx Match',
    ],
    'objtype': 'sensor',
}
response = handler.get(**get_kwargs)
export_kwargs['obj'] = response

# export the object to a string
# this should throw an exception: pytan.exceptions.HandlerError
import traceback

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

Output from Python Code

```
Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
Traceback (most recent call last):
  File "<string>", line 65, in <module>
  File "/Users/jolsen/gh/pytan/lib/pytan/utils.py", line 2699, in wrap
    ret = f(*args, **kwargs)
  File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 1078, in export_obj
    raise pytan.exceptions.HandlerError(err)
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.

class taniumpy.object_types.action.**Action**
Bases: taniumpy.object_types.base.BaseType

_soap_tag = 'action'

class taniumpy.object_types.action_list.**ActionList**
Bases: taniumpy.object_types.base.BaseType

_soap_tag = 'actions'

class taniumpy.object_types.action_list_info.**ActionListInfo**
Bases: taniumpy.object_types.base.BaseType

_soap_tag = 'info'

class taniumpy.object_types.action_stop.**ActionStop**
Bases: taniumpy.object_types.base.BaseType

_soap_tag = 'action_stop'

class taniumpy.object_types.action_stop_list.**ActionStopList**
Bases: taniumpy.object_types.base.BaseType

_soap_tag = 'action_stops'

class taniumpy.object_types.archived_question.**ArchivedQuestion**
Bases: taniumpy.object_types.base.BaseType

_soap_tag = 'archived_question'

class taniumpy.object_types.archived_question_list.**ArchivedQuestionList**
Bases: taniumpy.object_types.base.BaseType

_soap_tag = 'archived_questions'

class taniumpy.object_types.audit_data.**AuditData**
Bases: taniumpy.object_types.base.BaseType

_soap_tag = 'audit_data'

class taniumpy.object_types.base.**BaseType**(*simple_properties*, *complex_properties*, *list_properties*)
Bases: object

classmethod _from_json(*jsonable*)
Private helper to parse from JSON after type is instantiated

_soap_tag = None

append(*n*)
Allow adding to list.

Only supported on types that have a single property that is in list_properties

explode_json(*val*)

flatten_jsonable(*val*, *prefix*)

classmethod fromSOAPBody (*body*)
Parse body (text) and produce Python tanium objects.

This method assumes a single result_object, which may be a list or a single object.

classmethod fromSOAPElement (*el*)

static from_jsonable (*jsonable*)
Inverse of to_jsonable, with explode_json_string_values=False.

This can be used to import objects from serialized JSON. This JSON should come from BaseType.to_jsonable(explode_json_string_values=False, include+type=True)

Examples

```
>>> with open('question_list.json') as fd:
...     questions = json.loads(fd.read())
...     # is a list of serialized questions
...     question_objects = BaseType.from_jsonable(questions)
...     # will return a list of api.Question
```

toSOAPBody (*minimal=False*)

toSOAPElement (*minimal=False*)

to_flat_dict (*prefix=''*, *explode_json_string_values=False*)
Convert the object to a dict, flattening any lists or nested types

to_flat_dict_explode_json (*val*, *prefix=''*)
see if the value is json. If so, flatten it out into a dict

static to_json (*jsonable*, ***kwargs*)
Convert to a json string.

jsonable can be a single BaseType instance of a list of BaseType

to_jsonable (*explode_json_string_values=False*, *include_type=True*)

static write_csv (*fd*, *val*, *explode_json_string_values=False*, ***kwargs*)
Write 'val' to CSV. val can be a BaseType instance or a list of BaseType

This does a two-pass, calling to_flat_dict for each object, then finding the union of all headers, then writing out the value of each column for each object sorted by header name

explode_json_string_values attempts to see if any of the str values are parseable by json.loads, and if so treat each property as a column value

fd is a file-like object

exception taniumpy.object_types.base.**IncorrectTypeException** (*property*, *expected*, *actual*)
Bases: exceptions.Exception

Raised when a property is not of the expected type

class taniumpy.object_types.cache_filter.**CacheFilter**
Bases: taniumpy.object_types.base.BaseType

_soap_tag = 'filter'

class taniumpy.object_types.cache_filter_list.**CacheFilterList**
Bases: taniumpy.object_types.base.BaseType

_soap_tag = 'cache_filters'

class taniumpy.object_types.cache_info.**CacheInfo**
Bases: taniumpy.object_types.base.BaseType

_soap_tag = 'cache_info'

class taniumpy.object_types.client_count.**ClientCount**
Bases: taniumpy.object_types.base.BaseType

_soap_tag = 'client_count'

class taniumpy.object_types.client_status.**ClientStatus**
Bases: taniumpy.object_types.base.BaseType

_soap_tag = 'client_status'

class taniumpy.object_types.column.**Column**
Bases: object

classmethod fromSOAPElement(*el*)

class taniumpy.object_types.column_set.**ColumnSet**
Bases: object

classmethod fromSOAPElement(*el*)

class taniumpy.object_types.computer_group.**ComputerGroup**
Bases: taniumpy.object_types.base.BaseType

_soap_tag = 'computer_group'

class taniumpy.object_types.computer_group_list.**ComputerGroupList**
Bases: taniumpy.object_types.base.BaseType

_soap_tag = 'computer_groups'

class taniumpy.object_types.computer_group_spec.**ComputerGroupSpec**
Bases: taniumpy.object_types.base.BaseType

_soap_tag = 'computer_spec'

class taniumpy.object_types.computer_spec_list.**ComputerSpecList**
Bases: taniumpy.object_types.base.BaseType

_soap_tag = 'computer_specs'

class taniumpy.object_types.error_list.**ErrorList**
Bases: taniumpy.object_types.base.BaseType

_soap_tag = 'errors'

class taniumpy.object_types.filter.**Filter**
Bases: taniumpy.object_types.base.BaseType

_soap_tag = 'filter'

class taniumpy.object_types.filter_list.**FilterList**
Bases: taniumpy.object_types.base.BaseType

_soap_tag = 'filters'

class taniumpy.object_types.group.**Group**
Bases: taniumpy.object_types.base.BaseType

_soap_tag = 'group'

class taniumpy.object_types.group_list.**GroupList**
Bases: taniumpy.object_types.base.BaseType

_soap_tag = 'groups'

class taniumpy.object_types.metadata_item.**MetadataItem**
Bases: taniumpy.object_types.base.BaseType

_soap_tag = 'item'

class taniumpy.object_types.metadata_list.**MetadataList**
Bases: taniumpy.object_types.base.BaseType

_soap_tag = 'metadata'

class taniumpy.object_types.object_list.**ObjectList**
Bases: taniumpy.object_types.base.BaseType

_soap_tag = 'object_list'

class taniumpy.object_types.options.**Options**
Bases: taniumpy.object_types.base.BaseType

_soap_tag = 'options'

class taniumpy.object_types.package_file.**PackageFile**
Bases: taniumpy.object_types.base.BaseType

_soap_tag = 'file'

class taniumpy.object_types.package_file_list.**PackageFileList**
Bases: taniumpy.object_types.base.BaseType

_soap_tag = 'package_files'

class taniumpy.object_types.package_file_status.**PackageFileStatus**
Bases: taniumpy.object_types.base.BaseType

_soap_tag = 'status'

class taniumpy.object_types.package_file_status_list.**PackageFileStatusList**
Bases: taniumpy.object_types.base.BaseType

_soap_tag = 'file_status'

class taniumpy.object_types.package_file_template.**PackageFileTemplate**
Bases: taniumpy.object_types.base.BaseType

_soap_tag = 'file_template'

class taniumpy.object_types.package_file_template_list.**PackageFileTemplateList**
Bases: taniumpy.object_types.base.BaseType

_soap_tag = 'file_templates'

class taniumpy.object_types.package_spec.**PackageSpec**
Bases: taniumpy.object_types.base.BaseType

_soap_tag = 'package_spec'

class taniumpy.object_types.package_spec_list.**PackageSpecList**
Bases: taniumpy.object_types.base.BaseType

_soap_tag = 'package_specs'

class taniumpy.object_types.parameter.**Parameter**
Bases: taniumpy.object_types.base.BaseType

_soap_tag = 'parameter'

class taniumpy.object_types.parameter_list.**ParameterList**
Bases: taniumpy.object_types.base.BaseType

_soap_tag = 'parameters'

class taniumpy.object_types.parse_job.**ParseJob**
Bases: taniumpy.object_types.base.BaseType

_soap_tag = 'parse_job'

class taniumpy.object_types.parse_job_list.**ParseJobList**
Bases: taniumpy.object_types.base.BaseType

_soap_tag = 'parse_jobs'

class taniumpy.object_types.parse_result.**ParseResult**
Bases: taniumpy.object_types.base.BaseType

_soap_tag = 'parse_result'

class taniumpy.object_types.parse_result_group.**ParseResultGroup**
Bases: taniumpy.object_types.base.BaseType

_soap_tag = 'parse_result_group'

class taniumpy.object_types.parse_result_group_list.**ParseResultGroupList**
Bases: taniumpy.object_types.base.BaseType

_soap_tag = 'parse_result_groups'

class taniumpy.object_types.parse_result_list.**ParseResultList**
Bases: taniumpy.object_types.base.BaseType

_soap_tag = 'parse_results'

class taniumpy.object_types.permission_list.**PermissionList**
Bases: taniumpy.object_types.base.BaseType

_soap_tag = 'permissions'

class taniumpy.object_types.plugin.**Plugin**
Bases: taniumpy.object_types.base.BaseType

_soap_tag = 'plugin'

class taniumpy.object_types.plugin_argument.**PluginArgument**
Bases: taniumpy.object_types.base.BaseType

_soap_tag = 'argument'

class taniumpy.object_types.plugin_argument_list.**PluginArgumentList**
Bases: taniumpy.object_types.base.BaseType

_soap_tag = 'arguments'

class taniumpy.object_types.plugin_command_list.**PluginCommandList**
Bases: taniumpy.object_types.base.BaseType

_soap_tag = 'commands'

class taniumpy.object_types.plugin_list.**PluginList**
Bases: taniumpy.object_types.base.BaseType

_soap_tag = 'plugins'

class taniumpy.object_types.plugin_schedule.**PluginSchedule**
Bases: taniumpy.object_types.base.BaseType

_soap_tag = 'plugin_schedule'

class taniumpy.object_types.plugin_schedule_list.**PluginScheduleList**
Bases: taniumpy.object_types.base.BaseType

_soap_tag = 'plugin_schedules'

class taniumpy.object_types.plugin_sql.**PluginSql**
Bases: taniumpy.object_types.base.BaseType

_soap_tag = 'sql_response'

class taniumpy.object_types.plugin_sql_column.**PluginSqlColumn**
Bases: taniumpy.object_types.base.BaseType

_soap_tag = 'columns'

class taniumpy.object_types.plugin_sql_result.**PluginSqlResult**
Bases: taniumpy.object_types.base.BaseType

_soap_tag = 'result_row'

class taniumpy.object_types.question.**Question**
Bases: taniumpy.object_types.base.BaseType

_soap_tag = 'question'

class taniumpy.object_types.question_list.**QuestionList**
Bases: taniumpy.object_types.base.BaseType

_soap_tag = 'questions'

class taniumpy.object_types.question_list_info.**QuestionListInfo**
Bases: taniumpy.object_types.base.BaseType

_soap_tag = 'info'

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)

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*)

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*)

class taniumpy.object_types.saved_action.**SavedAction**
Bases: taniumpy.object_types.base.BaseType

_soap_tag = 'saved_action'

class taniumpy.object_types.saved_action_approval.**SavedActionApproval**
Bases: taniumpy.object_types.base.BaseType

_soap_tag = 'saved_action_approval'

class taniumpy.object_types.saved_action_list.**SavedActionList**
Bases: taniumpy.object_types.base.BaseType

_soap_tag = 'saved_actions'

class taniumpy.object_types.saved_action_policy.**SavedActionPolicy**
Bases: taniumpy.object_types.base.BaseType

_soap_tag = 'policy'

class taniumpy.object_types.saved_action_row_id_list.**SavedActionRowIdList**
Bases: taniumpy.object_types.base.BaseType

_soap_tag = 'row_ids'

class taniumpy.object_types.saved_question.**SavedQuestion**
Bases: taniumpy.object_types.base.BaseType

_soap_tag = 'saved_question'

class taniumpy.object_types.saved_question_list.**SavedQuestionList**
Bases: taniumpy.object_types.base.BaseType

_soap_tag = 'saved_questions'

class taniumpy.object_types.select.**Select**
Bases: taniumpy.object_types.base.BaseType

_soap_tag = 'select'

class taniumpy.object_types.select_list.**SelectList**
Bases: taniumpy.object_types.base.BaseType

_soap_tag = 'selects'

class taniumpy.object_types.sensor.**Sensor**
Bases: taniumpy.object_types.base.BaseType

_soap_tag = 'sensor'

class taniumpy.object_types.sensor_list.**SensorList**
Bases: taniumpy.object_types.base.BaseType

_soap_tag = 'sensors'

class taniumpy.object_types.sensor_query.**SensorQuery**
Bases: taniumpy.object_types.base.BaseType

_soap_tag = 'query'

class taniumpy.object_types.sensor_query_list.**SensorQueryList**
Bases: taniumpy.object_types.base.BaseType

_soap_tag = 'queries'

class taniumpy.object_types.string_hint_list.**StringHintList**
Bases: taniumpy.object_types.base.BaseType

_soap_tag = 'string_hints'

class taniumpy.object_types.sensor_subcolumn.**SensorSubcolumn**
Bases: taniumpy.object_types.base.BaseType

_soap_tag = 'subcolumn'

class taniumpy.object_types.sensor_subcolumn_list.**SensorSubcolumnList**
Bases: taniumpy.object_types.base.BaseType

_soap_tag = 'subcolumns'

class taniumpy.object_types.soap_error.**SoapError**
Bases: taniumpy.object_types.base.BaseType

_soap_tag = 'soap_error'

class taniumpy.object_types.system_setting.**SystemSetting**
Bases: taniumpy.object_types.base.BaseType

_soap_tag = 'system_setting'

class taniumpy.object_types.system_setting_list.**SystemSettingList**
Bases: taniumpy.object_types.base.BaseType

_soap_tag = 'system_settings'

class taniumpy.object_types.system_status_aggregate.**SystemStatusAggregate**
Bases: taniumpy.object_types.base.BaseType

_soap_tag = 'aggregate'

class taniumpy.object_types.system_status_list.**SystemStatusList**
Bases: taniumpy.object_types.base.BaseType

_soap_tag = 'system_status'

class taniumpy.object_types.upload_file.**UploadFile**
Bases: taniumpy.object_types.base.BaseType

_soap_tag = 'upload_file'

class taniumpy.object_types.upload_file_list.**UploadFileList**
Bases: taniumpy.object_types.base.BaseType

_soap_tag = 'file_parts'

class taniumpy.object_types.upload_file_status.**UploadFileStatus**
Bases: taniumpy.object_types.base.BaseType

_soap_tag = 'upload_file_status'

class taniumpy.object_types.user.**User**
Bases: taniumpy.object_types.base.BaseType

_soap_tag = 'user'

class taniumpy.object_types.user_list.**UserList**
Bases: taniumpy.object_types.base.BaseType

_soap_tag = 'users'

class taniumpy.object_types.user_role.**UserRole**
Bases: taniumpy.object_types.base.BaseType

_soap_tag = 'role'

class taniumpy.object_types.user_role_list.**UserRoleList**
Bases: taniumpy.object_types.base.BaseType

_soap_tag = 'roles'

class taniumpy.object_types.version_aggregate.**VersionAggregate**
Bases: taniumpy.object_types.base.BaseType

_soap_tag = 'version'

class taniumpy.object_types.version_aggregate_list.**VersionAggregateList**
Bases: taniumpy.object_types.base.BaseType

_soap_tag = 'versions'

class taniumpy.object_types.white_listed_url.**WhiteListedUrl**
Bases: taniumpy.object_types.base.BaseType

_soap_tag = 'white_listed_url'

class taniumpy.object_types.white_listed_url_list.**WhiteListedUrlList**
Bases: taniumpy.object_types.base.BaseType

_soap_tag = 'white_listed_urls'

class taniumpy.object_types.xml_error.**XmlError**
Bases: taniumpy.object_types.base.BaseType

_soap_tag = 'error'

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

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.

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.

CHAPTER

TWO

INDICES AND TABLES

- *genindex*
- *modindex*
- *search*

d

p

r

t

X

Symbols

A

B

C

D

E

F

G

H

I

J

L

M

N

O

P

Q

R

S

T

U

V

W

X