PyTan Documentation

Release 2.1.0

Jim Olsen

CONTENTS

1	20002	Table of Contents					
		PyTan Introduction					
	1.2	pytan package	3				
	1.3	taniumpy package					
	1.4	xmltodict module	379				
	1.5	ddt module	381				
	1.6	threaded_http module	381				
	1.7	requests package	382				
2	Indices and tables						
Рy	Python Module Index						
In	Index						

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 describes 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

3

- 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.

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
   •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.SessionSession 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 Re-
          sult 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': 'Sensorl',
... '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

```
\textbf{Parametersobj}: \texttt{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 ini-
      tial 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_MAPSvalid filter dictionaries for filters
pytan.constants.OPTION_MAPSvalid 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': 'Sensorl',
... '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
            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
        Parametersobi
                              taniumpy.object_types.result_set.ResultSet
            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
                     Parameterssse 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
                              \mathbf{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.SavedAction_approval.SavedAction_approval.SavedAction_approval.SavedAction_approval.SavedAction_approval.SavedAction_approval.SavedAction_approval.SavedAction_approval.SavedAction_approval.SavedAction_approval.SavedAction_approval.SavedAction_approval.SavedAction_approval.SavedAction_approval.SavedAction_approval.SavedAction_approval.SavedAction_approval.SavedAction_approval.SavedAction_approval.SavedAction_approval.SavedAction_approval.SavedAction_approval.SavedAction_approval.SavedAction_approval.SavedAction_approval.SavedAction_approval.SavedAction_approval.SavedAction_approval.SavedAction_approval.SavedAction_approval.SavedAction_approval.SavedAction_approval.SavedAction_approval.SavedAction_approval.SavedAction_approval.SavedAction_approval.SavedAction_approval.SavedAction_approval.SavedAction_approval.SavedAction_approval.SavedAction_approval.SavedAction_approval.SavedAction_approval.SavedAction_approval.SavedAction_approval.SavedAction_approval.SavedAction_approval.SavedAction_approval.SavedAction_approval.SavedAction_approval.SavedAction_approval.SavedAction_approval.SavedAction_approval.SavedAction_approval.SavedAction_approval.SavedAction_approval.SavedAction_approval.SavedAction_approval.SavedAction_approval.SavedAction_approval.SavedAction_approval.SavedAction_approval.SavedAction_approval.SavedAction_approval.SavedAction_approval.SavedAction_approval.SavedAction_approval.SavedAction_approval.SavedAction_approval.SavedAction_approval.SavedAction_approval.SavedAction_approval.SavedAction_approval.SavedAction_approval.SavedAction_approval.SavedAction_approval.SavedAction_approval.SavedAction_approval.SavedAction_approval.SavedAction_approval.SavedAction_approval.SavedAction_approval.SavedAction_approval.SavedAction_approval.SavedAction_approval.SavedAction_approval.SavedAction_approval.SavedAction_approval.SavedAction_approval.SavedAction_approval.SavedAction_approval.SavedAction_approval.SavedAction_approval.SavedAction_appr
                                  •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
                                                     of
                                                            the
                                                                    following
                                                                                  depending
                                                                                                 on
                                   taniumpy.object_types.question.Question
               qtype:
                                                                                                 or
               taniumpy.object_types.saved_question.SavedQuestion
               •question_results: taniumpy.object_types.result_set.ResultSet
     See also:
     pytan.constants.Q_OBJ_MAPmaps 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 Re-
          sult 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 cre-
    ate and add the question object
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"
```

Notes

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

•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, optionaldefault: 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 Re-
      sult 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 re-
               sults 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
     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
```

Notes

```
•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
             ison 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_MAPS valid filters for filters
     pytan.constants.OPTION_MAPS valid options for filter_options
create_package (name, command, display_name="', file_urls=[], command_timeout_seconds=600,
                     expire seconds=600,
                                             parameters_json_file='',
                                                                          verify filters=[],
                     ify_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_MAPS valid filters for verify_filters
     pytan.constants.OPTION_MAPS valid 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_MAP maps 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

```
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
    Returnsresult: str
         the contents of exporting export_format
See also:
```

29

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 export-
pytan.handler.Handler.create_report_file() method that performs the actual work to
    write the report file
When performing a CSV export and importing that CSV into excel, keep in mind that Excel has a per cell
```

Notes

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
                        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 scenar-
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

$\textbf{exception} \; \texttt{pytan.exceptions.DefinitionParserError}$

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

$\textbf{exception} \; \texttt{pytan.exceptions.VersionParseError}$

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 Relation 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 in-
               valid/restricted matches
               •False: When XML cleaning the response_body, disable logging messages about in-
               valid/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 in-
  valid/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

Returnsobj: taniumpy.object_types.base.BaseType

deleted object

```
disable stats loop(sleep=None)
    Disables
                                                  which
                the
                       stats
                               loop
                                       thread.
                                                            will
                                                                   print
                                                                                   the
                                                                                          re-
    sults
                           pytan.sessions.Session.get server stats()
                                                                                       every
    pytan.sessions.Session.STATS_LOOP_SLEEP_SEC
        Parameterssleep: 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
                                                  which
                      stats
                                       thread,
                                                            will
                                                                   print
                                                                            out
                                                                                   the
                                                                                          re-
    sults
                 of
                           pytan.sessions.Session.get_server_stats()
                                                                                        every
    pytan.sessions.Session.STATS LOOP SLEEP SEC
        Parameterssleep: 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
                              checks
                                         self.STATS_LOOP_ENABLED
                                                                          before
                   which
                                                                                      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
        Parametersobj: taniumpy.object_types.base.BaseType
             object to find
        Returnsobj: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 appropri-
    ate taniumpy object
        Parametersobj: taniumpy.object_types.base.BaseType

    object to get result set for

        Returnsobj: 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.
        Parametersobj: 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 re-
         ceived 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_TARGETSlist 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
```

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

49

•False: throw exception if session_id fails

•default: self.HTTP_RETRY_COUNT

retry_count: int, optional

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

•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

•finished_count is calculated from a full GetResultData call that is parsed into self.action_result_map

```
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

- callback choose calling Any can to get data from the session by 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
```

the object (self.obj) itself.

Will generate a datetime string from self.EXPIRY_FALLBACK_SECS if unable to get the expiration from

1.2. pytan package 55

_derive_expiration(**kwargs)

Derive the expiration datetime string from a object

```
_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()

```
•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)
```

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

1.2. pytan package 57

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.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': ['<', 'properties of the constant of the con

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': 'User

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 attrexists 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 = '(?<!\\\)\\{(.*?)(?<!\\\)\\}'</pre>

The regex that is used to parse parameters from a human string. Ex: ala {key1=value1}

pytan.constants.PARAM_SPLIT_RE = '(?<!\\\),'</pre>

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: 'IPAG
          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, 'build': 4300, 'minor': 5, 'revision': 314}], 2: [{'major': 6, 'build': 4300, 'minor': 5, 'revision': 314}], 2: [{'major': 6, 'build': 4300, 'minor': 5, 'revision': 314}], 2: [{'major': 6, 'build': 4300, 'minor': 5, 'revision': 314}], 2: [{'major': 6, 'build': 4300, 'minor': 5, 'revision': 314}], 2: [{'major': 6, 'build': 4300, 'minor': 5, 'revision': 314}], 2: [{'major': 6, 'build': 4300, 'minor': 5, 'revision': 314}], 2: [{'major': 6, 'build': 4300, 'minor': 5, 'revision': 314}], 3: [{'major': 6, 'build': 4300, 'minor': 5, 'revision': 314}], 3: [{'major': 6, 'build': 4300, 'minor': 5, 'revision': 314}], 3: [{'major': 6, 'build': 4300, 'minor': 5, 'revision': 314}], 3: [{'major': 6, 'build': 4300, 'minor': 6, 'build': 6, '
          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
                                                                    delim='',
pytan.utils.build_param_objlist(obj,
                                                   user_params,
                                                                                  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
                           If user did not supply a value for a given parameter, throw a
                 •False:
                  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
                                                                                                  throw
                                                                                     empty,
                  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()
             all
                    loggers
                             currently
                                                                                                   exist
     Gets
                                           known
                                                           pythons
                                                                      logging
                                                                                  system
                                                                                            that
                                                                                                            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}}/4-\%H_{m_{d}}/5-\%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
          Returns socket 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

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. Argument Parser 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=[])
```

1.2. pytan package 73

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
          Parametersparser: argparse. ArgParse

    ArgParse object

              grp_names: list of str
                 •list of str of argument group names to get arguments for
          Returnsgrp_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
          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
                 •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
          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
```

```
•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
         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_create_json_object_args (parser, handler, obj, args)
     Process command line args supplied by user for create ison 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 create json object
             args: args object
                •args parsed from parser
         Returnsresponse: 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
         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_package()
pytan.binsupport.process_create_sensor_args (parser, handler, args)
     Process command line args supplied by user for create sensor 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_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
          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
                •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
          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 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
          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 re-
                 turn of pytan.handler.Handler.get_result_data()
pytan.binsupport.process_handler_args (parser, args)
     Process command line args supplied by user for handler
          Parametersparser: 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
```

79

•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.

$\verb|pytan.binsupport.setup_create_json_object_argparser|(obj, doc)|$

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.

81

```
pytan.binsupport.version_check (reqver)
```

Allows scripts using pytan to validate the version of the script aginst the version of pytan

Parametersrequer: str

•string containing version number to check against Exception

RaisesVersionMismatchError: Exception

•if pytan. __version__ is not greater or equal to requer

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], [#xDFFFE-#xDFFFF], [#xEFFFE-#xEFFFF], [#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
           Parameterstext: 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
           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
                                                      encoding='utf-8',
pytan.xml_clean.xml_cleaner(s,
                                                                                   clean_restricted=True,
                                       log_clean_messages=True, log_bad_characters=False, replace-
                                       ment=None, **kwargs)
     Removes invalid /restricted characters per XML 1.0 spec
           Parameterss: 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
           Returnsstr
                  •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_parameters_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_parameters_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)
10
    pytan_root_dir = os.path.dirname(parent_dir)
11
    lib_dir = os.path.join(pytan_root_dir, 'lib')
12
    path_adds = [lib_dir]
13
    for aa in path_adds:
15
        if aa not in sys.path:
16
            sys.path.append(aa)
17
18
19
    # connection info for Tanium Server
20
    USERNAME = "Tanium User"
21
    PASSWORD = "T@n!um"
22
    HOST = "172.16.31.128"
23
    PORT = "443"
24
25
    # Logging conrols
26
    LOGLEVEL = 2
27
   DEBUGFORMAT = False
28
29
   import tempfile
30
31
    import pytan
32
33
    handler = pytan.Handler(
34
        username=USERNAME,
        password=PASSWORD,
35
        host=HOST,
36
        port=PORT,
37
        loglevel=LOGLEVEL,
38
        debugformat=DEBUGFORMAT,
39
40
41
   print handler
```

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

```
# 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'])
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
```

```
Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
1
    2015-08-07 19:37:49,991 DEBUG
                                      pytan.handler.QuestionPoller: ID 1279: id resolved to 1279
2
                                      pytan.handler.QuestionPoller: ID 1279: expiration resolved to 2015-
    2015-08-07 19:37:49,991 DEBUG
3
   2015-08-07 19:37:49,991 DEBUG
                                      pytan.handler.QuestionPoller: ID 1279: query_text resolved to Get r
4
   2015-08-07 19:37:49,991 DEBUG
                                      pytan.handler.QuestionPoller: ID 1279: id resolved to 1279
   2015-08-07 19:37:49,991 DEBUG
                                      pytan.handler.QuestionPoller: ID 1279: Object Info resolved to Ques
6
   2015-08-07 19:37:49,997 DEBUG
                                      pytan.handler.QuestionPoller: ID 1279: Progress: Tested: 0, Passed:
   2015-08-07 19:37:49,997 DEBUG
                                      pytan.handler.QuestionPoller: ID 1279: Timing: Started: 2015-08-07
    2015-08-07 19:37:49,997 INFO
                                      pytan.handler.QuestionPoller: ID 1279: Progress Changed 0% (0 of 2)
    2015-08-07 19:37:55,007 DEBUG
                                      pytan.handler.QuestionPoller: ID 1279: Progress: Tested: 2, Passed:
10
    2015-08-07 19:37:55,008 DEBUG
                                      pytan.handler.QuestionPoller: ID 1279: Timing: Started: 2015-08-07
11
    2015-08-07 19:37:55,008 INFO
                                      pytan.handler.QuestionPoller: ID 1279: Progress Changed 100% (2 of
12
    2015-08-07 19:37:55,008 INFO
                                      pytan.handler.QuestionPoller: ID 1279: Reached Threshold of 99% (2
13
14
    Type of response: <type 'dict'>
15
16
17
    Pretty print of response:
    {'poller_object': <pytan.pollers.QuestionPoller object at 0x10a6c0410>,
18
     'poller_success': True,
     'question_object': <taniumpy.object_types.question.Question object at 0x10a7e7f50>,
20
     'question_results': <taniumpy.object_types.result_set.ResultSet object at 0x10a808190$,
21
     'saved_question_object': <taniumpy.object_types.saved_question.SavedQuestion object at 0x10a7ecb90>
22
23
    Equivalent Question if it were to be asked in the Tanium Console:
24
    Get number of machines
25
26
    CSV Results of response:
27
    Name, Silent Uninstall String, Uninstallable, Version
28
    Image Capture Extension, nothing, Not Uninstallable, 10.2
29
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
   Time Machine, nothing, Not Uninstallable, 1.3
33
34
   AppleGraphicsWarning, nothing, Not Uninstallable, 2.3.0
    soagent, nothing, Not Uninstallable, 7.0
35
   Feedback Assistant, nothing, Not Uninstallable, 4.1.3
36
   AinuIM, nothing, Not Uninstallable, 1.0
37
    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

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

Example Python Code

```
import os
    import sys
2
    sys.dont_write_bytecode = True
3
    # Determine our script name, script dir
5
    my_file = os.path.abspath(sys.argv[0])
6
    my_dir = os.path.dirname(my_file)
7
8
    # determine the pytan lib dir and add it to the path
    parent_dir = os.path.dirname(my_dir)
10
    pytan_root_dir = os.path.dirname(parent_dir)
    lib_dir = os.path.join(pytan_root_dir, 'lib')
13
   path_adds = [lib_dir]
14
    for aa in path_adds:
15
16
        if aa not in sys.path:
             sys.path.append(aa)
17
18
19
    # connection info for Tanium Server
20
    USERNAME = "Tanium User"
21
    PASSWORD = "T@n!um"
22
    HOST = "172.16.31.128"
23
    PORT = "443"
24
25
    # Logging conrols
26
    LOGLEVEL = 2
27
    DEBUGFORMAT = False
28
29
30
    import tempfile
31
32
    import pytan
    handler = pytan.Handler(
33
        username=USERNAME,
34
        password=PASSWORD,
35
36
        host=HOST,
37
        port=PORT,
        loglevel=LOGLEVEL,
        debugformat=DEBUGFORMAT,
39
40
41
   print handler
42
43
    # setup the arguments for the handler method
```

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

```
Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2
   Type of response: <type 'dict'>
3
4
   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>
10
11
12
   Equivalent Question if it were to be asked in the Tanium Console:
   Get Installed Applications from all machines
13
   CSV Results of response:
15
   Name, Silent Uninstall String, Uninstallable, Version
16
   Image Capture Extension, nothing, Not Uninstallable, 10.2
17
   Dictation, nothing, Not Uninstallable, 1.6.1
18
   Wish, nothing, Not Uninstallable, 8.5.9
19
   Uninstall AnyConnect, nothing, Not Uninstallable, 3.1.08009
   Time Machine, nothing, Not Uninstallable, 1.3
```

```
AppleGraphicsWarning, nothing, Not Uninstallable, 2.3.0
22
    soagent, nothing, Not Uninstallable, 7.0
23
    Feedback Assistant, nothing, Not Uninstallable, 4.1.3
24
   AinuIM, nothing, Not Uninstallable, 1.0
25
    vpndownloader, nothing, Not Uninstallable, 3.1.08009
    Pass Viewer, nothing, Not Uninstallable, 1.0
27
   ARDAgent, nothing, Not Uninstallable, 3.8.4
28
   OBEXAgent, nothing, Not Uninstallable, 4.3.5
29
   PressAndHold, nothing, Not Uninstallable, 1.2
30
    ..trimmed for brevity..
31
```

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
1
    import sys
2
    sys.dont_write_bytecode = True
3
    # 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
Q
    parent_dir = os.path.dirname(my_dir)
10
    pytan_root_dir = os.path.dirname(parent_dir)
11
    lib_dir = os.path.join(pytan_root_dir, 'lib')
12
   path_adds = [lib_dir]
13
14
    for aa in path_adds:
15
        if aa not in sys.path:
16
            sys.path.append(aa)
17
19
    # connection info for Tanium Server
20
    USERNAME = "Tanium User"
21
    PASSWORD = "T@n!um"
22
    HOST = "172.16.31.128"
23
   PORT = "443"
24
25
    # Logging conrols
26
    LOGLEVEL = 2
27
    DEBUGFORMAT = False
28
29
30
    import tempfile
31
32
    import pytan
   handler = pytan.Handler(
33
        username=USERNAME,
34
        password=PASSWORD,
35
36
        host=HOST,
37
        port=PORT,
        loglevel=LOGLEVEL,
```

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

```
Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2
   Type of response: <type 'dict'>
3
4
   Pretty print of response:
5
6
   { '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>
10
11
   Equivalent Question if it were to be asked in the Tanium Console:
12
   Get Installed Applications from all machines
13
14
   CSV Results of response:
```

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

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

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

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

```
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 Computation Poller: ID 1280: id resolved to 1280
```

```
2015-08-07 19:37:55,255 DEBUG
                                      pytan.handler.QuestionPoller: ID 1280: Object Info resolved to Ques
6
                                      pytan.handler.QuestionPoller: ID 1280: Progress: Tested: 0, Passed:
    2015-08-07 19:37:55,260 DEBUG
7
    2015-08-07 19:37:55,260 DEBUG
                                      pytan.handler.QuestionPoller: ID 1280: Timing: Started: 2015-08-07
                                      pytan.handler.QuestionPoller: ID 1280: Progress Changed 0% (0 of 2)
   2015-08-07 19:37:55,260 INFO
9
   2015-08-07 19:38:00,263 DEBUG
                                      pytan.handler.QuestionPoller: ID 1280: Progress: Tested: 1, Passed:
10
   2015-08-07 19:38:00,263 DEBUG
                                      pytan.handler.QuestionPoller: ID 1280: Timing: Started: 2015-08-07
11
   2015-08-07 19:38:00,264 INFO
                                      pytan.handler.QuestionPoller: ID 1280: Progress Changed 50% (1 of 2
12
   2015-08-07 19:38:05,271 DEBUG
                                      pytan.handler.QuestionPoller: ID 1280: Progress: Tested: 1, Passed:
13
   2015-08-07 19:38:05,271 DEBUG
                                      pytan.handler.QuestionPoller: ID 1280: Timing: Started: 2015-08-07
14
   2015-08-07 19:38:10,277 DEBUG
                                      pytan.handler.QuestionPoller: ID 1280: Progress: Tested: 2, Passed:
15
   2015-08-07 19:38:10,277 DEBUG
                                      pytan.handler.QuestionPoller: ID 1280: Timing: Started: 2015-08-07
16
    2015-08-07 19:38:10,277 INFO
                                      pytan.handler.QuestionPoller: ID 1280: Progress Changed 100% (2 of
17
    2015-08-07 19:38:10,277 INFO
                                      pytan.handler.QuestionPoller: ID 1280: Reached Threshold of 99% (2
18
19
    Type of response: <type 'dict'>
20
21
    Pretty print of response:
22
    {'poller_object': <pytan.pollers.QuestionPoller object at 0x10a808810>,
23
     'poller_success': True,
24
     'question_object': <taniumpy.object_types.question.Question object at 0x10a810650>,
25
     'question_results': <taniumpy.object_types.result_set.ResultSet object at 0x10a6c0410♭}
26
27
    Equivalent Question if it were to be asked in the Tanium Console:
28
    Get Computer Name and Installed Applications from all machines
29
    CSV Results of response:
31
    Computer Name, Name, Silent Uninstall String, Uninstallable, Version
32
    Casus-Belli.local, "Image Capture Extension
33
    Dictation
34
   Wish
35
   Uninstall AnyConnect
36
   Time Machine
37
   AppleGraphicsWarning
38
    soagent
39
   Feedback Assistant
40
   AinuIM
41
   vpndownloader
42
   Pass Viewer
43
44
    ARDAgent
   OBEXAgent
45
   PressAndHold
46
    ..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
Q
    parent_dir = os.path.dirname(my_dir)
10
    pytan_root_dir = os.path.dirname(parent_dir)
11
    lib_dir = os.path.join(pytan_root_dir, 'lib')
12
    path_adds = [lib_dir]
13
14
    for aa in path_adds:
15
        if aa not in sys.path:
16
17
             sys.path.append(aa)
18
19
    # connection info for Tanium Server
20
    USERNAME = "Tanium User"
21
    PASSWORD = "T@n!um"
22
    HOST = "172.16.31.128"
23
    PORT = "443"
24
25
    # Logging conrols
26
    LOGLEVEL = 2
27
    DEBUGFORMAT = False
28
29
    import tempfile
30
31
    import pytan
32
    handler = pytan.Handler(
33
        username=USERNAME,
34
        password=PASSWORD,
35
        host=HOST,
36
        port=PORT,
37
        loglevel=LOGLEVEL,
38
        debugformat=DEBUGFORMAT,
39
40
41
    print handler
42
43
44
    # setup the arguments for the handler method
    kwargs = {}
45
    kwargs["sensors"] = u'Computer Name'
46
    kwargs["qtype"] = u'manual'
47
48
    # call the handler with the ask method, passing in kwargs for arguments
49
    response = handler.ask(**kwargs)
50
    import pprint, io
51
52
    print ""
53
    print "Type of response: ", type(response)
54
55
    print ""
56
    print "Pretty print of response:"
57
    print pprint.pformat(response)
58
59
    print ""
60
    print "Equivalent Question if it were to be asked in the Tanium Console: "
61
    print response['question_object'].query_text
62
```

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

```
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
2
   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 (
   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
6
   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)
10
   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
11
   2015-08-07 19:38:15,351 INFO
                                     pytan.handler.QuestionPoller: ID 1281: Progress Changed 50% (1 of 2
12
   2015-08-07 19:38:20,357 DEBUG
                                     pytan.handler.QuestionPoller: ID 1281: Progress: Tested: 2, Passed:
13
                                     pytan.handler.QuestionPoller: ID 1281: Timing: Started: 2015-08-07
   2015-08-07 19:38:20,357 DEBUG
14
   2015-08-07 19:38:20,357 INFO
                                     pytan.handler.QuestionPoller: ID 1281: Progress Changed 100% (2 of
15
   2015-08-07 19:38:20,357 INFO
                                     pytan.handler.QuestionPoller: ID 1281: Reached Threshold of 99% (2
16
17
   Type of response: <type 'dict'>
18
19
   Pretty print of response:
20
   {'poller_object': <pytan.pollers.QuestionPoller object at 0x10a7ecb90>,
21
     'poller_success': True,
22
23
     'question_object': <taniumpy.object_types.question.Question object at 0x10a808290>,
     'question_results': <taniumpy.object_types.result_set.ResultSet object at 0x10a6133d0♭}
24
25
   Equivalent Question if it were to be asked in the Tanium Console:
26
   Get Computer Name from all machines
27
28
   CSV Results of response:
29
   Computer Name
30
   Casus-Belli.local
31
   JTANIUM1.localdomain
32
```

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

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

```
Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
   2015-08-07 19:38:20,411 DEBUG
                                     pytan.handler.QuestionPoller: ID 1282: id resolved to 1282
2
   2015-08-07 19:38:20,411 DEBUG
                                      pytan.handler.QuestionPoller: ID 1282: expiration resolved to 2015-
    2015-08-07 19:38:20,411 DEBUG
                                      pytan.handler.QuestionPoller: ID 1282: query_text resolved to Get C
    2015-08-07 19:38:20,411 DEBUG
                                      pytan.handler.QuestionPoller: ID 1282: id resolved to 1282
    2015-08-07 19:38:20,411 DEBUG
                                     pytan.handler.QuestionPoller: ID 1282: Object Info resolved to Ques
6
    2015-08-07 19:38:20,414 DEBUG
                                     pytan.handler.QuestionPoller: ID 1282: Progress: Tested: 0, Passed:
7
    2015-08-07 19:38:20,414 DEBUG
                                     pytan.handler.QuestionPoller: ID 1282: Timing: Started: 2015-08-07
    2015-08-07 19:38:20,414 INFO
                                     pytan.handler.QuestionPoller: ID 1282: Progress Changed 0% (0 of 2)
   2015-08-07 19:38:25,422 DEBUG
10
                                     pytan.handler.QuestionPoller: ID 1282: Progress: Tested: 2, Passed:
   2015-08-07 19:38:25,422 DEBUG
                                     pytan.handler.QuestionPoller: ID 1282: Timing: Started: 2015-08-07
   2015-08-07 19:38:25,423 INFO
                                     pytan.handler.QuestionPoller: ID 1282: Progress Changed 100% (2 of
12
   2015-08-07 19:38:25,423 INFO
                                     pytan.handler.QuestionPoller: ID 1282: Reached Threshold of 99% (2
13
14
   Type of response: <type 'dict'>
15
16
    Pretty print of response:
17
18
    {'poller_object': <pytan.pollers.QuestionPoller object at 0x10a7ecc10>,
19
     'poller_success': True,
     'question_object': <taniumpy.object_types.question.Question object at 0x10a7ec090>,
20
     'question_results': <taniumpy.object_types.result_set.ResultSet object at 0x10a7ecc90}}
21
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
    CSV Results of response:
26
    Computer Name, Name, Silent Uninstall String, Uninstallable, Version
2.7
   Casus-Belli.local, "Image Capture Extension
28
   Dictation
29
   Wish
   Uninstall AnyConnect
```

```
Time Machine
32
    AppleGraphicsWarning
33
    soagent
34
    Feedback Assistant
35
   AinuIM
    vpndownloader
37
   Pass Viewer
38
   ARDAgent
39
   OBEXAgent
40
   PressAndHold
41
    ..trimmed for brevity..
42
```

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.

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

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

```
Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
   2015-08-07 19:38:25,510 DEBUG
2
                                  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
11
    2015-08-07 19:38:35,526 DEBUG
                                      pytan.handler.QuestionPoller: ID 1283: Progress: Tested: 0, Passed:
12
    2015-08-07 19:38:35,526 DEBUG
                                     pytan.handler.QuestionPoller: ID 1283: Timing: Started: 2015-08-07
13
                                     pytan.handler.QuestionPoller: ID 1283: Progress: Tested: 0, Passed:
   2015-08-07 19:38:40,532 DEBUG
14
    2015-08-07 19:38:40,532 DEBUG
                                     pytan.handler.QuestionPoller: ID 1283: Timing: Started: 2015-08-07
15
    2015-08-07 19:38:45,536 DEBUG
                                     pytan.handler.QuestionPoller: ID 1283: Progress: Tested: 0, Passed:
                                     pytan.handler.QuestionPoller: ID 1283: Timing: Started: 2015-08-07
    2015-08-07 19:38:45,536 DEBUG
17
    2015-08-07 19:38:50,539 DEBUG
                                     pytan.handler.QuestionPoller: ID 1283: Progress: Tested: 0, Passed:
18
    2015-08-07 19:38:50,539 DEBUG
                                     pytan.handler.QuestionPoller: ID 1283: Timing: Started: 2015-08-07
19
    2015-08-07 19:38:55,543 DEBUG
                                     pytan.handler.QuestionPoller: ID 1283: Progress: Tested: 0, Passed:
20
    2015-08-07 19:38:55,543 DEBUG
                                     pytan.handler.QuestionPoller: ID 1283: Timing: Started: 2015-08-07
21
    2015-08-07 19:39:00,547 DEBUG
                                     pytan.handler.QuestionPoller: ID 1283: Progress: Tested: 0, Passed:
22
    2015-08-07 19:39:00,547 DEBUG
                                     pytan.handler.QuestionPoller: ID 1283: Timing: Started: 2015-08-07
23
    2015-08-07 19:39:05,554 DEBUG
                                      pytan.handler.QuestionPoller: ID 1283: Progress: Tested: 0, Passed:
24
    2015-08-07 19:39:05,554 DEBUG
                                     pytan.handler.QuestionPoller: ID 1283: Timing: Started: 2015-08-07
25
    2015-08-07 19:39:10,558 DEBUG
                                     pytan.handler.QuestionPoller: ID 1283: Progress: Tested: 0, Passed:
26
    2015-08-07 19:39:10,558 DEBUG
                                     pytan.handler.QuestionPoller: ID 1283: Timing: Started: 2015-08-07
27
    2015-08-07 19:39:15,561 DEBUG
                                     pytan.handler.QuestionPoller: ID 1283: Progress: Tested: 0, Passed:
28
   2015-08-07 19:39:15,561 DEBUG
                                      pytan.handler.QuestionPoller: ID 1283: Timing: Started: 2015-08-07
29
   2015-08-07 19:39:20,566 DEBUG
                                     pytan.handler.QuestionPoller: ID 1283: Progress: Tested: 0, Passed:
30
   2015-08-07 19:39:20,566 DEBUG
                                     pytan.handler.QuestionPoller: ID 1283: Timing: Started: 2015-08-07
31
   2015-08-07 19:39:25,571 DEBUG
                                     pytan.handler.QuestionPoller: ID 1283: Progress: Tested: 0, Passed:
32
   2015-08-07 19:39:25,571 DEBUG
                                     pytan.handler.QuestionPoller: ID 1283: Timing: Started: 2015-08-07
33
   2015-08-07 19:39:30,577 DEBUG
                                     pytan.handler.QuestionPoller: ID 1283: Progress: Tested: 0, Passed:
34
    2015-08-07 19:39:30,577 DEBUG
                                     pytan.handler.QuestionPoller: ID 1283: Timing: Started: 2015-08-07
35
    2015-08-07 19:39:35,581 DEBUG
                                      pytan.handler.QuestionPoller: ID 1283: Progress: Tested: 0, Passed:
36
    2015-08-07 19:39:35,581 DEBUG
                                      pytan.handler.QuestionPoller: ID 1283: Timing: Started: 2015-08-07
37
    2015-08-07 19:39:40,585 DEBUG
                                     pytan.handler.QuestionPoller: ID 1283: Progress: Tested: 0, Passed:
38
    2015-08-07 19:39:40,585 DEBUG
                                     pytan.handler.QuestionPoller: ID 1283: Timing: Started: 2015-08-07
39
    2015-08-07 19:39:45,588 DEBUG
                                     pytan.handler.QuestionPoller: ID 1283: Progress: Tested: 0, Passed:
40
    2015-08-07 19:39:45,588 DEBUG
                                     pytan.handler.QuestionPoller: ID 1283: Timing: Started: 2015-08-07
41
    2015-08-07 19:39:50,592 DEBUG
                                     pytan.handler.QuestionPoller: ID 1283: Progress: Tested: 0, Passed:
42
                                     pytan.handler.QuestionPoller: ID 1283: Timing: Started: 2015-08-07
    2015-08-07 19:39:50,592 DEBUG
43
    2015-08-07 19:39:55,597 DEBUG
                                     pytan.handler.QuestionPoller: ID 1283: Progress: Tested: 0, Passed:
44
   2015-08-07 19:39:55,597 DEBUG
                                     pytan.handler.QuestionPoller: ID 1283: Timing: Started: 2015-08-07
45
    2015-08-07 19:40:00,603 DEBUG
                                     pytan.handler.QuestionPoller: ID 1283: Progress: Tested: 0, Passed:
46
    2015-08-07 19:40:00,603 DEBUG
                                     pytan.handler.QuestionPoller: ID 1283: Timing: Started: 2015-08-07
47
    2015-08-07 19:40:05,612 DEBUG
                                     pytan.handler.QuestionPoller: ID 1283: Progress: Tested: 0, Passed:
48
    2015-08-07 19:40:05,613 DEBUG
                                     pytan.handler.QuestionPoller: ID 1283: Timing: Started: 2015-08-07
49
    2015-08-07 19:40:10,618 DEBUG
                                      pytan.handler.QuestionPoller: ID 1283: Progress: Tested: 1, Passed:
50
    2015-08-07 19:40:10,618 DEBUG
                                      pytan.handler.QuestionPoller: ID 1283: Timing: Started: 2015-08-07
51
    2015-08-07 19:40:10,618 INFO
                                     pytan.handler.QuestionPoller: ID 1283: Progress Changed 50% (1 of 2
52
    2015-08-07 19:40:15,626 DEBUG
                                     pytan.handler.QuestionPoller: ID 1283: Progress: Tested: 1, Passed:
53
                                     pytan.handler.QuestionPoller: ID 1283: Timing: Started: 2015-08-07
    2015-08-07 19:40:15,626 DEBUG
54
   2015-08-07 19:40:20,631 DEBUG
                                     pytan.handler.QuestionPoller: ID 1283: Progress: Tested: 1, Passed:
55
                                     pytan.handler.QuestionPoller: ID 1283: Timing: Started: 2015-08-07
   2015-08-07 19:40:20,631 DEBUG
56
   2015-08-07 19:40:25,635 DEBUG
                                     pytan.handler.QuestionPoller: ID 1283: Progress: Tested: 1, Passed:
57
   2015-08-07 19:40:25,635 DEBUG
                                     pytan.handler.QuestionPoller: ID 1283: Timing: Started: 2015-08-07
58
   2015-08-07 19:40:30,639 DEBUG
                                     pytan.handler.QuestionPoller: ID 1283: Progress: Tested: 2, Passed:
59
   2015-08-07 19:40:30,639 DEBUG
                                     pytan.handler.QuestionPoller: ID 1283: Timing: Started: 2015-08-07
60
    2015-08-07 19:40:30,639 INFO
                                     pytan.handler.QuestionPoller: ID 1283: Progress Changed 100% (2 of
61
    2015-08-07 19:40:30,639 INFO
62
                                     pytan.handler.QuestionPoller: ID 1283: Reached Threshold of 99% (2
63
    Type of response: <type 'dict'>
64
65
    Pretty print of response:
66
    {'poller_object': <pytan.pollers.QuestionPoller object at 0x10a613090>,
67
    'poller_success': True,
```

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

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

```
USERNAME = "Tanium User"
21
    PASSWORD = "T@n!um"
22
    HOST = "172.16.31.128"
23
    PORT = "443"
24
25
    # Logging conrols
26
    LOGLEVEL = 2
27
    DEBUGFORMAT = False
28
29
    import tempfile
30
31
    import pytan
32
33
    handler = pytan.Handler(
        username=USERNAME,
34
        password=PASSWORD,
35
        host=HOST,
36
        port=PORT,
37
        loglevel=LOGLEVEL,
38
        debugformat=DEBUGFORMAT,
39
40
41
    print handler
42
43
    # setup the arguments for the handler method
44
    kwargs = {}
45
    kwargs["sensors"] = [u'Folder Name Search with RegEx Match{dirname=Program Files,regex=Microsoft.*}"
46
     u'Computer Name']
47
    kwargs["qtype"] = u'manual'
48
49
    # call the handler with the ask method, passing in kwargs for arguments
50
    response = handler.ask(**kwargs)
51
    import pprint, io
52
53
    print ""
54
    print "Type of response: ", type(response)
55
56
    print ""
57
    print "Pretty print of response:"
    print pprint.pformat(response)
59
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
```

```
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
2
    2015-08-07 19:40:30,754 DEBUG
                                     pytan.handler.QuestionPoller: ID 1284: expiration resolved to 2015-
3
    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:
10
    2015-08-07 19:40:35,761 DEBUG
                                     pytan.handler.QuestionPoller: ID 1284: Timing: Started: 2015-08-07
11
   2015-08-07 19:40:40,766 DEBUG
                                     pytan.handler.QuestionPoller: ID 1284: Progress: Tested: 0, Passed:
12
    2015-08-07 19:40:40,766 DEBUG
                                      pytan.handler.QuestionPoller: ID 1284: Timing: Started: 2015-08-07
13
    2015-08-07 19:40:45,773 DEBUG
                                     pytan.handler.QuestionPoller: ID 1284: Progress: Tested: 0, Passed:
14
    2015-08-07 19:40:45,773 DEBUG
                                     pytan.handler.QuestionPoller: ID 1284: Timing: Started: 2015-08-07
15
    2015-08-07 19:40:50,777 DEBUG
                                     pytan.handler.QuestionPoller: ID 1284: Progress: Tested: 0, Passed:
16
                                     pytan.handler.QuestionPoller: ID 1284: Timing: Started: 2015-08-07
    2015-08-07 19:40:50,777 DEBUG
17
    2015-08-07 19:40:55,782 DEBUG
                                     pytan.handler.QuestionPoller: ID 1284: Progress: Tested: 0, Passed:
18
    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:
20
    2015-08-07 19:41:00,790 DEBUG
                                     pytan.handler.QuestionPoller: ID 1284: Timing: Started: 2015-08-07
21
   2015-08-07 19:41:05,798 DEBUG
                                     pytan.handler.QuestionPoller: ID 1284: Progress: Tested: 0, Passed:
22
    2015-08-07 19:41:05,798 DEBUG
                                     pytan.handler.QuestionPoller: ID 1284: Timing: Started: 2015-08-07
23
    2015-08-07 19:41:10,805 DEBUG
                                     pytan.handler.QuestionPoller: ID 1284: Progress: Tested: 0, Passed:
24
    2015-08-07 19:41:10,805 DEBUG
                                     pytan.handler.QuestionPoller: ID 1284: Timing: Started: 2015-08-07
25
    2015-08-07 19:41:15,809 DEBUG
                                     pytan.handler.QuestionPoller: ID 1284: Progress: Tested: 0, Passed:
26
    2015-08-07 19:41:15,809 DEBUG
                                      pytan.handler.QuestionPoller: ID 1284: Timing: Started: 2015-08-07
27
    2015-08-07 19:41:20,813 DEBUG
                                      pytan.handler.QuestionPoller: ID 1284: Progress: Tested: 0, Passed:
28
    2015-08-07 19:41:20,813 DEBUG
                                     pytan.handler.QuestionPoller: ID 1284: Timing: Started: 2015-08-07
29
    2015-08-07 19:41:25,817 DEBUG
                                     pytan.handler.QuestionPoller: ID 1284: Progress: Tested: 2, Passed:
30
31
    2015-08-07 19:41:25,817 DEBUG
                                     pytan.handler.QuestionPoller: ID 1284: Timing: Started: 2015-08-07
                                     pytan.handler.QuestionPoller: ID 1284: Progress Changed 100% (2 of
32
   2015-08-07 19:41:25,817 INFO
   2015-08-07 19:41:25,817 INFO
                                      pytan.handler.QuestionPoller: ID 1284: Reached Threshold of 99% (2
33
34
35
   Type of response: <type 'dict'>
36
    Pretty print of response:
37
    {'poller_object': <pytan.pollers.QuestionPoller object at 0x10a614b50>,
38
     'poller_success': True,
     'question_object': <taniumpy.object_types.question.Question object at 0x10a5f51d0>,
     'question_results': <taniumpy.object_types.result_set.ResultSet object at 0x10a614f10}
41
42.
    Equivalent Question if it were to be asked in the Tanium Console:
43
44
    Get Folder Name Search with RegEx Match[Program Files, , No, No, Microsoft.*] and Computer Name from
45
    CSV Results of response:
    Computer Name, "Folder Name Search with RegEx Match[Program Files, , No, No, Microsoft. | ]"
47
48
    Casus-Belli.local, Windows Only
    JTANIUM1.localdomain, "C:\Program Files\VMware\VMware Tools\plugins\vmsvc
49
   C:\Program Files\Common Files\Microsoft Shared\VS7Debug
50
51
   C:\Program Files\Tanium\Tanium Server\http\taniumjs\sensor-query\src
   C:\Program Files\Microsoft SQL Server\110\LocalDB\Binn\Resources\1033
52
   C:\Program Files\Tanium\Tanium Server\http\tux\spin\src
```

```
C:\Program Files\Tanium\Tanium Server\http\taniumjs\archived-question\src
54
   C:\Program Files\Tanium\Tanium Module Server\plugins\content
55
   C:\Program Files\Tanium\Tanium Server\http\libraries\kendoui\styles\Moonlight
56
   C:\Program Files\Common Files\VMware\Drivers\vmci\sockets\include
57
   C:\Program Files\Tanium\Tanium Server\http\taniumjs\plugin
58
   C:\Program Files\Common Files\Microsoft Shared\ink\ar-SA
   C:\Program Files\Tanium\Tanium Server\plugins\console\WorkbenchesManager
60
   C:\Program Files\Tanium\Tanium Module Server\logs
61
   ..trimmed for brevity...
62
```

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.

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

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

```
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
2
   2015-08-07 19:41:25,887 DEBUG
                                     pytan.handler.QuestionPoller: ID 1286: expiration resolved to 2015-
3
   2015-08-07 19:41:25,887 DEBUG
                                     pytan.handler.QuestionPoller: ID 1286: query_text resolved to Get C
4
   2015-08-07 19:41:25,887 DEBUG
                                     pytan.handler.QuestionPoller: ID 1286: id resolved to 1286
5
   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:
10
   2015-08-07 19:41:30,900 DEBUG
                                     pytan.handler.QuestionPoller: ID 1286: Timing: Started: 2015-08-07
11
   2015-08-07 19:41:35,905 DEBUG
                                     pytan.handler.QuestionPoller: ID 1286: Progress: Tested: 0, Passed:
12
   2015-08-07 19:41:35,905 DEBUG
                                     pytan.handler.QuestionPoller: ID 1286: Timing: Started: 2015-08-07
13
   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
15
    2015-08-07 19:41:45,915 DEBUG
                                      pytan.handler.QuestionPoller: ID 1286: Progress: Tested: 0, Passed:
16
    2015-08-07 19:41:45,915 DEBUG
                                     pytan.handler.QuestionPoller: ID 1286: Timing: Started: 2015-08-07
17
                                      pytan.handler.QuestionPoller: ID 1286: Progress: Tested: 0, Passed:
   2015-08-07 19:41:50,919 DEBUG
18
   2015-08-07 19:41:50,919 DEBUG
                                      pytan.handler.QuestionPoller: ID 1286: Timing: Started: 2015-08-07
19
   2015-08-07 19:41:55,923 DEBUG
                                      pytan.handler.QuestionPoller: ID 1286: Progress: Tested: 1, Passed:
20
                                     pytan.handler.QuestionPoller: ID 1286: Timing: Started: 2015-08-07
   2015-08-07 19:41:55,924 DEBUG
21
   2015-08-07 19:41:55,924 INFO
                                     pytan.handler.QuestionPoller: ID 1286: Progress Changed 50% (1 of 2
22
   2015-08-07 19:42:00,928 DEBUG
                                     pytan.handler.QuestionPoller: ID 1286: Progress: Tested: 1, Passed:
23
   2015-08-07 19:42:00,929 DEBUG
                                     pytan.handler.QuestionPoller: ID 1286: Timing: Started: 2015-08-07
24
                                      pytan.handler.QuestionPoller: ID 1286: Progress: Tested: 2, Passed:
   2015-08-07 19:42:05,933 DEBUG
25
    2015-08-07 19:42:05,933 DEBUG
                                      pytan.handler.QuestionPoller: ID 1286: Timing: Started: 2015-08-07
26
27
    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
28
29
    Type of response: <type 'dict'>
30
31
    Pretty print of response:
32
    {'poller_object': <pytan.pollers.QuestionPoller object at 0x10a614e10>,
33
     'poller_success': True,
34
     'question_object': <taniumpy.object_types.question.Question object at 0x10a613450>,
35
     'question_results': <taniumpy.object_types.result_set.ResultSet object at 0x10a614490$
36
37
    Equivalent Question if it were to be asked in the Tanium Console:
38
    Get Computer Name[Dweedle] from all machines
39
40
    CSV Results of response:
41
    Computer Name[Dweedle]
42
    [no results]
43
    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.

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

```
for aa in path_adds:
15
        if aa not in sys.path:
16
             sys.path.append(aa)
17
18
19
    # connection info for Tanium Server
20
    USERNAME = "Tanium User"
21
    PASSWORD = "T@n!um"
22
    HOST = "172.16.31.128"
23
    PORT = "443"
24
25
    # Logging conrols
26
    LOGLEVEL = 2
27
    DEBUGFORMAT = False
28
29
    import tempfile
30
31
    import pytan
32
33
    handler = pytan.Handler(
        username=USERNAME,
34
        password=PASSWORD,
35
        host=HOST,
36
        port=PORT,
37
        loglevel=LOGLEVEL,
38
        debugformat=DEBUGFORMAT,
39
40
41
    print handler
42
43
    # setup the arguments for the handler method
44
    kwargs = {}
45
    kwargs["sensors"] = u'Folder Name Search with RegEx Match'
46
    kwargs["qtype"] = u'manual'
47
48
    # call the handler with the ask method, passing in kwargs for arguments
49
    response = handler.ask(**kwargs)
50
    import pprint, io
51
52
    print ""
53
    print "Type of response: ", type(response)
54
55
    print ""
56
    print "Pretty print of response:"
57
    print pprint.pformat(response)
58
59
    print ""
    print "Equivalent Question if it were to be asked in the Tanium Console: "
61
    print response['question_object'].query_text
62
63
    # create an IO stream to store CSV results to
64
    out = io.BytesIO()
65
66
    # call the write_csv() method to convert response to CSV and store it in out
67
    response['question_results'].write_csv(out, response['question_results'])
68
69
    print ""
70
    print "CSV Results of response: "
71
   out = out.getvalue()
```

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

```
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
2
   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
6
    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:
10
   2015-08-07 19:42:11,021 DEBUG
                                     pytan.handler.QuestionPoller: ID 1288: Timing: Started: 2015-08-07
11
    2015-08-07 19:42:16,025 DEBUG
                                     pytan.handler.QuestionPoller: ID 1288: Progress: Tested: 0, Passed:
12
    2015-08-07 19:42:16,025 DEBUG
                                     pytan.handler.QuestionPoller: ID 1288: Timing: Started: 2015-08-07
13
    2015-08-07 19:42:21,032 DEBUG
                                     pytan.handler.QuestionPoller: ID 1288: Progress: Tested: 0, Passed:
14
   2015-08-07 19:42:21,032 DEBUG
                                     pytan.handler.QuestionPoller: ID 1288: Timing: Started: 2015-08-07
15
   2015-08-07 19:42:26,037 DEBUG
                                     pytan.handler.QuestionPoller: ID 1288: Progress: Tested: 2, Passed:
16
    2015-08-07 19:42:26,037 DEBUG
                                     pytan.handler.QuestionPoller: ID 1288: Timing: Started: 2015-08-07
17
    2015-08-07 19:42:26,038 INFO
                                     pytan.handler.QuestionPoller: ID 1288: Progress Changed 100% (2 of
18
19
    2015-08-07 19:42:26,038 INFO
                                     pytan.handler.QuestionPoller: ID 1288: Reached Threshold of 99% (2
20
    Type of response: <type 'dict'>
21
22
    Pretty print of response:
23
    {'poller_object': <pytan.pollers.QuestionPoller object at 0x10a6147d0>,
24
     'poller_success': True,
25
     'question_object': <taniumpy.object_types.question.Question object at 0x10a613790>,
26
     'question_results': <taniumpy.object_types.result_set.ResultSet object at 0x10a5f5190}
27
28
    Equivalent Question if it were to be asked in the Tanium Console:
29
   Get Folder Name Search with RegEx Match[, , No, No] from all machines
30
31
32
    CSV Results of response:
    Count, "Folder Name Search with RegEx Match[, , No, No]"
33
    24705, [too many results]
34
    1,C:\Windows\winsxs\amd64_microsoft-windows-s..structure.resources_31bf3856ad364e35_6.1.7600.16385_e
35
    1,C:\Windows\winsxs\x86_microsoft-windows-e..-host-authenticator_31bf3856ad364e35_6.1.7601.17514_nor
36
    1,C:\Windows\winsxs\amd64_microsoft-windows-ocspsvc_31bf3856ad364e35_6.1.7601.22807_nohe_3bfeae72930
37
   1,C:\Windows\winsxs\amd64_microsoft-windows-c..ityclient.resources_31bf3856ad364e35_6.1.7601.22865_6
38
   1, C:\Windows\assembly\NativeImages_v2.0.50727_64\System.Xml
39
   1,C:\Windows\winsxs\amd64_microsoft-windows-winsetupui_31bf3856ad364e35_6.1.7601.18804_none_bd3cf1bb
40
    1,C:\Windows\winsxs\amd64_microsoft-windows-scripting.resources_31bf3856ad364e35_6.1.7600.16385_en-u
41
    1,C:\Windows\winsxs\x86_microsoft-windows-mlang.resources_31bf3856ad364e35_6.1.7600.16$85_ru-ru_cf3a
42.
    1,C:\Windows\winsxs\x86_microsoft-windows-minkernelapinamespace_31bf3856ad364e35_6.1.7601.21728_none
43
    1,C:\Users\Jim Olsen\AppData\Local\Google
44
    1,C:\Windows\winsxs\x86_microsoft-windows-e..nt-client.resources_31bf3856ad364e35_6.1.7600.16385_en-
45
    1,C:\Windows\winsxs\amd64_microsoft-windows-d..e-eashared-kjshared_31bf3856ad364e35_6.1.7600.16385_r
46
   1, C:\Windows\assembly\NativeImages_v4.0.30319_32\RadLangSvc
47
    ..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
2
    sys.dont_write_bytecode = True
    # Determine our script name, script dir
5
   my_file = os.path.abspath(sys.argv[0])
6
   my_dir = os.path.dirname(my_file)
7
    # determine the pytan lib dir and add it to the path
   parent_dir = os.path.dirname(my_dir)
10
11
   pytan_root_dir = os.path.dirname(parent_dir)
    lib_dir = os.path.join(pytan_root_dir, 'lib')
12
   path_adds = [lib_dir]
13
14
    for aa in path_adds:
15
        if aa not in sys.path:
16
            sys.path.append(aa)
17
18
19
    # connection info for Tanium Server
20
    USERNAME = "Tanium User"
21
    PASSWORD = "T@n!um"
22
    HOST = "172.16.31.128"
23
    PORT = "443"
24
25
    # Logging conrols
26
    LOGLEVEL = 2
27
    DEBUGFORMAT = False
28
29
    import tempfile
30
31
    import pytan
32
    handler = pytan.Handler(
33
        username=USERNAME,
34
        password=PASSWORD,
35
        host=HOST,
36
        port=PORT,
37
        loglevel=LOGLEVEL,
38
        debugformat=DEBUGFORMAT,
39
40
41
   print handler
42
43
    # setup the arguments for the handler method
44
    kwarqs = {}
45
    kwargs["sensors"] = u'Folder Name Search with RegEx Match{dirname=Program Files,regex=Microsoft.*},
46
   kwargs["qtype"] = u'manual'
```

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

```
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
   2015-08-07 19:42:26,175 DEBUG
                                     pytan.handler.QuestionPoller: ID 1289: expiration resolved to 2015-
   2015-08-07 19:42:26,175 DEBUG
                                     pytan.handler.QuestionPoller: ID 1289: query_text resolved to Get F
   2015-08-07 19:42:26,175 DEBUG
                                     pytan.handler.QuestionPoller: ID 1289: id resolved to 1289
   2015-08-07 19:42:26,175 DEBUG
                                     pytan.handler.QuestionPoller: ID 1289: Object Info resolved to Ques
   2015-08-07 19:42:26,178 DEBUG
                                     pytan.handler.QuestionPoller: ID 1289: Progress: Tested: 0, Passed:
   2015-08-07 19:42:26,178 DEBUG
                                     pytan.handler.QuestionPoller: ID 1289: Timing: Started: 2015-08-07
   2015-08-07 19:42:26,178 INFO
                                     pytan.handler.QuestionPoller: ID 1289: Progress Changed 0% (0 of 2)
    2015-08-07 19:42:31,183 DEBUG
                                     pytan.handler.QuestionPoller: ID 1289: Progress: Tested: 0, Passed:
10
11
    2015-08-07 19:42:31,183 DEBUG
                                     pytan.handler.QuestionPoller: ID 1289: Timing: Started: 2015-08-07
    2015-08-07 19:42:36,189 DEBUG
                                     pytan.handler.QuestionPoller: ID 1289: Progress: Tested: 1, Passed:
12
    2015-08-07 19:42:36,189 DEBUG
                                     pytan.handler.QuestionPoller: ID 1289: Timing: Started: 2015-08-07
13
    2015-08-07 19:42:36,189 INFO
                                     pytan.handler.QuestionPoller: ID 1289: Progress Changed 50% (1 of 2
14
15
    2015-08-07 19:42:41,193 DEBUG
                                     pytan.handler.QuestionPoller: ID 1289: Progress: Tested: 1, Passed:
   2015-08-07 19:42:41,194 DEBUG
                                     pytan.handler.QuestionPoller: ID 1289: Timing: Started: 2015-08-07
16
                                     pytan.handler.QuestionPoller: ID 1289: Progress: Tested: 1, Passed:
17
   2015-08-07 19:42:46,197 DEBUG
   2015-08-07 19:42:46,197 DEBUG
                                     pytan.handler.QuestionPoller: ID 1289: Timing: Started: 2015-08-07
18
   2015-08-07 19:42:51,202 DEBUG
                                     pytan.handler.QuestionPoller: ID 1289: Progress: Tested: 1, Passed:
19
   2015-08-07 19:42:51,202 DEBUG
                                     pytan.handler.OuestionPoller: ID 1289: Timing: Started: 2015-08-07
20
   2015-08-07 19:42:56,207 DEBUG
                                     pytan.handler.QuestionPoller: ID 1289: Progress: Tested: 2, Passed:
21
                                     pytan.handler.QuestionPoller: ID 1289: Timing: Started: 2015-08-07
   2015-08-07 19:42:56,207 DEBUG
22
   2015-08-07 19:42:56,207 INFO
                                     pytan.handler.QuestionPoller: ID 1289: Progress Changed 100% (2 of
23
   2015-08-07 19:42:56,207 INFO
                                     pytan.handler.QuestionPoller: ID 1289: Reached Threshold of 99% (2
```

```
25
   Type of response: <type 'dict'>
26
27
   Pretty print of response:
28
   {'poller_object': <pytan.pollers.QuestionPoller object at 0x10a614f50>,
29
     'poller_success': True,
     'question_object': <taniumpy.object_types.question.Question object at 0x10a5f5190>,
31
     'question_results': <taniumpy.object_types.result_set.ResultSet object at 0x10a615c10♭}
32
33
   Equivalent Question if it were to be asked in the Tanium Console:
34
   Get Folder Name Search with RegEx Match[Program Files, , No, No, Microsoft.*] containing "Shared" fr
35
   CSV Results of response:
37
    "Folder Name Search with RegEx Match[Program Files, , No, No, Microsoft.*]"
38
   [no results]
39
   C:\Program Files\Common Files\Microsoft Shared\VS7Debug
40
   C:\Program Files\Common Files\Microsoft Shared\ink\ar-SA
41
   C:\Program Files\Common Files\Microsoft Shared\ink\ru-RU
42
   C:\Program Files\Common Files\Microsoft Shared\ink\fsdefinitions\keypad
43
   C:\Program Files\Common Files\Microsoft Shared\ink
44
   C:\Program Files\Common Files\Microsoft Shared\ink\sv-SE
45
   C:\Program Files\Common Files\Microsoft Shared\ink\uk-UA
46
   C:\Program Files\Common Files\Microsoft Shared\ink\sl-SI
47
   C:\Program Files\Common Files\Microsoft Shared\ink\hu-HU
48
   C:\Program Files\Common Files\Microsoft Shared\ink\zh-TW
49
   C:\Program Files\Common Files\Microsoft Shared\ink\zh-CN
50
   C:\Program Files\Common Files\Microsoft Shared\ink\fi-FI
51
   C:\Program Files\Common Files\Microsoft Shared
52
   ..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

```
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]
13
14
    for aa in path_adds:
15
        if aa not in sys.path:
16
            sys.path.append(aa)
17
18
19
    # connection info for Tanium Server
20
    USERNAME = "Tanium User"
21
    PASSWORD = "T@n!um"
22
    HOST = "172.16.31.128"
23
    PORT = "443"
24
25
    # Logging conrols
26
    LOGLEVEL = 2
27
    DEBUGFORMAT = False
28
29
    import tempfile
30
31
    import pytan
32
    handler = pytan.Handler(
33
        username=USERNAME,
34
        password=PASSWORD,
35
        host=HOST,
36
        port=PORT,
37
        loglevel=LOGLEVEL,
38
        debugformat=DEBUGFORMAT,
39
40
41
    print handler
42
43
    # setup the arguments for the handler method
44
    kwargs = \{\}
45
    kwargs["sensors"] = u'Operating System, that contains:Windows, opt:max_data_age:3600, opt:value_type
46
    kwargs["qtype"] = u'manual'
47
48
    # call the handler with the ask method, passing in kwargs for arguments
49
    response = handler.ask(**kwargs)
50
51
    import pprint, io
52
53
    print "Type of response: ", type(response)
54
55
   print ""
56
    print "Pretty print of response:"
57
    print pprint.pformat(response)
58
   print ""
60
    print "Equivalent Question if it were to be asked in the Tanium Console: "
61
    print response['question_object'].query_text
62
63
    # create an IO stream to store CSV results to
    out = io.BytesIO()
65
66
    # call the write_csv() method to convert response to CSV and store it in out
67
    response['question_results'].write_csv(out, response['question_results'])
68
69
   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
```

```
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
2
   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 (
   2015-08-07 19:42:56,259 DEBUG
                                     pytan.handler.QuestionPoller: ID 1290: id resolved to 1290
5
   2015-08-07 19:42:56,259 DEBUG
                                     pytan.handler.QuestionPoller: ID 1290: Object Info resolved to Ques
6
   2015-08-07 19:42:56,262 DEBUG
                                     pytan.handler.QuestionPoller: ID 1290: Progress: Tested: 0, Passed:
7
   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:
10
   2015-08-07 19:43:01,266 DEBUG
                                     pytan.handler.QuestionPoller: ID 1290: Timing: Started: 2015-08-07
11
   2015-08-07 19:43:01,266 INFO
                                     pytan.handler.QuestionPoller: ID 1290: Progress Changed 50% (1 of 2
12
   2015-08-07 19:43:06,271 DEBUG
                                     pytan.handler.QuestionPoller: ID 1290: Progress: Tested: 2, Passed:
13
   2015-08-07 19:43:06,271 DEBUG
                                     pytan.handler.QuestionPoller: ID 1290: Timing: Started: 2015-08-07
14
   2015-08-07 19:43:06,271 INFO
                                     pytan.handler.QuestionPoller: ID 1290: Progress Changed 100% (2 of
15
   2015-08-07 19:43:06,271 INFO
                                     pytan.handler.QuestionPoller: ID 1290: Reached Threshold of 99% (2
16
17
   Type of response: <type 'dict'>
18
19
   Pretty print of response:
20
   {'poller_object': <pytan.pollers.QuestionPoller object at 0x10a5f5f90>,
21
     'poller_success': True,
22
     'question_object': <taniumpy.object_types.question.Question object at 0x10a613b10>,
23
     'question_results': <taniumpy.object_types.result_set.ResultSet object at 0x10a5b9cd0}}
24
25
   Equivalent Question if it were to be asked in the Tanium Console:
26
   Get Operating System containing "Windows" from all machines
27
28
   CSV Results of response:
29
30
   Operating System
    [no results]
31
   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
2
    sys.dont_write_bytecode = True
    # Determine our script name, script dir
5
    my_file = os.path.abspath(sys.argv[0])
6
    my_dir = os.path.dirname(my_file)
7
    # determine the pytan lib dir and add it to the path
10
    parent_dir = os.path.dirname(my_dir)
   pytan_root_dir = os.path.dirname(parent_dir)
    lib_dir = os.path.join(pytan_root_dir, 'lib')
12
   path_adds = [lib_dir]
13
14
   for aa in path_adds:
15
        if aa not in sys.path:
16
            sys.path.append(aa)
17
18
19
    # connection info for Tanium Server
20
    USERNAME = "Tanium User"
21
    PASSWORD = "T@n!um"
22
    HOST = "172.16.31.128"
23
    PORT = "443"
24
25
    # Logging conrols
26
    LOGLEVEL = 2
27
   DEBUGFORMAT = False
28
29
    import tempfile
30
31
    import pytan
32
    handler = pytan.Handler(
33
        username=USERNAME,
34
        password=PASSWORD,
35
36
        host=HOST,
37
        port=PORT,
        loglevel=LOGLEVEL,
        debugformat=DEBUGFORMAT,
39
40
41
   print handler
42
43
    # setup the arguments for the handler method
45
    kwarqs["sensors"] = u'Operating System, that contains: Windows'
46
    kwargs["qtype"] = u'manual'
47
48
49
    # call the handler with the ask method, passing in kwargs for arguments
50
    response = handler.ask(**kwargs)
    import pprint, io
52
53
   print "Type of response: ", type(response)
54
55
   print ""
56
   print "Pretty print of response:"
   print pprint.pformat(response)
```

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

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

```
import os
1
    import sys
2
    sys.dont_write_bytecode = True
3
    # 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
9
    parent_dir = os.path.dirname(my_dir)
10
    pytan_root_dir = os.path.dirname(parent_dir)
11
    lib_dir = os.path.join(pytan_root_dir, 'lib')
12
    path_adds = [lib_dir]
13
14
    for aa in path_adds:
15
        if aa not in sys.path:
16
            sys.path.append(aa)
17
18
19
    # connection info for Tanium Server
20
    USERNAME = "Tanium User"
21
    PASSWORD = "T@n!um"
22
    HOST = "172.16.31.128"
23
    PORT = "443"
24
25
    # Logging conrols
26
27
    LOGLEVEL = 2
    DEBUGFORMAT = False
28
29
    import tempfile
30
31
    import pytan
32
    handler = pytan.Handler(
33
        username=USERNAME,
34
        password=PASSWORD,
35
        host=HOST,
36
        port=PORT,
37
        loglevel=LOGLEVEL,
38
        debugformat=DEBUGFORMAT,
39
40
41
   print handler
42
43
    # setup the arguments for the handler method
44
45
    kwargs = {}
   kwargs["sensors"] = u'Folder Name Search with RegEx Match{dirname=Program Files,regex=Nicrosoft.*},
```

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

```
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
2
   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)
10
   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
11
   2015-08-07 19:43:26,420 DEBUG
                                     pytan.handler.QuestionPoller: ID 1294: Progress: Tested: 0, Passed:
12
   2015-08-07 19:43:26,420 DEBUG
                                     pytan.handler.QuestionPoller: ID 1294: Timing: Started: 2015-08-07
13
   2015-08-07 19:43:31,424 DEBUG
                                     pytan.handler.QuestionPoller: ID 1294: Progress: Tested: 0, Passed:
14
15
   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
17
   2015-08-07 19:43:36,428 INFO
                                     pytan.handler.QuestionPoller: ID 1294: Progress Changed 50% (1 of 2
18
   2015-08-07 19:43:41,432 DEBUG
                                     pytan.handler.QuestionPoller: ID 1294: Progress: Tested: 1, Passed:
19
   2015-08-07 19:43:41,432 DEBUG
                                     pytan.handler.QuestionPoller: ID 1294: Timing: Started: 2015-08-07
20
   2015-08-07 19:43:46,441 DEBUG
                                     pytan.handler.QuestionPoller: ID 1294: Progress: Tested: 1, Passed:
21
   2015-08-07 19:43:46,442 DEBUG
                                     pytan.handler.QuestionPoller: ID 1294: Timing: Started: 2015-08-07
22
   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
24
    2015-08-07 19:43:51,449 INFO
                                      pytan.handler.QuestionPoller: ID 1294: Progress Changed 100% (2 of
25
    2015-08-07 19:43:51,449 INFO
                                      pytan.handler.QuestionPoller: ID 1294: Reached Threshold of 99% (2
26
27
    Type of response: <type 'dict'>
28
29
    Pretty print of response:
30
    {'poller_object': <pytan.pollers.QuestionPoller object at 0x10a615f10>,
31
     'poller_success': True,
32
     'question_object': <taniumpy.object_types.question.Question object at 0x10a5f57d0>,
33
     'question_results': <taniumpy.object_types.result_set.ResultSet object at 0x10a5b9a10♭}
34
    Equivalent Question if it were to be asked in the Tanium Console:
36
    Get Folder Name Search with RegEx Match[Program Files, , No, No, Microsoft.*] containing "Shared" fr
37
38
    CSV Results of response:
39
    "Folder Name Search with RegEx Match[Program Files, , No, No, Microsoft.*]"
40
    [no results]
41
   C:\Program Files\Common Files\Microsoft Shared\VS7Debug
42
   C:\Program Files\Common Files\Microsoft Shared\ink\ar-SA
43
   C:\Program Files\Common Files\Microsoft Shared\ink\ru-RU
44
   C:\Program Files\Common Files\Microsoft Shared\ink\fsdefinitions\keypad
45
   C:\Program Files\Common Files\Microsoft Shared\ink
46
   C:\Program Files\Common Files\Microsoft Shared\ink\sv-SE
47
   C:\Program Files\Common Files\Microsoft Shared\ink\uk-UA
48
49
    C:\Program Files\Common Files\Microsoft Shared\ink\sl-SI
    C:\Program Files\Common Files\Microsoft Shared\ink\hu-HU
50
    C:\Program Files\Common Files\Microsoft Shared\ink\zh-TW
51
   C:\Program Files\Common Files\Microsoft Shared\ink\zh-CN
52
   C:\Program Files\Common Files\Microsoft Shared\ink\fi-FI
53
   C:\Program Files\Common Files\Microsoft Shared
54
   ..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.

```
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)
10
    pytan root dir = os.path.dirname(parent dir)
11
    lib_dir = os.path.join(pytan_root_dir, 'lib')
12
    path_adds = [lib_dir]
13
14
    for aa in path_adds:
15
        if aa not in sys.path:
16
             sys.path.append(aa)
17
18
19
    # connection info for Tanium Server
20
    USERNAME = "Tanium User"
21
    PASSWORD = "T@n!um"
22
    HOST = "172.16.31.128"
23
    PORT = "443"
24
25
    # Logging conrols
26
    LOGLEVEL = 2
27
    DEBUGFORMAT = False
28
29
    import tempfile
30
31
    import pytan
32
    handler = pytan.Handler(
33
34
        username=USERNAME,
        password=PASSWORD,
35
        host=HOST,
36
        port=PORT,
37
        loglevel=LOGLEVEL,
38
        debugformat=DEBUGFORMAT,
39
40
41
    print handler
42
43
    # setup the arguments for the handler method
44
    kwargs = {}
45
    kwargs["sensors"] = u'Operating System, that contains:Windows, opt:match_all_values, opt:ignore_case
46
    kwargs["qtype"] = u'manual'
47
48
    # call the handler with the ask method, passing in kwargs for arguments
49
    response = handler.ask(**kwargs)
50
    import pprint, io
51
52
    print ""
53
    print "Type of response: ", type(response)
54
55
    print ""
56
    print "Pretty print of response:"
57
    print pprint.pformat(response)
58
59
    print ""
60
    print "Equivalent Question if it were to be asked in the Tanium Console: "
61
    print response['question_object'].query_text
62
63
    # create an IO stream to store CSV results to
64
    out = io.BvtesIO()
65
66
    # call the write_csv() method to convert response to CSV and store it in out
```

```
response['question_results'].write_csv(out, response['question_results'])
68
69
   print ""
70
   print "CSV Results of response: "
71
   out = out.getvalue()
72
   if len(out.splitlines()) > 15:
73
        out = out.splitlines()[0:15]
74
        out.append('..trimmed for brevity..')
75
        out = ' \ n'. join (out)
76
   print out
```

```
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
2
   2015-08-07 19:43:51,504 DEBUG
                                     pytan.handler.QuestionPoller: ID 1295: expiration resolved to 2015-
3
   2015-08-07 19:43:51,504 DEBUG
                                     pytan.handler.QuestionPoller: ID 1295: query_text resolved to Get C
4
   2015-08-07 19:43:51,504 DEBUG
                                     pytan.handler.QuestionPoller: ID 1295: id resolved to 1295
5
   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:
10
   2015-08-07 19:43:56,512 DEBUG
                                     pytan.handler.QuestionPoller: ID 1295: Timing: Started: 2015-08-07
11
   2015-08-07 19:44:01,520 DEBUG
                                     pytan.handler.QuestionPoller: ID 1295: Progress: Tested: 2, Passed:
12
   2015-08-07 19:44:01,520 DEBUG
                                     pytan.handler.QuestionPoller: ID 1295: Timing: Started: 2015-08-07
13
   2015-08-07 19:44:01,520 INFO
14
                                     pytan.handler.QuestionPoller: ID 1295: Progress Changed 100% (2 of
                                     pytan.handler.QuestionPoller: ID 1295: Reached Threshold of 99% (2
   2015-08-07 19:44:01,520 INFO
15
16
   Type of response: <type 'dict'>
17
18
   Pretty print of response:
19
   {'poller_object': <pytan.pollers.QuestionPoller object at 0x10a5b98d0>,
20
     'poller_success': True,
21
     'question_object': <taniumpy.object_types.question.Question object at 0x10a5b9810>,
22
     'question_results': <taniumpy.object_types.result_set.ResultSet object at 0x10a615710}}
23
24
   Equivalent Question if it were to be asked in the Tanium Console:
25
   Get Operating System containing "Windows" from all machines
26
27
   CSV Results of response:
28
   Operating System
29
   [no results]
30
   Windows Server 2008 R2 Standard
31
```

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
2
    sys.dont_write_bytecode = True
3
    # Determine our script name, script dir
5
    my_file = os.path.abspath(sys.argv[0])
6
    my_dir = os.path.dirname(my_file)
7
    # determine the pytan lib dir and add it to the path
    parent_dir = os.path.dirname(my_dir)
10
    pytan_root_dir = os.path.dirname(parent_dir)
11
12
    lib_dir = os.path.join(pytan_root_dir, 'lib')
    path_adds = [lib_dir]
13
14
    for aa in path_adds:
15
        if aa not in sys.path:
16
             sys.path.append(aa)
17
18
19
    # connection info for Tanium Server
20
    USERNAME = "Tanium User"
21
    PASSWORD = "T@n!um"
22
    HOST = "172.16.31.128"
23
    PORT = "443"
24
25
    # Logging conrols
26
    LOGLEVEL = 2
27
    DEBUGFORMAT = False
28
29
    import tempfile
30
31
    import pytan
32
    handler = pytan.Handler(
33
        username=USERNAME,
34
        password=PASSWORD,
35
        host=HOST,
36
        port=PORT,
37
        loglevel=LOGLEVEL,
38
        debugformat=DEBUGFORMAT,
39
40
41
    print handler
42
43
    # setup the arguments for the handler method
44
45
    kwargs["question_filters"] = [u'Operating System, that contains:Windows',
46
     u'Operating System, that does not contain: Windows']
47
    kwargs["sensors"] = [u'Computer Name',
48
     u'Folder Name Search with RegEx Match{dirname=Program Files, regex=Microsoft.*, invalidparam=test},
49
    kwarqs["question_options"] = [u'ignore_case', u'or']
50
51
    kwargs["qtype"] = u'manual'
52
    # call the handler with the ask method, passing in kwargs for arguments
53
```

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

```
Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
1
    2015-08-07 19:44:01,651 DEBUG
                                     pytan.handler.QuestionPoller: ID 1296: id resolved to 1296
2
    2015-08-07 19:44:01,651 DEBUG
                                     pytan.handler.QuestionPoller: ID 1296: expiration resolved to 2015-
3
4
   2015-08-07 19:44:01,651 DEBUG
                                     pytan.handler.QuestionPoller: ID 1296: query_text resolved to Get C
                                     pytan.handler.QuestionPoller: ID 1296: id resolved to 1296
   2015-08-07 19:44:01,651 DEBUG
   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:
10
   2015-08-07 19:44:06,659 DEBUG
                                     pytan.handler.QuestionPoller: ID 1296: Timing: Started: 2015-08-07
11
    2015-08-07 19:44:11,666 DEBUG
                                     pytan.handler.QuestionPoller: ID 1296: Progress: Tested: 0, Passed:
12
13
    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:
14
    2015-08-07 19:44:16,670 DEBUG
                                     pytan.handler.QuestionPoller: ID 1296: Timing: Started: 2015-08-07
15
    2015-08-07 19:44:21,677 DEBUG
                                     pytan.handler.QuestionPoller: ID 1296: Progress: Tested: 1, Passed:
16
                                     pytan.handler.QuestionPoller: ID 1296: Timing: Started: 2015-08-07
17
    2015-08-07 19:44:21,677 DEBUG
                                     pytan.handler.QuestionPoller: ID 1296: Progress Changed 50% (1 of 2
18
   2015-08-07 19:44:21,677 INFO
19
   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
20
   2015-08-07 19:44:31,691 DEBUG
                                     pytan.handler.QuestionPoller: ID 1296: Progress: Tested: 2, Passed:
21
   2015-08-07 19:44:31,692 DEBUG
                                     pytan.handler.OuestionPoller: ID 1296: Timing: Started: 2015-08-07
22
   2015-08-07 19:44:31,692 INFO
                                     pytan.handler.QuestionPoller: ID 1296: Progress Changed 100% (2 of
23
   2015-08-07 19:44:31,692 INFO
                                     pytan.handler.QuestionPoller: ID 1296: Reached Threshold of 99% (2
24
25
   Type of response: <type 'dict'>
```

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

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

```
# connection info for Tanium Server
20
    USERNAME = "Tanium User"
21
    PASSWORD = "T@n!um"
22
    HOST = "172.16.31.128"
23
    PORT = "443"
24
25
    # Logging conrols
26
    LOGLEVEL = 2
27
    DEBUGFORMAT = False
28
29
    import tempfile
30
31
32
    import pytan
    handler = pytan.Handler(
33
        username=USERNAME,
34
        password=PASSWORD,
35
        host=HOST,
36
        port=PORT,
37
        loglevel=LOGLEVEL,
38
        debugformat=DEBUGFORMAT,
39
40
41
    print handler
42
43
    # setup the arguments for the handler method
44
45
    kwargs = {}
    kwargs["question_filters"] = [u'Installed Applications, that regex match:.*Google (Search|Chrome).*'
46
    kwargs["sensors"] = [u'Computer Name',
47
     u'Last Logged In User',
48
     u'Installed Applications, that regex match:.*Google (Search|Chrome).*']
49
    kwargs["question_options"] = [u'ignore_case', u'or']
50
    kwargs["qtype"] = u'manual'
51
52
    # call the handler with the ask method, passing in kwargs for arguments
53
    response = handler.ask(**kwargs)
54
    import pprint, io
55
56
    print ""
57
    print "Type of response: ", type(response)
58
59
60
    print "Pretty print of response:"
61
    print pprint.pformat(response)
62
63
    print ""
    print "Equivalent Question if it were to be asked in the Tanium Console: "
65
    print response['question_object'].query_text
66
67
    # create an IO stream to store CSV results to
68
    out = io.BytesIO()
69
70
    # call the write_csv() method to convert response to CSV and store it in out
71
    response['question_results'].write_csv(out, response['question_results'])
72
73
    print ""
74
    print "CSV Results of response: "
75
    out = out.getvalue()
   if len(out.splitlines()) > 15:
```

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

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

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.

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

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

```
Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
1
   2015-08-07 19:44:46,869 DEBUG
                                    pytan.handler.QuestionPoller: ID 1298: id resolved to 1298
2
   2015-08-07 19:44:46,869 DEBUG
                                    pytan.handler.QuestionPoller: ID 1298: expiration resolved to 2015-
3
   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
6
   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:
10
   2015-08-07 19:44:51,877 DEBUG
                                     pytan.handler.QuestionPoller: ID 1298: Timing: Started: 2015-08-07
11
   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:
13
                                      pytan.handler.QuestionPoller: ID 1298: Timing: Started: 2015-08-07
    2015-08-07 19:44:56,881 DEBUG
14
    2015-08-07 19:45:01,885 DEBUG
                                      pytan.handler.QuestionPoller: ID 1298: Progress: Tested: 1, Passed:
15
   2015-08-07 19:45:01,885 DEBUG
                                      pytan.handler.QuestionPoller: ID 1298: Timing: Started: 2015-08-07
16
   2015-08-07 19:45:06,890 DEBUG
                                      pytan.handler.QuestionPoller: ID 1298: Progress: Tested: 2, Passed:
17
   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
19
   2015-08-07 19:45:06,890 INFO
                                     pytan.handler.QuestionPoller: ID 1298: Reached Threshold of 99% (2
20
21
   Type of response: <type 'dict'>
22
23
    Pretty print of response:
24
    {'poller_object': <pytan.pollers.QuestionPoller object at 0x10a5c9c90>,
25
     'poller_success': True,
26
     'question_object': <taniumpy.object_types.question.Question object at 0x10a5b98d0>,
27
     'question_results': <taniumpy.object_types.result_set.ResultSet object at 0x10a5e1410}}
28
29
    Equivalent Question if it were to be asked in the Tanium Console:
30
    Get Computer Name and Folder Name Search with RegEx Match[Program Files, , No, No] containing "Share
31
32
    CSV Results of response:
33
   Computer Name, "Folder Name Search with RegEx Match[Program Files, , No, No]"
34
   JTANIUM1.localdomain, "C:\Program Files\Common Files\Microsoft Shared\VS7Debug
35
   C:\Program Files\Common Files\Microsoft Shared\ink\ar-SA
36
   C:\Program Files\Common Files\Microsoft Shared\ink\ru-RU
37
    C:\Program Files\Common Files\Microsoft Shared\ink\fsdefinitions\keypad
38
    C:\Program Files\Common Files\Microsoft Shared\ink
39
    C:\Program Files\Common Files\Microsoft Shared\ink\sv-SE
40
   C:\Program Files\Common Files\Microsoft Shared\ink\uk-UA
41
   C:\Program Files\Common Files\Microsoft Shared\ink\sl-SI
42
   C:\Program Files\Common Files\Microsoft Shared\ink\hu-HU
43
   C:\Program Files\Common Files\Microsoft Shared\ink\zh-TW
44
   C:\Program Files\Common Files\Microsoft Shared\ink\zh-CN
45
   C:\Program Files\Common Files\Microsoft Shared\ink\fi-FI
46
   C:\Program Files\Common Files\Microsoft Shared
47
   C:\Program Files\Common Files\Microsoft Shared\ink\da-DK
48
    ..trimmed for brevity..
```

PyTan API Invalid Question Examples

Invalid ask manual question sensor help

Have ask_manual() return the help for sensors

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

```
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_sensors())
```

```
PytanHelp:
    Sensors Help
9
    ______
10
11
    Supplying sensors controls what columns will be showed when you ask a
12
    question.
13
14
    A sensor string is a human string that describes, at a minimum, a sensor.
15
    It can also optionally define a selector for the sensor, parameters for
16
    the sensor, a filter for the sensor, and options for the filter for the
17
    sensor. Sensors can be provided as a string or a list of strings.
18
19
    Examples for basic sensors
20
21
22
    Supplying a single sensor:
23
24
        'Computer Name'
25
26
    Supplying two sensors in a list of strings:
27
28
        ['Computer Name', 'IP Route Details']
29
30
    Supplying multiple sensors with selectors (name is the default
31
    selector if none is supplied):
32
33
34
             'Computer Name',
35
             'name: Computer Name',
36
            'id:1',
37
             'hash:123456789',
38
        ]
39
40
    Sensor Parameters
41
42
43
    Supplying parameters to a sensor can control the arguments that are
44
    supplied to a sensor, if that sensor takes any arguments.
45
46
    Sensor parameters must be surrounded with curly braces '{}',
47
    and must have a key and value specified that is separated by
48
    an equals '='. Multiple parameters must be seperated by
49
    a comma ', '. The key should match up to a valid parameter key
50
    for the sensor in question.
51
52
    If a parameter is supplied and the sensor doesn't have a
53
    corresponding key name, it will be ignored. If the sensor has
54
    parameters and a parameter is NOT supplied then one of two
55
    paths will be taken:
56
57
        * if the parameter does not require a default value, the
58
        parameter is left blank and not supplied.
59
        * if the parameter does require a value (pulldowns, for
60
        example), a default value is derived (for pulldowns,
61
        the first value available as a pulldown entry is used).
62
63
    Examples for sensors with parameters
64
```

```
66
    Supplying a single sensor with a single parameter 'dirname':
67
68
         'Sensor With Params{dirname=Program Files}'
69
70
    Supplying a single sensor with two parameters, 'param1' and
71
    'param2':
72
73
         'Sensor With Params{param1=value1,param2=value2}'
74
75
    Sensor Filters
76
77
78
    Supplying a filter to a sensor controls what data will be shown in
79
    those columns (sensors) you've provided.
80
81
    Sensor filters can be supplied by adding ', that FILTER: VALUE',
82
    where FILTER is a valid filter string, and VALUE is the string
83
    that you want FILTER to match on.
84
85
    See filter help for a list of all possible FILTER strings.
86
87
    See options help for a list of options that can control how
88
    the filter works.
89
91
    Examples for sensors with filters
92
93
    Supplying a sensor with a filter that limits the results to only
94
    show column data that matches the regular expression
95
    '.*Windows.*' (Tanium does a case insensitive match by default):
96
97
         'Computer Name, that contains: Windows'
98
99
    Supplying a sensor with a filter that limits the results to only
100
    show column data that matches the regular expression
101
    'Microsoft.*':
102
103
         'Computer Name, that starts with:Microsoft'
104
105
    Supply a sensor with a filter that limits the results to only
106
    show column data that has a version greater or equal to
107
    '39.0.0.0'. Since this sensor uses Version as its default result
108
    type, there is no need to change the value type using filter
109
    options.
110
111
         'Installed Application Version' \
112
         '{Application Name=Google Chrome}, that =>:39.0.0.0'
113
114
    Sensor Options
115
116
    ______
117
    Supplying options to a sensor can change how the filter for
118
    that sensor works.
119
120
    Sensor options can be supplied by adding ', opt:OPTION' or
121
    ', opt:OPTION:VALUE' for those options that require values,
122
    where OPTION is a valid option string, and VALUE is the
```

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

Invalid ask manual question filter help

Have ask_manual() return the help for filters

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

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

```
Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2
   Traceback (most recent call last):
     File "<string>", line 55, in <module>
3
     File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 130, in ask
4
       result = getattr(self, q_obj_map['handler']) (**kwargs)
5
     File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 274, in ask_manual
6
       raise pytan.exceptions.PytanHelp(pytan.help.help_filters())
   PytanHelp:
   Filters Help
   _____
10
11
   Filters are used generously throughout pytan. When used as part of a
12
   sensor string, they control what data is shown for the columns that
13
   the sensor returns. When filters are used for whole question filters,
14
   they control what rows will be returned. They are used by Groups to
```

```
define group membership, deploy actions to determine which machines
16
    should have the action deployed to it, and more.
17
18
    A filter string is a human string that describes, a sensor followed
19
    by ', that FILTER: VALUE', where FILTER is a valid filter string,
20
    and VALUE is the string that you want FILTER to match on.
21
22
    Valid Filters
23
24
25
         ' < '
26
             Help: Filter for less than VALUE
27
             Example: "Sensor1, that <: VALUE"
28
29
         'less'
30
             Help: Filter for less than VALUE
31
             Example: "Sensor1, that less: VALUE"
32
33
        '1t'
34
             Help: Filter for less than VALUE
35
             Example: "Sensor1, that lt:VALUE"
36
37
         'less than'
38
             Help: Filter for less than VALUE
39
             Example: "Sensor1, that less than: VALUE"
40
41
         '!<'
42
             Help: Filter for not less than VALUE
43
             Example: "Sensor1, that !<: VALUE"
44
45
         'notless'
46
             Help: Filter for not less than VALUE
47
             Example: "Sensor1, that notless: VALUE"
48
49
         'not less'
50
             Help: Filter for not less than VALUE
51
             Example: "Sensor1, that not less: VALUE"
52
53
54
         'not less than'
             Help: Filter for not less than VALUE
55
             Example: "Sensor1, that not less than: VALUE"
56
57
58
             Help: Filter for less than or equal to VALUE
59
             Example: "Sensor1, that <=:VALUE"
60
61
         'less equal'
62
             Help: Filter for less than or equal to VALUE
63
             Example: "Sensor1, that less equal: VALUE"
64
65
         'lessequal'
66
67
             Help: Filter for less than or equal to VALUE
             Example: "Sensor1, that lessequal: VALUE"
68
69
70
             Help: Filter for less than or equal to VALUE
71
             Example: "Sensor1, that le:VALUE"
72
73
```

```
'!<='
74
             Help: Filter for not less than or equal to VALUE
75
             Example: "Sensor1, that !<=:VALUE"
76
77
         'not less equal'
78
             Help: Filter for not less than or equal to VALUE
79
             Example: "Sensor1, that not less equal: VALUE"
80
81
         'not lessequal'
82
             Help: Filter for not less than or equal to VALUE
83
             Example: "Sensor1, that not lessequal:VALUE"
84
85
         151
86
             Help: Filter for greater than VALUE
87
             Example: "Sensor1, that >: VALUE"
88
89
         'greater'
90
             Help: Filter for greater than VALUE
91
92
             Example: "Sensor1, that greater: VALUE"
93
         'gt'
94
             Help: Filter for greater than VALUE
95
             Example: "Sensor1, that gt:VALUE"
96
97
         'greater than'
             Help: Filter for greater than VALUE
             Example: "Sensor1, that greater than: VALUE"
100
101
         '!>'
102
             Help: Filter for not greater than VALUE
103
             Example: "Sensor1, that !>: VALUE"
104
105
         'not greater'
106
             Help: Filter for not greater than VALUE
107
             Example: "Sensor1, that not greater: VALUE"
108
109
         'notgreater'
110
111
             Help: Filter for not greater than VALUE
112
             Example: "Sensor1, that notgreater: VALUE"
113
         'not greater than'
114
             Help: Filter for not greater than VALUE
115
             Example: "Sensor1, that not greater than: VALUE"
116
117
         ! => !
118
             Help: Filter for greater than or equal to VALUE
119
             Example: "Sensor1, that =>:VALUE"
120
121
         'greater equal'
122
             Help: Filter for greater than or equal to VALUE
123
             Example: "Sensor1, that greater equal: VALUE"
124
125
         'greaterequal'
126
             Help: Filter for greater than or equal to VALUE
127
             Example: "Sensor1, that greaterequal: VALUE"
128
129
         'ae'
130
             Help: Filter for greater than or equal to VALUE
131
```

```
Example: "Sensor1, that ge:VALUE"
132
133
         '!=>'
134
             Help: Filter for not greater than VALUE
135
             Example: "Sensor1, that !=>:VALUE"
136
137
         'not greater equal'
138
             Help: Filter for not greater than VALUE
139
             Example: "Sensor1, that not greater equal: VALUE"
140
141
         'notgreaterequal'
142
             Help: Filter for not greater than VALUE
143
144
              Example: "Sensor1, that notgreaterequal: VALUE"
145
146
             Help: Filter for equals to VALUE
147
             Example: "Sensor1, that =: VALUE"
148
149
         'equal'
150
             Help: Filter for equals to VALUE
151
             Example: "Sensor1, that equal: VALUE"
152
153
         'equals'
154
             Help: Filter for equals to VALUE
155
             Example: "Sensor1, that equals: VALUE"
156
157
         'eq'
158
             Help: Filter for equals to VALUE
159
             Example: "Sensor1, that eq:VALUE"
160
161
         '!='
162
             Help: Filter for not equals to VALUE
163
             Example: "Sensor1, that !=:VALUE"
164
165
         'not equal'
166
             Help: Filter for not equals to VALUE
167
             Example: "Sensor1, that not equal: VALUE"
168
169
170
         'notequal'
              Help: Filter for not equals to VALUE
171
             Example: "Sensor1, that notequal: VALUE"
172
173
         'not equals'
174
             Help: Filter for not equals to VALUE
175
176
             Example: "Sensor1, that not equals: VALUE"
177
         'notequals'
178
             Help: Filter for not equals to VALUE
179
             Example: "Sensor1, that notequals: VALUE"
180
181
         'ne'
182
183
              Help: Filter for not equals to VALUE
             Example: "Sensor1, that ne:VALUE"
184
185
         'contains'
186
             Help: Filter for contains VALUE (adds .* before and after VALUE)
187
             Example: "Sensor1, that contains: VALUE"
188
189
```

```
'does not contain'
190
             Help: Filter for does not contain VALUE (adds .* before and after VALUE)
191
             Example: "Sensor1, that does not contain: VALUE"
192
193
         'doesnotcontain'
194
             Help: Filter for does not contain VALUE (adds .* before and after VALUE)
195
             Example: "Sensor1, that doesnotcontain: VALUE"
196
197
         'not contains'
198
             Help: Filter for does not contain VALUE (adds .* before and after VALUE)
199
             Example: "Sensor1, that not contains: VALUE"
200
         'notcontains'
202
             Help: Filter for does not contain VALUE (adds .* before and after VALUE)
203
             Example: "Sensor1, that notcontains: VALUE"
204
205
         'starts with'
206
             Help: Filter for starts with VALUE (adds .* after VALUE)
207
             Example: "Sensor1, that starts with: VALUE"
208
209
         'startswith'
210
             Help: Filter for starts with VALUE (adds .* after VALUE)
211
             Example: "Sensor1, that startswith: VALUE"
212
213
         'does not start with'
214
             Help: Filter for does not start with VALUE (adds .* after VALUE)
215
             Example: "Sensor1, that does not start with: VALUE"
216
217
         'doesnotstartwith'
218
             Help: Filter for does not start with VALUE (adds .* after VALUE)
219
             Example: "Sensor1, that doesnotstartwith: VALUE"
220
221
         'not starts with'
222
             Help: Filter for does not start with VALUE (adds .* after VALUE)
223
             Example: "Sensor1, that not starts with: VALUE"
224
225
         'notstartswith'
226
             Help: Filter for does not start with VALUE (adds .* after VALUE)
227
             Example: "Sensor1, that notstartswith: VALUE"
228
229
         'ends with'
230
             Help: Filter for ends with VALUE (adds .* before VALUE)
231
             Example: "Sensor1, that ends with: VALUE"
232
233
         'endswith'
234
             Help: Filter for ends with VALUE (adds .* before VALUE)
235
             Example: "Sensor1, that endswith: VALUE"
236
237
         'does not end with'
238
             Help: Filter for does bit end with VALUE (adds .* before VALUE)
239
             Example: "Sensor1, that does not end with: VALUE"
240
241
         'doesnotendwith'
242
             Help: Filter for does bit end with VALUE (adds .* before VALUE)
243
             Example: "Sensor1, that doesnotendwith: VALUE"
244
245
         'not ends with'
246
             Help: Filter for does bit end with VALUE (adds .* before VALUE)
```

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

Invalid ask manual question option help

Have ask manual() return the help for options

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

```
Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
    Traceback (most recent call last):
2
      File "<string>", line 55, in <module>
3
      File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 130, in ask
4
        result = getattr(self, q_obj_map['handler']) (**kwargs)
5
      File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 277, in ask_manual
6
        raise pytan.exceptions.PytanHelp(pytan.help.help_options())
    PytanHelp:
8
    Options Help
    -----
10
11
    Options are used for controlling how filters act. When options are
12
    used as part of a sensor string, they change how the filters
13
14
    supplied as part of that sensor operate. When options are used for
    whole question options, they change how all of the question filters
15
    operate.
16
17
    When options are supplied for a sensor string, they must be
18
    supplied as ', opt:OPTION' or ', opt:OPTION:VALUE' for options
19
    that require a value.
20
21
    When options are supplied for question options, they must be
22
    supplied as 'OPTION' or 'OPTION: VALUE' for options that require
23
    a value.
24
25
    Options can be used on 'filter' or 'group', where 'group' pertains
26
    to group filters or question filters. All 'filter' options are also
27
    applicable to 'group' for question options.
28
29
    Valid Options
30
31
32
        'ignore_case'
33
            Help: Make the filter do a case insensitive match
34
            Usable on: filter
35
            Example for sensor: "Sensor1, opt:ignore_case"
36
            Example for question: "ignore_case"
37
38
        'match_case'
39
40
            Help: Make the filter do a case sensitive match
            Usable on: filter
41
            Example for sensor: "Sensor1, opt:match_case"
42
            Example for question: "match_case"
43
44
        'match_any_value'
45
            Help: Make the filter match any value
46
            Usable on: filter
47
            Example for sensor: "Sensor1, opt:match_any_value"
48
            Example for question: "match_any_value"
49
50
        'match_all_values'
51
52
            Help: Make the filter match all values
53
            Usable on: filter
            Example for sensor: "Sensor1, opt:match_all_values"
54
            Example for question: "match_all_values"
55
56
        'max_data_age'
57
```

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

Invalid ask manual question sensor

Ask a question using a sensor that does not exist

Example Python Code

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

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

```
Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
    Traceback (most recent call last):
2
      File "<string>", line 55, in <module>
3
      File "/Users/jolsen/qh/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 305, in ask_manual
6
        **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 1882, in _ask_manual
10
        sensor_defs = self._get_sensor_defs(sensor_defs)
11
     File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 1418, in _get_sensor_defs
12
13
        d['sensor_obj'] = self.get('sensor', **def_search)[0]
     File "/Users/jolsen/gh/pytan/lib/pytan/utils.py", line 2699, in wrap
14
15
        ret = f(*args, **kwargs)
      File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 1250, in get
16
17
        return self._get_multi(obj_map, **kwargs)
      File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 1363, in _get_multi
18
        found = self._find(api_obj_multi, **kwargs)
19
     File "/Users/jolsen/gh/pytan/lib/pytan/utils.py", line 2699, in wrap
20
        ret = f(*args, **kwargs)
21
      File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 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
2
    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
9
    parent_dir = os.path.dirname(my_dir)
10
    pytan_root_dir = os.path.dirname(parent_dir)
11
    lib_dir = os.path.join(pytan_root_dir, 'lib')
12
13
   path_adds = [lib_dir]
15
   for aa in path_adds:
        if aa not in sys.path:
16
            sys.path.append(aa)
17
18
19
    # connection info for Tanium Server
20
    USERNAME = "Tanium User"
21
    PASSWORD = "T@n!um"
22
    HOST = "172.16.31.128"
23
    PORT = "443"
24
25
    # Logging conrols
27
    LOGLEVEL = 2
28
   DEBUGFORMAT = False
29
    import tempfile
30
31
32
    import pytan
33
    handler = pytan.Handler(
34
        username=USERNAME,
        password=PASSWORD,
35
        host=HOST,
36
        port=PORT,
37
38
        loglevel=LOGLEVEL,
        debugformat=DEBUGFORMAT,
41
   print handler
42.
43
44
    # setup the arguments for the handler method
45
   kwarqs = {}
   kwargs["sensors"] = u'Computer name, that does not meet:little'
```

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

```
Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
   Traceback (most recent call last):
2
     File "<string>", line 55, in <module>
3
     File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 130, in ask
4
       result = getattr(self, q_obj_map['handler']) (**kwargs)
5
     File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 297, in ask_manual
6
       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
10
       raise pytan.exceptions.HumanParserError(err(split_filter[1]))
11
   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 ({}).

```
import os
    import sys
2
    sys.dont_write_bytecode = True
3
    # 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)
10
    pytan_root_dir = os.path.dirname(parent_dir)
    lib_dir = os.path.join(pytan_root_dir, 'lib')
12
    path_adds = [lib_dir]
13
14
    for aa in path_adds:
15
        if aa not in sys.path:
16
            sys.path.append(aa)
17
18
19
    # connection info for Tanium Server
20
    USERNAME = "Tanium User"
21
   PASSWORD = "T@n!um"
22
```

```
HOST = "172.16.31.128"
    PORT = "443"
24
25
    # Logging conrols
26
    LOGLEVEL = 2
27
    DEBUGFORMAT = False
28
29
    import tempfile
30
31
    import pytan
32
    handler = pytan.Handler(
33
        username=USERNAME,
34
        password=PASSWORD,
35
        host=HOST,
36
        port=PORT,
37
        loglevel=LOGLEVEL,
38
        debugformat=DEBUGFORMAT,
39
40
41
    print handler
42
43
    # setup the arguments for the handler method
44
    kwargs = {}
45
    kwargs["sensors"] = u'Folder Name Search with RegEx Match{dirname=Program Files, regex= ! * } { } } '
46
    kwargs["qtype"] = u'manual'
47
48
49
    # call the handler with the ask method, passing in kwargs for arguments
50
    # this should throw an exception: pytan.exceptions.HumanParserError
51
    import traceback
52
    try:
53
        handler.ask(**kwargs)
54
    except Exception as e:
55
        traceback.print_exc(file=sys.stdout)
```

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

Invalid ask manual question option

Ask a question using an invalid option.

```
import os
    import sys
2
    sys.dont_write_bytecode = True
3
    # 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)
10
   pytan_root_dir = os.path.dirname(parent_dir)
11
    lib_dir = os.path.join(pytan_root_dir, 'lib')
12
    path_adds = [lib_dir]
13
14
    for aa in path_adds:
15
        if aa not in sys.path:
16
            sys.path.append(aa)
17
18
19
    # connection info for Tanium Server
20
    USERNAME = "Tanium User"
21
    PASSWORD = "T@n!um"
22
    HOST = "172.16.31.128"
23
    PORT = "443"
24
25
26
    # Logging conrols
    LOGLEVEL = 2
27
    DEBUGFORMAT = False
28
29
    import tempfile
30
31
    import pytan
32
    handler = pytan.Handler(
33
        username=USERNAME,
34
        password=PASSWORD,
35
        host=HOST,
36
        port=PORT,
37
        loglevel=LOGLEVEL,
38
39
        debugformat=DEBUGFORMAT,
41
    print handler
42
43
    # setup the arguments for the handler method
44
    kwargs = \{\}
45
    kwargs["sensors"] = u'Operating system, opt:bad'
46
47
    kwargs["qtype"] = u'manual'
48
49
    # call the handler with the ask method, passing in kwargs for arguments
50
    # this should throw an exception: pytan.exceptions.HumanParserError
51
    import traceback
52
53
        handler.ask(**kwargs)
54
    except Exception as e:
55
        traceback.print_exc(file=sys.stdout)
```

```
Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
   Traceback (most recent call last):
2
     File "<string>", line 55, in <module>
3
     File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 130, in ask
4
        result = getattr(self, q_obj_map['handler']) (**kwargs)
     File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 297, in ask_manual
6
       sensor_defs = pytan.utils.dehumanize_sensors(sensors)
     File "/Users/jolsen/gh/pytan/lib/pytan/utils.py", line 1477, in dehumanize_sensors
8
       s, parsed_options = extract_options(s)
     File "/Users/jolsen/gh/pytan/lib/pytan/utils.py", line 1721, in extract_options
10
       parsed_options = map_options(parsed_options, ['filter'])
11
     File "/Users/jolsen/gh/pytan/lib/pytan/utils.py", line 1751, in map_options
12
       raise pytan.exceptions.HumanParserError(err(option))
13
   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
2
    sys.dont_write_bytecode = True
    # Determine our script name, script dir
   my file = os.path.abspath(sys.argv[0])
6
   my_dir = os.path.dirname(my_file)
    # determine the pytan lib dir and add it to the path
9
    parent_dir = os.path.dirname(my_dir)
10
    pytan_root_dir = os.path.dirname(parent_dir)
11
    lib dir = os.path.join(pytan root dir, 'lib')
12
   path_adds = [lib_dir]
13
14
   for aa in path_adds:
15
        if aa not in sys.path:
16
            sys.path.append(aa)
17
18
19
    # connection info for Tanium Server
20
    USERNAME = "Tanium User"
21
    PASSWORD = "T@n!um"
22
    HOST = "172.16.31.128"
23
    PORT = "443"
24
25
    # Logging conrols
26
    LOGLEVEL = 2
27
   DEBUGFORMAT = False
28
29
    import tempfile
30
31
   import pytan
32
   handler = pytan.Handler(
```

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

```
Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
   Traceback (most recent call last):
2
     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
6
       sensor_defs = pytan.utils.dehumanize_sensors(sensors)
7
     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
10
       raise pytan.exceptions.HumanParserError(err(sp, pytan.constants.PARAM_KEY_SPLIT))
   HumanParserError: Parameter Dweedle missing key/value seperator (=)
12
```

PyTan API Valid Get Object Examples

Get action by id

Get an action by id

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

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

Get question by id

Get a question by id

```
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]
13
14
    for aa in path_adds:
15
        if aa not in sys.path:
16
             sys.path.append(aa)
17
18
19
    # connection info for Tanium Server
20
    USERNAME = "Tanium User"
21
    PASSWORD = "T@n!um"
22
    HOST = "172.16.31.128"
23
    PORT = "443"
24
25
    # Logging conrols
26
    LOGLEVEL = 2
27
    DEBUGFORMAT = False
28
29
    import tempfile
30
31
    import pytan
32
    handler = pytan.Handler(
33
        username=USERNAME,
34
        password=PASSWORD,
35
36
        host=HOST,
37
        port=PORT,
        loglevel=LOGLEVEL,
38
        debugformat=DEBUGFORMAT,
39
40
41
    print handler
42
43
    # setup the arguments for the handler method
44
    kwargs = {}
45
    kwargs["objtype"] = u'question'
46
    kwargs["id"] = 1
47
48
    # call the handler with the get method, passing in kwargs for arguments
49
    response = handler.get(**kwargs)
50
51
52
    print "Type of response: ", type(response)
53
54
    print ""
55
    print "print of response:"
56
    print response
57
    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)
```

```
70 print out
```

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

Get saved question by names

Get two saved questions by name

```
import os
   import sys
   sys.dont_write_bytecode = True
    # Determine our script name, script dir
   my_file = os.path.abspath(sys.argv[0])
6
   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)
10
   pytan_root_dir = os.path.dirname(parent_dir)
11
   lib_dir = os.path.join(pytan_root_dir, 'lib')
12
   path_adds = [lib_dir]
13
14
   for aa in path_adds:
15
```

```
if aa not in sys.path:
16
             sys.path.append(aa)
17
18
19
    # connection info for Tanium Server
20
    USERNAME = "Tanium User"
21
    PASSWORD = "T@n!um"
22
    HOST = "172.16.31.128"
23
    PORT = "443"
24
25
    # Logging conrols
26
    LOGLEVEL = 2
27
    DEBUGFORMAT = False
28
29
    import tempfile
30
31
    import pytan
32
    handler = pytan.Handler(
33
34
        username=USERNAME,
        password=PASSWORD,
35
        host=HOST,
36
        port=PORT,
37
        loglevel=LOGLEVEL,
38
        debugformat=DEBUGFORMAT,
39
40
41
    print handler
42
43
    # setup the arguments for the handler method
44
    kwargs = {}
45
    kwargs["objtype"] = u'saved_question'
46
    kwargs["name"] = [u'Installed Applications', u'Computer Name']
47
    # call the handler with the get method, passing in kwargs for arguments
49
    response = handler.get(**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
66
    if len(out.splitlines()) > 15:
67
        out = out.splitlines()[0:15]
        out.append('..trimmed for brevity..')
68
        out = ' \ n'. join (out)
69
70
   print out
71
```

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

Get userrole by id

Get a user role by id.

```
import os
   import sys
2
    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)
10
    pytan_root_dir = os.path.dirname(parent_dir)
11
    lib_dir = os.path.join(pytan_root_dir, 'lib')
12
   path_adds = [lib_dir]
13
14
   for aa in path_adds:
15
        if aa not in sys.path:
16
            sys.path.append(aa)
17
18
19
    # connection info for Tanium Server
20
```

```
USERNAME = "Tanium User"
21
    PASSWORD = "T@n!um"
22
    HOST = "172.16.31.128"
23
    PORT = "443"
24
25
    # Logging conrols
26
    LOGLEVEL = 2
27
    DEBUGFORMAT = False
28
29
    import tempfile
30
31
32
    import pytan
33
    handler = pytan.Handler(
        username=USERNAME,
34
        password=PASSWORD,
35
        host=HOST,
36
        port=PORT,
37
        loglevel=LOGLEVEL,
38
        debugformat=DEBUGFORMAT,
39
40
41
    print handler
42
43
    # setup the arguments for the handler method
44
45
    kwargs = {}
    kwargs["objtype"] = u'userrole'
46
    kwargs["id"] = 1
47
48
    # call the handler with the get method, passing in kwargs for arguments
49
    response = handler.get(**kwargs)
50
51
    print ""
52
    print "Type of response: ", type(response)
53
    print ""
55
    print "print of response:"
56
    print response
57
    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
```

```
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'>
```

```
4
    print of response:
5
    UserRoleList, len: 1
6
7
    length of response (number of objects returned):
10
    print the first object returned in JSON format:
11
12
      "_type": "role",
13
      "description": "Administrators can perform all functions in the system, including creating other u
14
      "id": 1,
15
      "name": "Administrator",
16
      "permissions": {
17
        "_type": "permissions",
18
        "permission": [
19
          "admin",
20
          "sensor_read",
21
          "sensor_write",
22
          "question_read",
23
          "question_write",
24
          "action_read",
25
          "action_write",
26
    ..trimmed for brevity..
27
```

Get leader clients

Get all clients that are Leader status

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

```
25
    # Logging conrols
26
    LOGLEVEL = 2
27
    DEBUGFORMAT = False
28
29
    import tempfile
30
31
    import pytan
32
    handler = pytan.Handler(
33
        username=USERNAME,
34
        password=PASSWORD,
35
        host=HOST,
36
37
        port=PORT,
        loglevel=LOGLEVEL,
38
        debugformat=DEBUGFORMAT,
39
40
41
    print handler
42
43
    # setup the arguments for the handler method
44
    kwargs = \{\}
45
    kwargs["objtype"] = u'client'
46
    kwargs["status"] = u'Leader'
47
48
    # call the handler with the get method, passing in kwargs for arguments
49
    response = handler.get(**kwargs)
50
51
    print ""
52
    print "Type of response: ", type(response)
53
54
    print ""
55
    print "print of response:"
    print response
57
    print ""
59
    print "length of response (number of objects returned): "
60
    print len(response)
61
    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
    print out
71
```

```
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: 1
```

```
length of response (number of objects returned):
10
    print the first object returned in JSON format:
11
12
      "_type": "client_status",
13
      "cache_row_id": 1,
14
      "computer_id": "3741604154",
15
      "full_version": "6.0.314.1195",
16
      "host_name": "JTANIUM1.localdomain",
17
      "ipaddress_client": "172.16.31.128",
18
      "ipaddress_server": "172.16.31.128",
19
      "last_registration": "2015-08-07T19:45:00",
20
      "port_number": 17473,
21
      "protocol_version": 314,
22
      "public_key_valid": 1,
23
      "receive_state": "Previous Only",
24
      "send_state": "Backward Only",
25
      "status": "Leader"
    ..trimmed for brevity..
```

Get setting by name

Get a system setting by name

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

```
29
    import tempfile
30
31
    import pytan
32
    handler = pytan.Handler(
33
        username=USERNAME,
34
        password=PASSWORD,
35
        host=HOST,
36
        port=PORT,
37
        loglevel=LOGLEVEL,
38
        debugformat=DEBUGFORMAT,
39
40
41
    print handler
42
43
    # setup the arguments for the handler method
44
    kwargs = {}
45
    kwargs["objtype"] = u'setting'
46
    kwargs["name"] = u'control_address'
47
48
    # call the handler with the get method, passing in kwargs for arguments
49
    response = handler.get(**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): "
    print len(response)
61
    print ""
63
    print "print the first object returned in JSON format:"
64
    out = response.to_json(response[0])
65
66
    if len(out.splitlines()) > 15:
67
        out = out.splitlines()[0:15]
        out.append('..trimmed for brevity..')
68
        out = ' \ n'. join (out)
69
70
    print out
71
```

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

```
12
      "_type": "system_setting",
13
      "default_value": "512:17473:127.0.0.1",
14
      "hidden_flag": 0,
15
      "id": 58,
      "name": "control_address",
17
      "read_only_flag": 0,
18
      "setting_type": "Server",
19
      "value": "512:17473:127.0.0.1",
20
      "value_type": "Text"
21
22
```

Get user by name

Get a user by name

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

```
loglevel=LOGLEVEL,
38
        debugformat=DEBUGFORMAT,
39
40
41
    print handler
42
43
    # setup the arguments for the handler method
44
    kwargs = \{\}
45
    kwarqs["objtype"] = u'user'
46
    kwargs["name"] = u'Tanium User'
47
48
49
    # call the handler with the get method, passing in kwargs for arguments
    response = handler.get(**kwargs)
50
51
    print ""
52
   print "Type of response: ", type(response)
53
54
   print ""
55
   print "print of response:"
   print response
57
58
   print ""
59
   print "length of response (number of objects returned): "
60
   print len(response)
61
   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)
   print out
```

```
Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2
   Type of response: <class 'taniumpy.object_types.user_list.UserList'>
3
4
   print of response:
   UserList, len: 1
    length of response (number of objects returned):
8
10
   print the first object returned in JSON format:
11
12
13
      "_type": "user",
      "deleted_flag": 0,
14
      "group_id": 0,
15
      "id": 2,
16
      "last_login": "2015-08-07T19:45:07",
17
      "local_admin_flag": 1,
18
      "name": "Tanium User",
19
      "permissions": {
```

```
"_type": "permissions",
"permission": [
"admin",
"sensor_read",
"sensor_write",
"question_read",
..trimmed for brevity..
```

Get sensor by id

Get a sensor by id

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

```
print handler
42
43
    # setup the arguments for the handler method
44
    kwarqs = \{\}
45
    kwargs["objtype"] = u'sensor'
46
    kwarqs["id"] = 1
47
48
    # call the handler with the get method, passing in kwargs for arguments
49
    response = handler.get(**kwargs)
50
51
   print ""
52
   print "Type of response: ", type(response)
53
54
55
   print "print of response:"
56
   print response
57
58
   print ""
59
   print "length of response (number of objects returned): "
   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:
        out = out.splitlines()[0:15]
67
        out.append('..trimmed for brevity..')
68
        out = ' \ n'. join (out)
69
70
   print out
71
```

```
Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
1
2
3
    Type of response: <class 'taniumpy.object_types.sensor_list.SensorList'>
    print of response:
    SensorList, len: 1
    length of response (number of objects returned):
10
    print the first object returned in JSON format:
11
12
      "_type": "sensor",
13
      "category": "Reserved",
14
      "description": "The recorded state of each action a client has taken recently in the form of id:st
15
16
      "exclude_from_parse_flag": 1,
      "hash": 1792443391,
17
      "hidden_flag": 0,
18
      "id": 1,
19
      "ignore_case_flag": 1,
20
      "max_age_seconds": 3600,
21
      "name": "Action Statuses",
22
      "queries": {
23
        "_type": "queries",
```

Get sensor by mixed

Get multiple sensors by id, name, and hash

```
import os
    import sys
    sys.dont_write_bytecode = True
    # Determine our script name, script dir
    my_file = os.path.abspath(sys.argv[0])
6
    my_dir = os.path.dirname(my_file)
    # determine the pytan lib dir and add it to the path
9
    parent_dir = os.path.dirname(my_dir)
10
   pytan_root_dir = os.path.dirname(parent_dir)
11
    lib_dir = os.path.join(pytan_root_dir, 'lib')
12
   path_adds = [lib_dir]
13
14
    for aa in path_adds:
15
        if aa not in sys.path:
16
            sys.path.append(aa)
17
18
19
    # connection info for Tanium Server
20
    USERNAME = "Tanium User"
21
    PASSWORD = "T@n!um"
22
    HOST = "172.16.31.128"
23
    PORT = "443"
24
25
26
    # Logging conrols
27
    LOGLEVEL = 2
   DEBUGFORMAT = False
28
29
    import tempfile
30
31
32
    import pytan
33
    handler = pytan.Handler(
        username=USERNAME,
34
        password=PASSWORD,
35
        host=HOST,
36
        port=PORT,
37
        loglevel=LOGLEVEL,
        debugformat=DEBUGFORMAT,
40
41
   print handler
42
43
    # setup the arguments for the handler method
44
   kwargs = {}
```

```
kwarqs["objtype"] = u'sensor'
    kwargs["hash"] = [u'322086833']
47
    kwargs["name"] = [u'Computer Name']
48
    kwargs["id"] = [1, 2]
49
50
    # call the handler with the get method, passing in kwargs for arguments
51
    response = handler.get(**kwargs)
52
53
   print ""
54
   print "Type of response: ", type(response)
55
    print ""
57
    print "print of response:"
58
   print response
59
60
   print ""
61
   print "length of response (number of objects returned): "
62
   print len(response)
63
   print ""
65
   print "print the first object returned in JSON format:"
66
   out = response.to_json(response[0])
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
73
```

```
Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
1
2
    Type of response: <class 'taniumpy.object_types.sensor_list.SensorList'>
3
    print of response:
5
    SensorList, len: 4
    length of response (number of objects returned):
10
    print the first object returned in JSON format:
11
12
      "_type": "sensor",
13
      "category": "Reserved",
14
      "description": "The recorded state of each download a client has made recently in the form of hash
15
      "exclude_from_parse_flag": 0,
16
      "hash": 322086833,
17
      "hidden_flag": 0,
18
      "id": 4,
      "ignore_case_flag": 1,
20
      "max_age_seconds": 900,
21
      "name": "Download Statuses",
22
      "queries": {
23
        "_type": "queries",
24
        "query": [
25
          {
```

```
27 ..trimmed for brevity..
```

Get whitelisted url by id

Get a whitelisted url by id

```
import os
    import sys
    sys.dont_write_bytecode = True
    # Determine our script name, script dir
5
   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
9
    parent_dir = os.path.dirname(my_dir)
10
    pytan_root_dir = os.path.dirname(parent_dir)
11
    lib_dir = os.path.join(pytan_root_dir, 'lib')
12
   path_adds = [lib_dir]
13
14
    for aa in path_adds:
15
        if aa not in sys.path:
16
            sys.path.append(aa)
17
18
19
    # connection info for Tanium Server
20
    USERNAME = "Tanium User"
21
    PASSWORD = "T@n!um"
22
    HOST = "172.16.31.128"
23
    PORT = "443"
24
25
    # Logging conrols
26
27
    LOGLEVEL = 2
   DEBUGFORMAT = False
28
29
   import tempfile
30
31
32
    import pytan
33
   handler = pytan.Handler(
34
        username=USERNAME,
        password=PASSWORD,
35
        host=HOST,
36
        port=PORT,
37
        loglevel=LOGLEVEL,
38
39
        debugformat=DEBUGFORMAT,
40
41
   print handler
42
43
    # setup the arguments for the handler method
44
   kwargs = {}
45
   kwargs["objtype"] = u'whitelisted_url'
46
   kwargs["id"] = 1
```

```
48
    # call the handler with the get method, passing in kwargs for arguments
49
    response = handler.get(**kwargs)
50
51
    print ""
52
   print "Type of response: ", type(response)
53
   print ""
55
   print "print of response:"
56
   print response
57
    print ""
    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:
        out = out.splitlines()[0:15]
67
        out.append('..trimmed for brevity..')
68
        out = '\n'.join(out)
69
70
   print out
71
```

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

Get group by name

Get a group by name

Example Python Code

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

```
# Determine our script name, script dir
5
   my_file = os.path.abspath(sys.argv[0])
6
   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)
10
   pytan_root_dir = os.path.dirname(parent_dir)
11
    lib_dir = os.path.join(pytan_root_dir, 'lib')
12
   path_adds = [lib_dir]
13
14
15
    for aa in path_adds:
16
        if aa not in sys.path:
            sys.path.append(aa)
17
18
19
    # connection info for Tanium Server
20
    USERNAME = "Tanium User"
21
   PASSWORD = "T@n!um"
22
    HOST = "172.16.31.128"
23
   PORT = "443"
24
25
    # Logging conrols
26
    LOGLEVEL = 2
27
   DEBUGFORMAT = False
28
29
    import tempfile
30
31
    import pytan
32
   handler = pytan.Handler(
33
        username=USERNAME,
34
        password=PASSWORD,
35
        host=HOST,
36
        port=PORT,
37
        loglevel=LOGLEVEL,
38
        debugformat=DEBUGFORMAT,
39
40
41
    print handler
42
43
    # setup the arguments for the handler method
44
    kwarqs = \{\}
45
    kwargs["objtype"] = u'group'
46
    kwargs["name"] = u'All Computers'
47
48
    # call the handler with the get method, passing in kwargs for arguments
    response = handler.get(**kwargs)
50
51
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): "
   print len(response)
```

```
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
71
```

```
Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2
    Type of response: <class 'taniumpy.object_types.group_list.GroupList'>
3
    print of response:
5
    GroupList, len: 1
    length of response (number of objects returned):
10
    print the first object returned in JSON format:
11
12
      "_type": "group",
13
      "and_flag": 0,
14
      "deleted_flag": 0,
15
      "filters": {
16
        "_type": "filters",
17
        "filter": []
18
19
      "id": 64,
20
      "name": "All Computers",
21
      "not_flag": 0,
22
      "sub_groups": {
23
        "_type": "groups",
24
        "group": []
25
26
      },
    ..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)
```

```
# determine the pytan lib dir and add it to the path
9
    parent_dir = os.path.dirname(my_dir)
10
    pytan_root_dir = os.path.dirname(parent_dir)
11
    lib_dir = os.path.join(pytan_root_dir, 'lib')
12
    path_adds = [lib_dir]
13
14
    for aa in path_adds:
15
        if aa not in sys.path:
16
             sys.path.append(aa)
17
18
19
    # connection info for Tanium Server
20
    USERNAME = "Tanium User"
21
    PASSWORD = "T@n!um"
22
    HOST = "172.16.31.128"
23
    PORT = "443"
24
25
    # Logging conrols
26
    LOGLEVEL = 2
27
    DEBUGFORMAT = False
28
29
    import tempfile
30
31
    import pytan
32
33
    handler = pytan.Handler(
        username=USERNAME,
34
        password=PASSWORD,
35
        host=HOST,
36
        port=PORT,
37
        loglevel=LOGLEVEL,
38
        debugformat=DEBUGFORMAT,
39
40
41
    print handler
42
43
    # setup the arguments for the handler method
44
    kwargs = {}
45
    kwargs["objtype"] = u'sensor'
46
    kwargs["hash"] = u'322086833'
47
48
    # call the handler with the get method, passing in kwargs for arguments
49
    response = handler.get(**kwargs)
50
51
52
    print "Type of response: ", type(response)
53
   print ""
55
    print "print of response:"
56
    print response
57
58
    print ""
    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:"
   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
```

```
Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2
   Type of response: <class 'taniumpy.object_types.sensor_list.SensorList'>
3
    print of response:
    SensorList, len: 1
6
    length of response (number of objects returned):
Q
10
    print the first object returned in JSON format:
11
12
      "_type": "sensor",
13
      "category": "Reserved",
14
      "description": "The recorded state of each download a client has made recently in the form of hash
15
      "exclude_from_parse_flag": 0,
16
      "hash": 322086833,
17
      "hidden_flag": 0,
18
      "id": 4,
19
      "ignore_case_flag": 1,
20
      "max_age_seconds": 900,
21
      "name": "Download Statuses",
22
      "queries": {
23
        "_type": "queries",
24
        "query": [
25
26
    ..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')
12
    path_adds = [lib_dir]
13
14
    for aa in path_adds:
15
        if aa not in sys.path:
16
17
             sys.path.append(aa)
18
19
    # connection info for Tanium Server
20
    USERNAME = "Tanium User"
21
    PASSWORD = "T@n!um"
22
    HOST = "172.16.31.128"
23
    PORT = "443"
24
25
    # Logging conrols
26
    LOGLEVEL = 2
27
    DEBUGFORMAT = False
28
29
    import tempfile
30
31
    import pytan
32
    handler = pytan.Handler(
33
        username=USERNAME,
34
        password=PASSWORD,
35
36
        host=HOST,
37
        port=PORT,
        loglevel=LOGLEVEL,
38
        debugformat=DEBUGFORMAT,
39
40
41
    print handler
42
43
    # setup the arguments for the handler method
44
    kwargs = {}
45
    kwargs["objtype"] = u'package'
46
    kwargs["name"] = u'Distribute Tanium Standard Utilities'
47
48
    # call the handler with the get method, passing in kwargs for arguments
49
50
    response = handler.get(**kwargs)
51
52
    print "Type of response: ", type(response)
53
54
   print ""
55
    print "print of response:"
56
   print response
57
   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)
```

```
70 print out
```

```
Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
   Type of response: <class 'taniumpy.object_types.package_spec_list.PackageSpecList'>
   print of response:
   PackageSpecList, len: 1
    length of response (number of objects returned):
10
    print the first object returned in JSON format:
11
12
      "_type": "package_spec",
13
      "available_time": "2015-08-07T13:23:22",
14
      "command": "cmd /c cscript install-standard-utils.vbs \"Tools\\StdUtils\"",
15
      "command_timeout": 2700,
16
      "creation_time": "2015-08-07T13:22:19",
17
      "deleted_flag": 0,
18
      "display_name": "Distribute Tanium Standard Utilities",
19
      "expire_seconds": 3300,
20
      "files": {
21
        "_type": "package_files",
22
        "file": [
23
24
            "_type": "file",
25
            "bytes_downloaded": 0,
26
    ..trimmed for brevity..
```

Get sensor by names

Get multiple sensors 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])
6
   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)
10
   pytan_root_dir = os.path.dirname(parent_dir)
11
   lib_dir = os.path.join(pytan_root_dir, 'lib')
12
   path_adds = [lib_dir]
13
14
   for aa in path_adds:
15
```

```
if aa not in sys.path:
16
             sys.path.append(aa)
17
18
19
    # connection info for Tanium Server
20
    USERNAME = "Tanium User"
21
    PASSWORD = "T@n!um"
22
    HOST = "172.16.31.128"
23
    PORT = "443"
24
25
    # Logging conrols
26
    LOGLEVEL = 2
27
    DEBUGFORMAT = False
28
29
    import tempfile
30
31
    import pytan
32
    handler = pytan.Handler(
33
34
        username=USERNAME,
        password=PASSWORD,
35
        host=HOST,
36
        port=PORT,
37
        loglevel=LOGLEVEL,
38
        debugformat=DEBUGFORMAT,
39
40
41
    print handler
42
43
    # setup the arguments for the handler method
44
    kwargs = {}
45
    kwargs["objtype"] = u'sensor'
46
    kwargs["name"] = [u'Computer Name', u'Action Statuses']
47
48
    # call the handler with the get method, passing in kwargs for arguments
49
    response = handler.get(**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
66
    if len(out.splitlines()) > 15:
67
        out = out.splitlines()[0:15]
        out.append('..trimmed for brevity..')
68
        out = ' \ n'. join (out)
69
70
   print out
71
```

```
Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2
    Type of response: <class 'taniumpy.object_types.sensor_list.SensorList'>
3
    print of response:
5
    SensorList, len: 2
    length of response (number of objects returned):
10
    print the first object returned in JSON format:
11
12
13
      "_type": "sensor",
      "category": "Reserved",
14
      "description": "The assigned name of the client machine.\nExample: workstation-1.company.com",
15
      "exclude_from_parse_flag": 0,
16
      "hash": 3409330187,
17
      "hidden_flag": 0,
18
      "id": 3,
19
      "ignore_case_flag": 1,
20
      "max_age_seconds": 86400,
21
      "name": "Computer Name",
22
      "queries": {
23
        "_type": "queries",
24
        "query": [
25
26
    ..trimmed for brevity..
```

Get saved question by name

Get saved question by name

Example Python Code

```
import os
   import sys
2
    sys.dont_write_bytecode = True
    # Determine our script name, script dir
   my_file = os.path.abspath(sys.argv[0])
6
   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)
10
    pytan_root_dir = os.path.dirname(parent_dir)
11
    lib_dir = os.path.join(pytan_root_dir, 'lib')
12
   path_adds = [lib_dir]
13
14
   for aa in path_adds:
15
        if aa not in sys.path:
16
            sys.path.append(aa)
17
18
19
    # connection info for Tanium Server
20
```

```
USERNAME = "Tanium User"
21
    PASSWORD = "T@n!um"
22
    HOST = "172.16.31.128"
23
    PORT = "443"
24
25
    # Logging conrols
26
    LOGLEVEL = 2
27
    DEBUGFORMAT = False
28
29
    import tempfile
30
31
32
    import pytan
33
    handler = pytan.Handler(
        username=USERNAME,
34
        password=PASSWORD,
35
        host=HOST,
36
        port=PORT,
37
        loglevel=LOGLEVEL,
38
        debugformat=DEBUGFORMAT,
39
40
41
    print handler
42
43
    # setup the arguments for the handler method
44
45
    kwargs = {}
    kwargs["objtype"] = u'saved_question'
46
    kwargs["name"] = u'Installed Applications'
47
48
    # call the handler with the get method, passing in kwargs for arguments
49
    response = handler.get(**kwargs)
50
51
    print ""
52
    print "Type of response: ", type(response)
53
    print ""
55
    print "print of response:"
56
    print response
57
    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
```

```
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:
5
    SavedQuestionList, len: 1
6
    length of response (number of objects returned):
10
    print the first object returned in JSON format:
11
12
      "_type": "saved_question",
13
      "action_tracking_flag": 0,
14
15
      "archive_enabled_flag": 0,
      "archive_owner": {
16
        "_type": "user"
17
      },
18
      "expire_seconds": 600,
19
      "hidden_flag": 0,
20
      "id": 64,
21
      "issue_seconds": 120,
22
      "issue_seconds_never_flag": 0,
23
      "keep_seconds": 0,
24
      "metadata": {
25
        "_type": "metadata",
26
    ..trimmed for brevity..
27
```

Get user by id

Get a user by id

Example Python Code

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

```
25
    # Logging conrols
26
    LOGLEVEL = 2
27
    DEBUGFORMAT = False
28
29
    import tempfile
31
    import pytan
32
    handler = pytan.Handler(
33
        username=USERNAME,
34
        password=PASSWORD,
35
        host=HOST,
36
37
        port=PORT,
        loglevel=LOGLEVEL,
38
        debugformat=DEBUGFORMAT,
39
40
41
    print handler
42
43
    # setup the arguments for the handler method
44
    kwargs = \{\}
45
    kwargs["objtype"] = u'user'
46
    kwargs["id"] = 1
47
48
    # call the handler with the get method, passing in kwargs for arguments
49
    response = handler.get(**kwargs)
50
51
52
    print "Type of response: ", type(response)
53
54
    print ""
55
    print "print of response:"
    print response
57
    print ""
59
    print "length of response (number of objects returned): "
60
    print len(response)
61
    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
    print out
```

```
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):
10
    print the first object returned in JSON format:
11
12
      "_type": "user",
13
      "deleted_flag": 0,
14
      "group_id": 0,
15
      "id": 1,
16
      "last_login": "2015-08-07T13:21:59",
17
      "local_admin_flag": -1,
18
      "name": "Jim Olsen",
19
      "permissions": {
20
        "_type": "permissions",
21
         "permission": [
22
           "admin",
23
           "sensor_read",
24
           "sensor_write",
25
           "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
3
    # Determine our script name, script dir
5
    my_file = os.path.abspath(sys.argv[0])
6
    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)
10
    pytan_root_dir = os.path.dirname(parent_dir)
11
    lib_dir = os.path.join(pytan_root_dir, 'lib')
12
    path_adds = [lib_dir]
13
14
15
    for aa in path_adds:
        if aa not in sys.path:
16
            sys.path.append(aa)
17
18
19
    # connection info for Tanium Server
20
    USERNAME = "Tanium User"
21
    PASSWORD = "T@n!um"
22
    HOST = "172.16.31.128"
23
   PORT = "443"
24
25
26
    # Logging conrols
   LOGLEVEL = 2
27
   DEBUGFORMAT = False
```

```
29
    import tempfile
30
31
    import pytan
32
33
    handler = pytan.Handler(
        username=USERNAME,
34
        password=PASSWORD,
35
        host=HOST,
36
        port=PORT,
37
        loglevel=LOGLEVEL,
38
        debugformat=DEBUGFORMAT,
39
40
41
    print handler
42
43
    # setup the arguments for the handler method
44
    kwargs = {}
45
    kwargs["objtype"] = u'sensor'
46
    kwargs["name"] = u'Computer Name'
47
48
    # call the handler with the get method, passing in kwargs for arguments
49
    response = handler.get(**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): "
    print len(response)
61
    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
67
        out = out.splitlines()[0:15]
        out.append('..trimmed for brevity..')
68
        out = ' \ n'. join (out)
69
70
    print out
71
```

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

```
12
      "_type": "sensor",
13
      "category": "Reserved",
14
      "description": "The assigned name of the client machine.\nExample: workstation-1.company.com",
15
      "exclude_from_parse_flag": 0,
      "hash": 3409330187,
17
      "hidden_flag": 0,
18
      "id": 3,
19
      "ignore_case_flag": 1,
20
      "max_age_seconds": 86400,
21
      "name": "Computer Name",
22
      "queries": {
23
        "_type": "queries",
24
        "query": [
25
26
    ..trimmed for brevity..
27
```

Get saved action by name

Get a saved action by name

Example Python Code

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

```
handler = pytan.Handler(
33
        username=USERNAME,
34
        password=PASSWORD,
35
        host=HOST,
36
        port=PORT,
37
        loglevel=LOGLEVEL,
38
        debugformat=DEBUGFORMAT,
39
40
41
   print handler
42
43
44
    # setup the arguments for the handler method
45
    kwargs = {}
    kwargs["objtype"] = u'saved_action'
46
    kwargs["name"] = u'Distribute Tanium Standard Utilities'
47
48
    # call the handler with the get method, passing in kwargs for arguments
49
    response = handler.get(**kwargs)
50
51
   print ""
52
   print "Type of response: ", type(response)
53
54
   print ""
55
   print "print of response:"
56
   print response
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:"
    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
```

```
Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2
   Type of response: <class 'taniumpy.object_types.saved_action_list.SavedActionList'>
3
4
   print of response:
5
6
   SavedActionList, len: 1
   length of response (number of objects returned):
10
   print the first object returned in JSON format:
11
12
      "_type": "saved_action",
13
      "action_group_id": 0,
14
      "approved_flag": 1,
15
```

```
"approver": {
16
        "_type": "user",
17
        "id": 1
18
19
      "comment": "Distribute Tanium Standard Utilities",
20
      "creation_time": "2015-08-07T13:22:26",
21
      "distribute_seconds": 3200,
22
      "end_time": "Never",
23
      "expire_seconds": 3300,
24
      "id": 1,
25
      "issue_count": 0,
26
    ..trimmed for brevity..
```

Get all users

Get all users

Example Python Code

```
import os
    import sys
2
    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)
10
    pytan_root_dir = os.path.dirname(parent_dir)
11
    lib_dir = os.path.join(pytan_root_dir, 'lib')
12
    path_adds = [lib_dir]
13
14
    for aa in path_adds:
15
        if aa not in sys.path:
            sys.path.append(aa)
17
18
19
    # connection info for Tanium Server
20
    USERNAME = "Tanium User"
21
   PASSWORD = "T@n!um"
22
    HOST = "172.16.31.128"
23
    PORT = "443"
24
25
    # Logging conrols
26
    LOGLEVEL = 2
27
   DEBUGFORMAT = False
28
29
   import tempfile
31
   import pytan
32
33
   handler = pytan.Handler(
34
        username=USERNAME,
        password=PASSWORD,
35
        host=HOST,
```

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

```
Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2
   Type of response: <class 'taniumpy.object_types.user_list.UserList'>
3
4
   print of response:
   UserList, len: 6
    length of response (number of objects returned):
8
10
   print the first object returned in JSON format:
11
12
13
      "_type": "user",
      "deleted_flag": 0,
14
      "group_id": 0,
15
      "id": 1,
16
      "last_login": "2015-08-07T13:21:59",
17
      "local_admin_flag": -1,
18
      "name": "Jim Olsen",
19
      "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
3
    # Determine our script name, script dir
5
   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)
10
   pytan_root_dir = os.path.dirname(parent_dir)
11
    lib_dir = os.path.join(pytan_root_dir, 'lib')
12
   path_adds = [lib_dir]
13
14
    for aa in path_adds:
15
        if aa not in sys.path:
16
            sys.path.append(aa)
17
18
19
    # connection info for Tanium Server
20
21
    USERNAME = "Tanium User"
    PASSWORD = "T@n!um"
23
    HOST = "172.16.31.128"
    PORT = "443"
24
25
26
    # Logging conrols
    LOGLEVEL = 2
27
   DEBUGFORMAT = False
28
29
    import tempfile
30
31
    import pytan
32
   handler = pytan.Handler(
33
34
        username=USERNAME,
35
        password=PASSWORD,
        host=HOST,
36
        port=PORT,
37
        loglevel=LOGLEVEL,
38
39
        debugformat=DEBUGFORMAT,
40
41
```

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

```
Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
1
2
3
    Type of response: <class 'taniumpy.object_types.saved_action_list.SavedActionList'>
    print of response:
    SavedActionList, len: 4
6
    length of response (number of objects returned):
8
10
    print the first object returned in JSON format:
11
12
      "_type": "saved_action",
13
      "action_group_id": 0,
14
      "approved_flag": 1,
15
      "approver": {
16
        "_type": "user",
17
        "id": 1
18
19
      "cache_row_id": 0,
20
      "comment": "Distribute Tanium Standard Utilities",
21
      "creation_time": "2015-08-07T13:22:26",
22
      "distribute_seconds": 3200,
23
      "end_time": "Never",
24
      "expire_seconds": 3300,
```

```
"id": 1,
..trimmed for brevity..
```

Get all settings

Get all system settings

Example Python Code

```
import os
    import sys
2
    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
9
    parent_dir = os.path.dirname(my_dir)
10
    pytan_root_dir = os.path.dirname(parent_dir)
11
    lib_dir = os.path.join(pytan_root_dir, 'lib')
12
13
   path_adds = [lib_dir]
15
   for aa in path_adds:
        if aa not in sys.path:
16
            sys.path.append(aa)
17
18
19
    # connection info for Tanium Server
20
    USERNAME = "Tanium User"
21
    PASSWORD = "T@n!um"
22
    HOST = "172.16.31.128"
23
    PORT = "443"
24
25
    # Logging conrols
27
    LOGLEVEL = 2
28
   DEBUGFORMAT = False
29
    import tempfile
30
31
32
    import pytan
33
    handler = pytan.Handler(
34
        username=USERNAME,
        password=PASSWORD,
35
        host=HOST,
36
        port=PORT,
37
38
        loglevel=LOGLEVEL,
        debugformat=DEBUGFORMAT,
41
   print handler
42.
43
44
    # setup the arguments for the handler method
45
   kwargs = {}
   kwargs["objtype"] = u'setting'
```

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

```
Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
1
2
    Type of response: <class 'taniumpy.object_types.system_setting_list.SystemSettingList'>
3
    print of response:
    SystemSettingList, len: 91
    length of response (number of objects returned):
10
    print the first object returned in JSON format:
11
12
      "_type": "system_setting",
13
      "audit_data": {
14
        "_type": "audit_data",
15
        "creation_time": "2015-08-07T13:22:35",
16
        "last_modified_by": "Jim Olsen",
17
        "modification_time": "2015-08-07T13:22:35"
18
      "cache_row_id": 0,
20
      "default_value": "0",
21
      "hidden_flag": 0,
22
      "id": 1,
23
      "name": "load_initial_content",
24
25
      "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)
8
    # determine the pytan lib dir and add it to the path
    parent_dir = os.path.dirname(my_dir)
10
    pytan_root_dir = os.path.dirname(parent_dir)
11
    lib_dir = os.path.join(pytan_root_dir, 'lib')
12
    path_adds = [lib_dir]
13
14
    for aa in path_adds:
15
        if aa not in sys.path:
16
            sys.path.append(aa)
17
18
19
    # connection info for Tanium Server
20
    USERNAME = "Tanium User"
21
    PASSWORD = "T@n!um"
22
    HOST = "172.16.31.128"
23
    PORT = "443"
24
25
    # Logging conrols
26
    LOGLEVEL = 2
27
   DEBUGFORMAT = False
28
29
   import tempfile
30
31
    import pytan
32
    handler = pytan.Handler(
33
        username=USERNAME,
34
35
        password=PASSWORD,
        host=HOST,
36
        port=PORT,
37
        loglevel=LOGLEVEL,
38
        debugformat=DEBUGFORMAT,
39
40
41
   print handler
42
43
    # setup the arguments for the handler method
44
    kwarqs = {}
45
    kwargs["objtype"] = u'saved_question'
46
    # call the handler with the get_all method, passing in kwargs for arguments
48
    response = handler.get_all(**kwargs)
49
50
   print ""
51
   print "Type of response: ", type(response)
```

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

```
Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2
    Type of response: <class 'taniumpy.object_types.saved_question_list.SavedQuestionList'>
3
    print of response:
    SavedQuestionList, len: 107
6
    length of response (number of objects returned):
8
10
    print the first object returned in JSON format:
11
12
      "_type": "saved_question",
13
      "action_tracking_flag": 0,
14
      "archive_enabled_flag": 0,
15
      "archive_owner": {
16
        "_type": "user"
17
19
      "cache_row_id": 0,
      "expire_seconds": 600,
20
      "hidden_flag": 0,
21
      "id": 1,
22
      "issue_seconds": 120,
23
      "issue_seconds_never_flag": 0,
24
      "keep_seconds": 0,
      "mod_time": "2015-08-07T13:22:22",
26
    ..trimmed for brevity..
```

Get all userroless

Get all user roles

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

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

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

Get all questions

Get all questions

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

# Determine our script name, script dir
```

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

```
Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2
    Type of response: <class 'taniumpy.object_types.question_list.QuestionList'>
    print of response:
5
    QuestionList, len: 174
6
    length of response (number of objects returned):
    174
10
    print the first object returned in JSON format:
11
12
      "_type": "question",
13
      "action_tracking_flag": 0,
14
      "cache_row_id": 1,
15
      "context_group": {
16
        "_type": "group",
17
        "id": 0
18
19
      },
      "expiration": "2015-08-07T13:32:29",
20
      "expire_seconds": 600,
21
      "hidden_flag": 0,
22
      "id": 104,
23
      "management_rights_group": {
24
        "_type": "group",
25
        "id": 0
26
    ..trimmed for brevity..
```

Get all groups

Get all groups

```
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)
11
    lib_dir = os.path.join(pytan_root_dir, 'lib')
12
    path_adds = [lib_dir]
13
14
    for aa in path_adds:
15
        if aa not in sys.path:
             sys.path.append(aa)
17
18
19
    # connection info for Tanium Server
20
    USERNAME = "Tanium User"
21
    PASSWORD = "T@n!um"
22
    HOST = "172.16.31.128"
23
    PORT = "443"
24
25
    # Logging conrols
26
    LOGLEVEL = 2
27
    DEBUGFORMAT = False
28
29
    import tempfile
30
31
    import pytan
32
    handler = pytan.Handler(
33
        username=USERNAME,
34
35
        password=PASSWORD,
        host=HOST,
36
        port=PORT,
37
        loglevel=LOGLEVEL,
38
        debugformat=DEBUGFORMAT,
39
40
41
    print handler
42
43
    # setup the arguments for the handler method
44
    kwarqs = \{\}
45
    kwargs["objtype"] = u'group'
46
47
    # call the handler with the get_all method, passing in kwargs for arguments
48
49
    response = handler.get_all(**kwargs)
50
51
    print "Type of response: ", type(response)
52
53
   print ""
54
    print "print of response:"
55
   print response
56
   print ""
58
    print "length of response (number of objects returned): "
59
    print len(response)
60
61
    print ""
62
    print "print the first object returned in JSON format:"
63
    out = response.to_json(response[0])
64
    if len(out.splitlines()) > 15:
65
        out = out.splitlines()[0:15]
66
        out.append('..trimmed for brevity..')
67
        out = '\n'.join(out)
```

```
69
70 print out
```

```
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):
10
    print the first object returned in JSON format:
11
12
      "_type": "group",
13
      "and_flag": 0,
14
      "deleted_flag": 0,
15
      "filters": {
16
        "_type": "filters",
17
        "filter": []
18
19
      "id": 64,
20
      "name": "All Computers",
21
      "not_flag": 0,
22
      "sub_groups": {
23
        "_type": "groups",
24
        "group": []
25
26
      },
    ..trimmed for brevity..
```

Get all sensors

Get all sensors

```
import os
   import sys
   sys.dont_write_bytecode = True
   # Determine our script name, script dir
   my_file = os.path.abspath(sys.argv[0])
6
   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)
10
   pytan_root_dir = os.path.dirname(parent_dir)
11
   lib_dir = os.path.join(pytan_root_dir, 'lib')
12
13
   path_adds = [lib_dir]
14
   for aa in path_adds:
15
```

```
if aa not in sys.path:
16
             sys.path.append(aa)
17
18
19
    # connection info for Tanium Server
20
    USERNAME = "Tanium User"
21
    PASSWORD = "T@n!um"
22
    HOST = "172.16.31.128"
23
    PORT = "443"
24
25
    # Logging conrols
26
    LOGLEVEL = 2
27
    DEBUGFORMAT = False
28
29
    import tempfile
30
31
    import pytan
32
    handler = pytan.Handler(
33
34
        username=USERNAME,
        password=PASSWORD,
35
        host=HOST,
36
        port=PORT,
37
        loglevel=LOGLEVEL,
38
        debugformat=DEBUGFORMAT,
39
40
41
    print handler
42
43
    # setup the arguments for the handler method
44
    kwargs = {}
45
    kwargs["objtype"] = u'sensor'
46
47
    # call the handler with the get_all method, passing in kwargs for arguments
48
    response = handler.get_all(**kwargs)
49
50
    print ""
51
    print "Type of response: ", type(response)
52
    print ""
54
    print "print of response:"
55
    print response
56
57
   print ""
58
    print "length of response (number of objects returned): "
59
60
   print len(response)
61
   print ""
62
   print "print the first object returned in JSON format:"
63
    out = response.to_json(response[0])
64
    if len(out.splitlines()) > 15:
65
66
        out = out.splitlines()[0:15]
67
        out.append('..trimmed for brevity..')
        out = '\n'.join(out)
68
69
    print out
70
```

```
Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2
    Type of response: <class 'taniumpy.object_types.sensor_list.SensorList'>
    print of response:
5
    SensorList, len: 419
6
    length of response (number of objects returned):
8
    419
10
   print the first object returned in JSON format:
11
12
      "_type": "sensor",
13
      "cache_row_id": 0,
14
      "category": "Reserved",
15
      "description": "The recorded state of each action a client has taken recently in the form of id:st
      "exclude_from_parse_flag": 1,
17
      "hash": 1792443391,
18
      "hidden_flag": 0,
19
      "id": 1,
20
      "ignore_case_flag": 1,
21
      "max_age_seconds": 3600,
22
      "name": "Action Statuses",
23
      "queries": {
        "_type": "queries",
25
        "query": [
26
    ..trimmed for brevity..
```

Get all whitelisted urls

Get all whitelisted urls

```
import os
   import sys
2
   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)
10
   pytan_root_dir = os.path.dirname(parent_dir)
11
   lib_dir = os.path.join(pytan_root_dir, 'lib')
12
   path_adds = [lib_dir]
13
14
   for aa in path_adds:
15
        if aa not in sys.path:
16
            sys.path.append(aa)
17
18
19
    # connection info for Tanium Server
20
   USERNAME = "Tanium User"
```

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

```
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
6
    length of response (number of objects returned):
9
10
   print the first object returned in JSON format:
11
12
      "_type": "white_listed_url",
13
      "download_seconds": 86400,
14
      "id": 1,
15
      "url_regex": "test1"
16
```

Get all clients

Get all clients

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

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

```
Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2
   Type of response: <class 'taniumpy.object_types.system_status_list.SystemStatusList'>
3
4
   print of response:
   SystemStatusList, len: 2
   length of response (number of objects returned):
8
10
   print the first object returned in JSON format:
11
12
13
      "_type": "client_status",
      "cache_row_id": 0,
14
      "computer_id": "3888017885",
15
      "full version": "5.1.314.7724",
16
      "host_name": "Casus-Belli.local",
17
      "ipaddress_client": "172.16.31.1",
18
      "ipaddress_server": "172.16.31.1",
19
      "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

```
import os
    import sys
2
    sys.dont_write_bytecode = True
    # Determine our script name, script dir
5
   my_file = os.path.abspath(sys.argv[0])
6
   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)
11
    lib_dir = os.path.join(pytan_root_dir, 'lib')
12
   path_adds = [lib_dir]
13
14
   for aa in path_adds:
15
        if aa not in sys.path:
16
            sys.path.append(aa)
17
18
19
    # connection info for Tanium Server
20
    USERNAME = "Tanium User"
21
    PASSWORD = "T@n!um"
22
    HOST = "172.16.31.128"
   PORT = "443"
24
2.5
    # Logging conrols
26
    I_iOGI_iEVEI_i = 2
27
   DEBUGFORMAT = False
28
29
    import tempfile
30
31
    import pytan
32
   handler = pytan.Handler(
33
34
        username=USERNAME,
35
        password=PASSWORD,
36
        host=HOST,
        port=PORT,
37
        loglevel=LOGLEVEL,
38
        debugformat=DEBUGFORMAT,
39
40
41
   print handler
```

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

```
Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
1
2
    Type of response: <class 'taniumpy.object_types.package_spec_list.PackageSpecList'>
3
    print of response:
   PackageSpecList, len: 72
    length of response (number of objects returned):
10
11
    print the first object returned in JSON format:
12
13
      "_type": "package_spec",
      "available_time": "2015-08-07T13:22:50",
14
      "cache row id": 0,
15
      "command": "cmd /c cscript //T:900 java-installer.vbs /KillAppsUsingJava:Yes /Reboot fNeeded:Yes /
16
17
      "command_timeout": 900,
      "creation_time": "2015-08-07T13:22:16",
18
      "deleted_flag": 0,
      "display_name": "Update Java 64-bit - Kill / Reboot",
20
      "expire_seconds": 1500,
21
      "files": {
22
        "_type": "package_files",
23
        "file": [
24
25
            "_type": "file",
```

```
27 ..trimmed for brevity..
```

Get all actions

Get all actions

```
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
9
    parent_dir = os.path.dirname(my_dir)
10
    pytan_root_dir = os.path.dirname(parent_dir)
11
    lib_dir = os.path.join(pytan_root_dir, 'lib')
12
   path_adds = [lib_dir]
13
14
    for aa in path_adds:
15
        if aa not in sys.path:
16
            sys.path.append(aa)
17
18
19
    # connection info for Tanium Server
20
    USERNAME = "Tanium User"
21
    PASSWORD = "T@n!um"
22
    HOST = "172.16.31.128"
23
    PORT = "443"
24
25
    # Logging conrols
26
27
    LOGLEVEL = 2
   DEBUGFORMAT = False
28
29
   import tempfile
30
31
32
    import pytan
33
    handler = pytan.Handler(
34
        username=USERNAME,
        password=PASSWORD,
35
        host=HOST,
36
        port=PORT,
37
        loglevel=LOGLEVEL,
38
        debugformat=DEBUGFORMAT,
39
40
41
   print handler
42
43
44
    # setup the arguments for the handler method
   kwargs = {}
45
   kwargs["objtype"] = u'action'
46
47
```

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

```
Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2
    Type of response: <class 'taniumpy.object_types.action_list.ActionList'>
3
    print of response:
5
    ActionList, len: 38
6
    length of response (number of objects returned):
10
    print the first object returned in JSON format:
11
12
      "_type": "action",
13
      "action_group": {
14
        "_type": "group",
15
        "id": 0,
16
        "name": "Default"
17
18
      },
      "approver": {
19
        "_type": "user",
20
        "id": 2,
21
        "name": "Tanium User"
22
23
      "cache_row_id": 0,
24
      "comment": "Distribute Tanium Standard Utilities",
25
      "creation_time": "2015-08-07T13:26:19",
26
    ..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])
6
   my_dir = os.path.dirname(my_file)
9
    # determine the pytan lib dir and add it to the path
    parent_dir = os.path.dirname(my_dir)
10
    pytan_root_dir = os.path.dirname(parent_dir)
11
    lib_dir = os.path.join(pytan_root_dir, 'lib')
12
   path_adds = [lib_dir]
13
14
    for aa in path_adds:
15
        if aa not in sys.path:
16
            sys.path.append(aa)
17
18
19
    # connection info for Tanium Server
20
    USERNAME = "Tanium User"
21
    PASSWORD = "T@n!um"
22
    HOST = "172.16.31.128"
23
    PORT = "443"
24
25
    # Logging conrols
26
    LOGLEVEL = 2
27
   DEBUGFORMAT = False
28
29
   import tempfile
30
31
    import pytan
32
   handler = pytan.Handler(
33
34
        username=USERNAME,
35
        password=PASSWORD,
36
        host=HOST,
        port=PORT,
37
        loglevel=LOGLEVEL,
38
        debugformat=DEBUGFORMAT,
39
40
41
   print handler
42
43
    # setup the arguments for the handler method
44
   kwargs = {}
45
    kwargs["objtype"] = u'action'
46
    kwargs["name"] = u'Distribute Tanium Standard Utilities'
47
```

```
# 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)
```

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

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

```
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)
```

```
# determine the pytan lib dir and add it to the path
    parent_dir = os.path.dirname(my_dir)
10
    pytan_root_dir = os.path.dirname(parent_dir)
11
    lib_dir = os.path.join(pytan_root_dir, 'lib')
12
    path_adds = [lib_dir]
13
14
    for aa in path_adds:
15
        if aa not in sys.path:
16
             sys.path.append(aa)
17
18
19
    # connection info for Tanium Server
20
    USERNAME = "Tanium User"
21
    PASSWORD = "T@n!um"
22
    HOST = "172.16.31.128"
23
    PORT = "443"
24
25
    # Logging conrols
26
    LOGLEVEL = 2
27
    DEBUGFORMAT = False
28
29
    import tempfile
30
31
    import pytan
32
33
    handler = pytan.Handler(
        username=USERNAME,
34
        password=PASSWORD,
35
        host=HOST,
36
        port=PORT,
37
        loglevel=LOGLEVEL,
38
        debugformat=DEBUGFORMAT,
39
40
41
    print handler
42
43
    # setup the arguments for the handler method
44
    kwargs = {}
45
    kwargs["run"] = True
    kwargs["package"] = u'Distribute Tanium Standard Utilities'
47
48
    # call the handler with the deploy_action method, passing in kwargs for arguments
49
    response = handler.deploy_action(**kwargs)
50
    import pprint, io
51
52
    print ""
53
    print "Type of response: ", type(response)
55
    print ""
56
    print "Pretty print of response:"
57
    print pprint.pformat(response)
58
    print ""
60
    print "Print of action object: "
61
    print response['action_object']
62
63
    # 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()
```

```
Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
   2015-08-07 19:45:08,345 DEBUG
                                     pytan.handler.ActionPoller: ID 56: id resolved to 56
   2015-08-07 19:45:08,345 DEBUG
                                     pytan.handler.ActionPoller: ID 56: package_spec resolved to Package
   2015-08-07 19:45:08,353 DEBUG
                                     pytan.handler.ActionPoller: ID 56: target_group resolved to Group,
    2015-08-07 19:45:08,353 DEBUG
                                     pytan.handler.ActionPoller: ID 56: Result Map resolved to {'failed'
    2015-08-07 19:45:08,353 DEBUG
                                     pytan.handler.ActionPoller: ID 56: expiration_time resolved to 2015
    2015-08-07 19:45:08,353 DEBUG
                                     pytan.handler.ActionPoller: ID 56: status resolved to Open
7
    2015-08-07 19:45:08,353 DEBUG
                                     pytan.handler.ActionPoller: ID 56: stopped_flag resolved to 0
    2015-08-07 19:45:08,353 DEBUG
                                     pytan.handler.ActionPoller: ID 56: Object Info resolved to ID 56: F
Q
    2015-08-07 19:45:08,353 DEBUG
                                     pytan.handler.ActionPoller: ID 56: Adding Question to derive passed
10
    2015-08-07 19:45:08,365 DEBUG
                                     pytan.handler.QuestionPoller: ID 1299: id resolved to 1299
11
    2015-08-07 19:45:08,365 DEBUG
                                     pytan.handler.QuestionPoller: ID 1299: expiration resolved to 2015-
12
    2015-08-07 19:45:08,365 DEBUG
                                     pytan.handler.QuestionPoller: ID 1299: query_text resolved to Get r
13
    2015-08-07 19:45:08,365 DEBUG
                                     pytan.handler.QuestionPoller: ID 1299: id resolved to 1299
14
    2015-08-07 19:45:08,365 DEBUG
                                     pytan.handler.QuestionPoller: ID 1299: Object Info resolved to Ques
15
    2015-08-07 19:45:08,368 DEBUG
                                     pytan.handler.QuestionPoller: ID 1299: Progress: Tested: 0, Passed:
16
    2015-08-07 19:45:08,368 DEBUG
                                     pytan.handler.QuestionPoller: ID 1299: Timing: Started: 2015-08-07
17
    2015-08-07 19:45:08,368 INFO
                                     pytan.handler.QuestionPoller: ID 1299: Progress Changed 0% (0 of 2)
18
    2015-08-07 19:45:13,372 DEBUG
                                      pytan.handler.QuestionPoller: ID 1299: Progress: Tested: 1, Passed:
19
    2015-08-07 19:45:13,372 DEBUG
                                      pytan.handler.QuestionPoller: ID 1299: Timing: Started: 2015-08-07
20
    2015-08-07 19:45:13,372 INFO
                                     pytan.handler.QuestionPoller: ID 1299: Progress Changed 50% (1 of 2
21
    2015-08-07 19:45:18,379 DEBUG
                                     pytan.handler.QuestionPoller: ID 1299: Progress: Tested: 2, Passed:
22
                                     pytan.handler.QuestionPoller: ID 1299: Timing: Started: 2015-08-07
23
    2015-08-07 19:45:18,379 DEBUG
24
   2015-08-07 19:45:18,380 INFO
                                     pytan.handler.QuestionPoller: ID 1299: Progress Changed 100% (2 of
   2015-08-07 19:45:18,380 INFO
                                     pytan.handler.QuestionPoller: ID 1299: Reached Threshold of 99% (2
25
   2015-08-07 19:45:18,380 DEBUG
                                     pytan.handler.ActionPoller: ID 56: Passed Count resolved to 2
26
   2015-08-07 19:45:18,390 DEBUG
                                     pytan.handler.ActionPoller: ID 56: Progress: Seen Action: 0, Expect
27
   2015-08-07 19:45:18,390 DEBUG
                                     pytan.handler.ActionPoller: ID 56: Timing: Started: 2015-08-07 19:4
28
    2015-08-07 19:45:18,390 INFO
                                     pytan.handler.ActionPoller: ID 56: Progress Changed for Seen Count
29
   2015-08-07 19:45:18,397 DEBUG
                                     pytan.handler.ActionPoller: ID 56: stopped_flag resolved to 0
30
    2015-08-07 19:45:18,397 DEBUG
                                     pytan.handler.ActionPoller: ID 56: status resolved to Open
31
32
    2015-08-07 19:45:23,411 DEBUG
                                      pytan.handler.ActionPoller: ID 56: Progress: Seen Action: 1, Expect
    2015-08-07 19:45:23,411 DEBUG
                                     pytan.handler.ActionPoller: ID 56: Timing: Started: 2015-08-07 19:4
33
    2015-08-07 19:45:23,411 INFO
                                     pytan.handler.ActionPoller: ID 56: Progress Changed for Seen Count
34
    2015-08-07 19:45:23,417 DEBUG
                                     pytan.handler.ActionPoller: ID 56: stopped_flag resolved to 0
35
36
    2015-08-07 19:45:23,417 DEBUG
                                     pytan.handler.ActionPoller: ID 56: status resolved to Open
    2015-08-07 19:45:28,427 DEBUG
                                     pytan.handler.ActionPoller: ID 56: Progress: Seen Action: 2, Expect
37
                                     pytan.handler.ActionPoller: ID 56: Timing: Started: 2015-08-07 19:4
    2015-08-07 19:45:28,427 DEBUG
    2015-08-07 19:45:28,427 INFO
                                     pytan.handler.ActionPoller: ID 56: Progress Changed for Seen Count
39
   2015-08-07 19:45:28,433 DEBUG
                                     pytan.handler.ActionPoller: ID 56: stopped_flag resolved to 0
40
    2015-08-07 19:45:28,433 DEBUG
                                     pytan.handler.ActionPoller: ID 56: status resolved to Open
41
   2015-08-07 19:45:28,433 INFO
                                     pytan.handler.ActionPoller: ID 56: Reached Threshold for Seen Count
42
   2015-08-07 19:45:28,443 DEBUG
                                     pytan.handler.ActionPoller: ID 56: failed: 0, finished: 2, running:
43
    2015-08-07 19:45:28,443 DEBUG
                                     pytan.handler.ActionPoller: ID 56: Timing: Started: 2015-08-07 19:4
44
   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
46
                                       pytan.handler.ActionPoller: ID 56: status resolved to Open
    2015-08-07 19:45:28,448 DEBUG
47
    2015-08-07 19:45:28,448 INFO
                                       pytan.handler.ActionPoller: ID 56: Reached Threshold for Finished C
48
49
    Type of response: <type 'dict'>
50
51
    Pretty print of response:
52
    {'action_info': <taniumpy.object_types.result_info.ResultInfo object at 0x10c063810>,
53
     'action_object': <taniumpy.object_types.action.Action_object_at_0x10c063a50>,
54
     'action_result_map': {'failed': {'56:Expired.': [],
55
                                        '56:Failed.': [],
56
                                        '56:NotSucceeded.': [],
57
                                        '56:Stopped.': [],
58
                                        'total': 0},
59
                             'finished': {'56:Completed.': ['Casus-Belli.local',
60
                                                              'JTANIUM1.localdomain'],
61
                                          '56:Expired.': [],
62
                                          '56:Failed.': [],
63
                                          '56:NotSucceeded.': [],
                                          '56:Stopped.': [],
65
                                          '56:Succeeded.': [],
66
                                          '56: Verified.': [],
67
                                          'total': 2},
68
                             'running': {'56:Copying.': [],
69
                                         '56:Downloading.': [],
                                         '56:PendingVerification.': [],
71
                                         '56:Running.': [],
72
                                         '56:Waiting.': [],
73
                                         'total': 0},
74
                             'success': {'56:Completed.': ['Casus-Belli.local',
75
                                                             'JTANIUM1.localdomain'],
76
                                         '56:Verified.': [],
77
                                         'total': 2},
78
                             'unknown': {'total': 0}},
79
     'action_results': <taniumpy.object_types.result_set.ResultSet object at 0x11acb14d0>,
80
     'group_object': None,
81
     'package_object': <taniumpy.object_types.package_spec.PackageSpec object at 0x10c03d0∮0>,
82
     'poller_object': <pytan.pollers.ActionPoller object at 0x10c063b10>,
83
     'poller_success': True,
84
     'saved_action_object': <taniumpy.object_types.saved_action.SavedAction object at 0x10\pf69750>}
85
86
    Print of action object:
87
    Action, name: 'API Deploy Distribute Tanium Standard Utilities', id: 56
88
89
    CSV Results of response:
90
   Action Statuses, Computer Name
    56:Completed., Casus-Belli.local
92
   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
2
    sys.dont_write_bytecode = True
    # Determine our script name, script dir
5
    my_file = os.path.abspath(sys.argv[0])
6
    my_dir = os.path.dirname(my_file)
7
    # determine the pytan lib dir and add it to the path
10
    parent_dir = os.path.dirname(my_dir)
   pytan_root_dir = os.path.dirname(parent_dir)
    lib_dir = os.path.join(pytan_root_dir, 'lib')
12
   path_adds = [lib_dir]
13
14
   for aa in path_adds:
15
        if aa not in sys.path:
16
            sys.path.append(aa)
17
18
19
    # connection info for Tanium Server
20
    USERNAME = "Tanium User"
21
    PASSWORD = "T@n!um"
22
    HOST = "172.16.31.128"
23
    PORT = "443"
25
    # Logging conrols
26
    LOGLEVEL = 2
27
   DEBUGFORMAT = False
28
29
    import tempfile
30
31
    import pytan
32
    handler = pytan.Handler(
33
        username=USERNAME,
34
        password=PASSWORD,
35
36
        host=HOST,
37
        port=PORT,
        loglevel=LOGLEVEL,
        debugformat=DEBUGFORMAT,
39
40
41
   print handler
42
43
    # setup the arguments for the handler method
    kwargs = \{\}
45
    kwarqs["get_results"] = False
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
    response = handler.deploy_action(**kwargs)
51
   import pprint, io
52
53
54
   print "Type of response: ", type(response)
55
   print ""
   print "Pretty print of response:"
```

```
print pprint.pformat(response)
59
60
   print ""
61
   print "Print of action object: "
62
   print response['action_object']
63
    # 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()
75
```

```
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
2
   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
8
   2015-08-07 19:45:28,504 DEBUG
                                     pytan.handler.ActionPoller: ID 57: Object Info resolved to ID 57: F
9
10
   Type of response: <type 'dict'>
11
12
   Pretty print of response:
13
   {'action_info': <taniumpy.object_types.result_info.ResultInfo object at 0x10be950d0>,
14
     'action_object': <taniumpy.object_types.action.Action object at 0x10bf70fd0>,
15
     'action_result_map': None,
16
     'action_results': None,
17
     'group_object': None,
18
     'package_object': <taniumpy.object_types.package_spec.PackageSpec object at 0x11aae60$0>,
20
     'poller_object': <pytan.pollers.ActionPoller object at 0x11aae6090>,
     'poller_success': None,
21
     'saved_action_object': <taniumpy.object_types.saved_action.SavedAction object at 0x10¢063a50>}
22
23
   Print of action object:
24
   Action, name: 'API Deploy Distribute Tanium Standard Utilities', id: 57
25
```

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
3
4
    # Determine our script name, script dir
5
   my_file = os.path.abspath(sys.argv[0])
6
   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)
10
    pytan_root_dir = os.path.dirname(parent_dir)
11
    lib_dir = os.path.join(pytan_root_dir, 'lib')
12
    path_adds = [lib_dir]
13
14
15
    for aa in path_adds:
        if aa not in sys.path:
16
            sys.path.append(aa)
17
18
19
    # connection info for Tanium Server
20
    USERNAME = "Tanium User"
21
    PASSWORD = "T@n!um"
22
    HOST = "172.16.31.128"
23
   PORT = "443"
24
25
    # Logging conrols
26
    LOGLEVEL = 2
27
    DEBUGFORMAT = False
28
29
    import tempfile
30
31
    import pytan
32
    handler = pytan.Handler(
33
34
        username=USERNAME,
        password=PASSWORD,
35
        host=HOST,
36
        port=PORT,
37
        loglevel=LOGLEVEL,
38
        debugformat=DEBUGFORMAT,
39
40
41
    print handler
42
43
    # setup the arguments for the handler method
44
    kwargs = {}
45
    kwargs["run"] = True
46
    kwargs["action_filters"] = u'Operating System, that contains:Windows'
47
    kwargs["package"] = u'Distribute Tanium Standard Utilities'
    # 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 ""
    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()
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
73
        print ""
        print "CSV Results of response: "
74
        print out.getvalue()
```

```
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
2
    2015-08-07 19:45:28,561 DEBUG
                                     pytan.handler.ActionPoller: ID 58: package_spec resolved to Package
3
   2015-08-07 19:45:28,567 DEBUG
4
                                     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: F
    2015-08-07 19:45:28,584 DEBUG
                                     pytan.handler.ActionPoller: ID 58: Adding Question to derive passed
10
    2015-08-07 19:45:28,615 DEBUG
                                     pytan.handler.QuestionPoller: ID 1302: id resolved to 1302
11
    2015-08-07 19:45:28,615 DEBUG
                                     pytan.handler.QuestionPoller: ID 1302: expiration resolved to 2015-
12
    2015-08-07 19:45:28,615 DEBUG
                                     pytan.handler.QuestionPoller: ID 1302: query_text resolved to Get r
13
    2015-08-07 19:45:28,615 DEBUG
                                     pytan.handler.QuestionPoller: ID 1302: id resolved to 1302
14
    2015-08-07 19:45:28,615 DEBUG
                                     pytan.handler.QuestionPoller: ID 1302: Object Info resolved to Ques
15
    2015-08-07 19:45:28,618 DEBUG
                                     pytan.handler.QuestionPoller: ID 1302: Progress: Tested: 0, Passed:
16
17
    2015-08-07 19:45:28,618 DEBUG
                                     pytan.handler.QuestionPoller: ID 1302: Timing: Started: 2015-08-07
18
   2015-08-07 19:45:28,618 INFO
                                     pytan.handler.QuestionPoller: ID 1302: Progress Changed 0% (0 of 2)
   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
20
21
   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:
22
   2015-08-07 19:45:38,626 DEBUG
                                     pytan.handler.QuestionPoller: ID 1302: Timing: Started: 2015-08-07
23
   2015-08-07 19:45:38,626 INFO
                                     pytan.handler.QuestionPoller: ID 1302: Progress Changed 100% (2 of
24
    2015-08-07 19:45:38,626 INFO
                                     pytan.handler.QuestionPoller: ID 1302: Reached Threshold of 99% (2
25
26
    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
27
    2015-08-07 19:45:38,638 DEBUG
                                     pytan.handler.ActionPoller: ID 58: Timing: Started: 2015-08-07 19:4
28
    2015-08-07 19:45:38,638 INFO
                                     pytan.handler.ActionPoller: ID 58: Progress Changed for Seen Count
29
30
    2015-08-07 19:45:38,645 DEBUG
                                     pytan.handler.ActionPoller: ID 58: stopped_flag resolved to 0
31
   2015-08-07 19:45:38,645 DEBUG
                                     pytan.handler.ActionPoller: ID 58: status resolved to Open
   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
33
   2015-08-07 19:45:43,667 DEBUG
                                     pytan.handler.ActionPoller: ID 58: stopped_flag resolved to 0
34
   2015-08-07 19:45:43,667 DEBUG
                                     pytan.handler.ActionPoller: ID 58: status resolved to Open
35
   2015-08-07 19:45:48,677 DEBUG
                                     pytan.handler.ActionPoller: ID 58: Progress: Seen Action: 1, Expect
36
   2015-08-07 19:45:48,677 DEBUG
                                     pytan.handler.ActionPoller: ID 58: Timing: Started: 2015-08-07 19:4
37
    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
40
    2015-08-07 19:45:48,684 INFO
                                      pytan.handler.ActionPoller: ID 58: Reached Threshold for Seen Count
41
    2015-08-07 19:45:48,692 DEBUG
                                      pytan.handler.ActionPoller: ID 58: failed: 0, finished: 1, running:
42.
   2015-08-07 19:45:48,692 DEBUG
                                      pytan.handler.ActionPoller: ID 58: Timing: Started: 2015-08-07 19:4
43
   2015-08-07 19:45:48,692 INFO
                                      pytan.handler.ActionPoller: ID 58: Progress Changed for Finished Co
44
   2015-08-07 19:45:48,698 DEBUG
                                      pytan.handler.ActionPoller: ID 58: stopped_flag resolved to 0
45
    2015-08-07 19:45:48,698 DEBUG
                                      pytan.handler.ActionPoller: ID 58: status resolved to Open
46
   2015-08-07 19:45:48,698 INFO
                                      pytan.handler.ActionPoller: ID 58: Reached Threshold for Finished O
47
48
   Type of response: <type 'dict'>
49
50
    Pretty print of response:
51
    {'action_info': <taniumpy.object_types.result_info.ResultInfo object at 0x11aadbcd0>,
52
     'action_object': <taniumpy.object_types.action.Action object at 0x11aae6090>,
53
     'action_result_map': {'failed': {'58:Expired.': [],
54
                                        '58:Failed.': [],
55
                                        '58:NotSucceeded.': [],
56
                                        '58:Stopped.': [],
57
                                        'total': 0},
                            'finished': {'58:Completed.': ['JTANIUM1.localdomain'],
59
                                          '58:Expired.': [],
60
                                          '58:Failed.': [],
61
                                          '58:NotSucceeded.': [],
62.
                                          '58:Stopped.': [],
63
                                          '58:Succeeded.': [],
                                          '58:Verified.': [],
65
                                          'total': 1},
66
                            'running': {'58:Copying.': [],
67
                                         '58:Downloading.': [],
68
                                         '58:PendingVerification.': [],
69
                                         '58:Running.': [],
70
                                         '58:Waiting.': [],
71
                                         'total': 0},
72
                            'success': {'58:Completed.': ['JTANIUM1.localdomain'],
73
                                         '58:Verified.': [],
74
                                         'total': 1},
75
                            'unknown': {'total': 0}},
76
     'action_results': <taniumpy.object_types.result_set.ResultSet object at 0x11ac77c90>,
77
     'group_object': <taniumpy.object_types.group.Group object at 0x10c03db90>,
78
     'package_object': <taniumpy.object_types.package_spec.PackageSpec object at 0x11aae60$0>,
79
     'poller_object': <pytan.pollers.ActionPoller object at 0x10bea6950>,
80
     'poller_success': True,
81
     'saved_action_object': <taniumpy.object_types.saved_action.SavedAction object at 0x11acb14d0>}
82
83
    Print of action object:
84
   Action, name: 'API Deploy Distribute Tanium Standard Utilities', id: 58
85
86
   CSV Results of response:
87
   Action Statuses, Computer Name
88
   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
5
   my_file = os.path.abspath(sys.argv[0])
6
    my_dir = os.path.dirname(my_file)
9
    # determine the pytan lib dir and add it to the path
    parent_dir = os.path.dirname(my_dir)
10
    pytan_root_dir = os.path.dirname(parent_dir)
11
    lib_dir = os.path.join(pytan_root_dir, 'lib')
12
   path_adds = [lib_dir]
13
14
    for aa in path_adds:
15
        if aa not in sys.path:
16
            sys.path.append(aa)
17
18
19
    # connection info for Tanium Server
20
    USERNAME = "Tanium User"
21
    PASSWORD = "T@n!um"
22
    HOST = "172.16.31.128"
23
    PORT = "443"
24
25
    # Logging conrols
26
    LOGLEVEL = 2
27
    DEBUGFORMAT = False
28
29
    import tempfile
30
31
    import pytan
32
    handler = pytan.Handler(
33
34
        username=USERNAME,
35
        password=PASSWORD,
        host=HOST,
36
        port=PORT,
37
        loglevel=LOGLEVEL,
38
        debugformat=DEBUGFORMAT,
39
40
41
   print handler
42
43
    # setup the arguments for the handler method
44
    kwargs = {}
45
    kwargs["run"] = True
46
    kwargs["action_filters"] = u'Operating System, that contains:Windows'
47
    kwargs["package"] = u'Custom Tagging - Add Tags{$1=tag_should_be_added,$2=tag_should_be_ignore}'
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 ""
```

```
print "Type of response: ", type(response)
55
56
   print ""
57
   print "Pretty print of response:"
58
   print pprint.pformat(response)
59
   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()
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()
```

```
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
2
   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,
4
   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: F
    2015-08-07 19:45:48,773 DEBUG
                                     pytan.handler.ActionPoller: ID 59: Adding Question to derive passed
10
    2015-08-07 19:45:48,802 DEBUG
                                     pytan.handler.QuestionPoller: ID 1303: id resolved to 1303
11
12
   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 r
13
   2015-08-07 19:45:48,802 DEBUG
                                     pytan.handler.QuestionPoller: ID 1303: id resolved to 1303
14
   2015-08-07 19:45:48,802 DEBUG
                                     pytan.handler.QuestionPoller: ID 1303: Object Info resolved to Ques
15
   2015-08-07 19:45:48,805 DEBUG
                                     pytan.handler.QuestionPoller: ID 1303: Progress: Tested: 0, Passed:
16
   2015-08-07 19:45:48,805 DEBUG
                                     pytan.handler.QuestionPoller: ID 1303: Timing: Started: 2015-08-07
17
   2015-08-07 19:45:48,805 INFO
                                     pytan.handler.QuestionPoller: ID 1303: Progress Changed 0% (0 of 2)
18
    2015-08-07 19:45:53,809 DEBUG
                                     pytan.handler.QuestionPoller: ID 1303: Progress: Tested: 2, Passed:
19
20
    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
21
    2015-08-07 19:45:53,809 INFO
                                     pytan.handler.OuestionPoller: ID 1303: Reached Threshold of 99% (2)
22
    2015-08-07 19:45:53,809 DEBUG
                                     pytan.handler.ActionPoller: ID 59: Passed Count resolved to 1
23
                                     pytan.handler.ActionPoller: ID 59: Progress: Seen Action: 0, Expect
24
    2015-08-07 19:45:53,819 DEBUG
   2015-08-07 19:45:53,819 DEBUG
                                     pytan.handler.ActionPoller: ID 59: Timing: Started: 2015-08-07 19:4
25
   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
2.7
   2015-08-07 19:45:53,824 DEBUG
                                     pytan.handler.ActionPoller: ID 59: status resolved to Open
28
   2015-08-07 19:45:58,837 DEBUG
                                     pytan.handler.ActionPoller: ID 59: Progress: Seen Action: 1, Expect
29
   2015-08-07 19:45:58,837 DEBUG
                                     pytan.handler.ActionPoller: ID 59: Timing: Started: 2015-08-07 19:4
30
   2015-08-07 19:45:58,837 INFO
                                     pytan.handler.ActionPoller: ID 59: Progress Changed for Seen Count
31
    2015-08-07 19:45:58,843 DEBUG
                                     pytan.handler.ActionPoller: ID 59: stopped_flag resolved to 0
32
   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
34
                                      pytan.handler.ActionPoller: ID 59: failed: 0, finished: 0, running:
    2015-08-07 19:45:58,852 DEBUG
35
    2015-08-07 19:45:58,852 DEBUG
                                      pytan.handler.ActionPoller: ID 59: Timing: Started: 2015-08-07 19:4
36
   2015-08-07 19:45:58,852 INFO
                                      pytan.handler.ActionPoller: ID 59: Progress Changed for Finished Co
37
   2015-08-07 19:45:58,858 DEBUG
                                      pytan.handler.ActionPoller: ID 59: stopped_flag resolved to 0
38
   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:
40
                                      pytan.handler.ActionPoller: ID 59: Timing: Started: 2015-08-07 19:4
   2015-08-07 19:46:03,873 DEBUG
41
   2015-08-07 19:46:03,879 DEBUG
                                      pytan.handler.ActionPoller: ID 59: stopped flag resolved to 0
42
                                      pytan.handler.ActionPoller: ID 59: status resolved to Open
    2015-08-07 19:46:03,879 DEBUG
43
   2015-08-07 19:46:08,892 DEBUG
                                      pytan.handler.ActionPoller: ID 59: failed: 0, finished: 1, running:
44
    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
46
    2015-08-07 19:46:08,898 DEBUG
                                      pytan.handler.ActionPoller: ID 59: stopped_flag resolved to 0
47
    2015-08-07 19:46:08,898 DEBUG
                                      pytan.handler.ActionPoller: ID 59: status resolved to Open
48
    2015-08-07 19:46:08,898 INFO
                                      pytan.handler.ActionPoller: ID 59: Reached Threshold for Finished C
49
50
   Type of response: <type 'dict'>
51
52
    Pretty print of response:
53
    {'action_info': <taniumpy.object_types.result_info.ResultInfo object at 0x10c063090>,
54
     'action_object': <taniumpy.object_types.action.Action object at 0x10c17ecd0>,
55
     'action_result_map': {'failed': {'59:Expired.': [],
56
                                       '59:Failed.': [],
57
                                       '59:NotSucceeded.': [],
                                       '59:Stopped.': [],
59
                                       'total': 0},
60
                            'finished': {'59:Completed.': ['JTANIUM1.localdomain'],
61
                                          '59:Expired.': [],
62
                                         '59:Failed.': [],
63
                                          '59:NotSucceeded.': [],
64
                                         '59:Stopped.': [],
65
                                         '59:Succeeded.': [],
66
                                         '59:Verified.': [],
67
                                         'total': 1},
68
                            'running': {'59:Copying.': [],
69
                                        '59:Downloading.': ['JTANIUM1.localdomain'],
70
                                        '59:PendingVerification.': [],
71
                                        '59:Running.': [],
72
                                        '59:Waiting.': [],
73
                                        'total': 1},
74
                            'success': {'59:Completed.': ['JTANIUM1.localdomain'],
75
                                         '59:Verified.': [],
76
                                        'total': 1},
77
                            'unknown': {'total': 0}},
78
     'action_results': <taniumpy.object_types.result_set.ResultSet object at 0x10c17ec10>,
     'group_object': <taniumpy.object_types.group.Group object at 0x10c03d210>,
80
     'package_object': <taniumpy.object_types.package_spec.PackageSpec object at 0x10bf69dd0>,
81
     'poller_object': <pytan.pollers.ActionPoller object at 0x10c063250>,
82
     'poller_success': True,
83
     'saved_action_object': <taniumpy.object_types.saved_action.SavedAction object at 0x10¢03ddd0>}
84
85
    Print of action object:
86
   Action, name: 'API Deploy Custom Tagging - Add Tags', id: 59
87
88
    CSV Results of response:
89
   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
2
    sys.dont_write_bytecode = True
4
    # 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)
10
    pytan_root_dir = os.path.dirname(parent_dir)
11
    lib_dir = os.path.join(pytan_root_dir, 'lib')
12
    path_adds = [lib_dir]
13
14
    for aa in path_adds:
15
        if aa not in sys.path:
16
             sys.path.append(aa)
17
18
19
    # connection info for Tanium Server
20
    USERNAME = "Tanium User"
21
    PASSWORD = "T@n!um"
22
    HOST = "172.16.31.128"
23
    PORT = "443"
24
25
    # Logging conrols
26
    LOGLEVEL = 2
27
    DEBUGFORMAT = False
28
29
    import tempfile
30
31
    import pytan
32
    handler = pytan.Handler(
33
        username=USERNAME,
        password=PASSWORD,
35
        host=HOST,
36
        port=PORT,
37
        loglevel=LOGLEVEL,
38
        debugformat=DEBUGFORMAT,
39
40
41
42
    print handler
43
    # setup the arguments for the handler method
```

```
kwarqs = \{\}
45
    kwargs['report_dir'] = tempfile.gettempdir()
46
    kwargs["package"] = u'Distribute Tanium Standard Utilities'
47
48
49
    # call the handler with the deploy_action method, passing in kwargs for arguments
50
    # this should throw an exception: pytan.exceptions.RunFalse
51
    import traceback
52
   try:
53
        handler.deploy_action(**kwargs)
54
    except Exception as e:
55
        traceback.print_exc(file=sys.stdout)
```

```
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
2
   2015-08-07 19:46:08,957 DEBUG
                                     pytan.handler.QuestionPoller: ID 1304: expiration resolved to 2015-
3
   2015-08-07 19:46:08,957 DEBUG
                                     pytan.handler.QuestionPoller: ID 1304: query_text resolved to Get C
4
   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:
10
    2015-08-07 19:46:13,969 DEBUG
                                     pytan.handler.QuestionPoller: ID 1304: Timing: Started: 2015-08-07
11
                                     pytan.handler.QuestionPoller: ID 1304: Progress Changed 100% (2 of
12
    2015-08-07 19:46:13,969 INFO
    2015-08-07 19:46:13,969 INFO
                                     pytan.handler.QuestionPoller: ID 1304: Reached Threshold of 99% (2
13
    2015-08-07 19:46:13,974 INFO
                                     pytan.handler: Report file '/var/folders/dk/vjr1r_c53\vx6k6qzp2bbt_c
14
    Traceback (most recent call last):
15
     File "<string>", line 55, in <module>
16
     File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 404, in deploy_action
17
        **kwargs
18
      File "/Users/jolsen/gh/pytan/lib/pytan/utils.py", line 2699, in wrap
19
        ret = f(*args, **kwargs)
20
     File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 1662, in _deploy_action
21
        raise pytan.exceptions.RunFalse(m(report_path, len(result)))
22
   RunFalse: 'Run' is not True!!
23
   View and verify the contents of /var/folders/dk/vjr1r_c53yx6k6gzp2bbt_c40000gn/T/VERIFY_BEFORE_DEPLO
24
    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
9
    parent_dir = os.path.dirname(my_dir)
10
    pytan_root_dir = os.path.dirname(parent_dir)
11
    lib_dir = os.path.join(pytan_root_dir, 'lib')
12
    path_adds = [lib_dir]
13
14
    for aa in path_adds:
15
        if aa not in sys.path:
16
             sys.path.append(aa)
17
18
19
    # connection info for Tanium Server
20
    USERNAME = "Tanium User"
21
    PASSWORD = "T@n!um"
22
    HOST = "172.16.31.128"
23
    PORT = "443"
24
25
    # Logging conrols
26
    LOGLEVEL = 2
27
    DEBUGFORMAT = False
28
29
    import tempfile
30
31
32
    import pytan
33
    handler = pytan.Handler(
        username=USERNAME,
34
        password=PASSWORD,
35
        host=HOST,
36
        port=PORT,
37
        loglevel=LOGLEVEL,
38
        debugformat=DEBUGFORMAT,
39
40
41
    print handler
42
43
    # setup the arguments for the handler method
44
45
    kwargs = \{\}
    kwargs['report_dir'] = tempfile.gettempdir()
46
    kwargs["package_help"] = True
47
48
49
    # call the handler with the deploy_action method, passing in kwargs for arguments
50
    # this should throw an exception: pytan.exceptions.PytanHelp
51
    import traceback
52
    try:
53
        handler.deploy_action(**kwargs)
54
    except Exception as e:
55
        traceback.print_exc(file=sys.stdout)
56
```

```
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_package())
```

```
PytanHelp:
6
    Package Help
7
Q
    Supplying package defines what package will be deployed as part of the
10
11
12
    A package string is a human string that describes, at a minimum, a
13
    package. It can also optionally define a selector for the package,
14
    and/or parameters for the package. A package must be provided as a string.
15
16
17
    Examples for package
18
19
    Supplying a package:
20
21
        'Distribute Tanium Standard Utilities'
22
23
    Supplying a package by id:
24
25
        'id:1'
26
27
    Supplying a package by hash:
28
29
        'hash:123456789'
30
31
    Supplying a package by name:
32
33
        'name:Distribute Tanium Standard Utilities'
34
35
    Package Parameters
36
37
38
    Supplying parameters to a package can control the arguments
39
    that are supplied to a package, if that package takes any arguments.
40
41
    Package parameters must be surrounded with curly braces '{}',
42
    and must have a key and value specified that is separated by
43
44
    an equals '='. Multiple parameters must be seperated by
    a comma ','. The key should match up to a valid parameter key
45
    for the package in question.
46
47
    If a parameter is supplied and the package doesn't have a
48
    corresponding key name, it will be ignored. If the package has
49
    parameters and a parameter is NOT supplied then an exception
50
    will be raised, printing out the JSON of the missing paramater
51
    for the package in question.
52
53
    Examples for package with parameters
54
55
56
57
    Supplying a package with a single parameter '$1':
58
        'Package With Params { $1=value1}'
59
60
    Supplying a package with two parameters, '$1' and '$2':
61
62
```

```
'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
5
    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
9
    parent_dir = os.path.dirname(my_dir)
10
    pytan_root_dir = os.path.dirname(parent_dir)
11
    lib_dir = os.path.join(pytan_root_dir, 'lib')
12
   path_adds = [lib_dir]
13
14
    for aa in path_adds:
15
        if aa not in sys.path:
16
            sys.path.append(aa)
17
18
19
    # connection info for Tanium Server
20
    USERNAME = "Tanium User"
21
    PASSWORD = "T@n!um"
22
    HOST = "172.16.31.128"
23
    PORT = "443"
24
25
    # Logging conrols
26
27
    LOGLEVEL = 2
   DEBUGFORMAT = False
28
29
   import tempfile
30
31
32
    import pytan
33
    handler = pytan.Handler(
34
        username=USERNAME,
        password=PASSWORD,
35
        host=HOST,
36
        port=PORT,
37
        loglevel=LOGLEVEL,
38
39
        debugformat=DEBUGFORMAT,
40
41
   print handler
42
43
    # setup the arguments for the handler method
44
45
   kwargs = \{\}
   kwargs['report_dir'] = tempfile.gettempdir()
46
   kwargs["run"] = True
```

```
kwargs["package"] = u'Invalid Package'
49
50
    # call the handler with the deploy_action method, passing in kwargs for arguments
51
    # this should throw an exception: pytan.exceptions.HandlerError
52
    import traceback
53
    try:
54
        handler.deploy_action(**kwargs)
55
    except Exception as e:
56
        traceback.print_exc(file=sys.stdout)
```

```
Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
    Traceback (most recent call last):
2
      File "<string>", line 56, in <module>
3
      File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 404, in deploy_action
4
        **kwarqs
5
     File "/Users/jolsen/gh/pytan/lib/pytan/utils.py", line 2699, in wrap
6
        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
10
        d['package_obj'] = self.get('package', **def_search)[0]
11
     File "/Users/jolsen/gh/pytan/lib/pytan/utils.py", line 2699, in wrap
12
        ret = f(*args, **kwargs)
13
14
      File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 1255, in get
        return self._get_single(obj_map, **kwargs)
15
      File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 1390, in _get_single
16
        for x in self._single_find(obj_map, k, v, **kwargs):
17
      File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 1400, in _single_find
18
        obj_ret = self._find(api_obj_single, **kwargs)
19
      File "/Users/jolsen/gh/pytan/lib/pytan/utils.py", line 2699, in wrap
20
        ret = f(*args, **kwargs)
21
      File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 1327, in _find
22
        raise pytan.exceptions.HandlerError(err(search_str))
23
    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)
11
    lib_dir = os.path.join(pytan_root_dir, 'lib')
12
    path_adds = [lib_dir]
13
14
    for aa in path_adds:
15
        if aa not in sys.path:
             sys.path.append(aa)
17
18
19
    # connection info for Tanium Server
20
    USERNAME = "Tanium User"
21
    PASSWORD = "T@n!um"
22
    HOST = "172.16.31.128"
23
    PORT = "443"
24
25
    # Logging conrols
26
    LOGLEVEL = 2
27
    DEBUGFORMAT = False
28
29
    import tempfile
30
31
    import pytan
32
    handler = pytan.Handler(
33
        username=USERNAME,
34
        password=PASSWORD,
35
        host=HOST,
36
        port=PORT,
37
        loglevel=LOGLEVEL,
38
        debugformat=DEBUGFORMAT,
39
40
41
    print handler
42
43
    # setup the arguments for the handler method
44
    kwarqs = \{\}
45
    kwargs['report_dir'] = tempfile.gettempdir()
46
    kwargs["options_help"] = True
47
48
49
    # call the handler with the deploy_action method, passing in kwargs for arguments
50
    # this should throw an exception: pytan.exceptions.PytanHelp
51
    import traceback
52
53
    try:
        handler.deploy_action(**kwargs)
54
    except Exception as e:
55
        traceback.print_exc(file=sys.stdout)
```

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

```
Example for question: "value_type:value_type"
67
68
         'and'
69
             Help: Use 'and' for all of the filters supplied
70
             Usable on: group
71
             Example for sensor: "Sensor1, opt:and"
72
             Example for question: "and"
73
74
         'or'
75
             Help: Use 'or' for all of the filters supplied
76
             Usable on: group
77
             Example for sensor: "Sensor1, opt:or"
78
             Example for question: "or"
79
80
         'ignore_case'
81
             Help: Make the filter do a case insensitive match
82
             Usable on: filter
83
             Example for sensor: "Sensor1, opt:ignore_case"
84
             Example for question: "ignore_case"
85
86
         'match_case'
87
             Help: Make the filter do a case sensitive match
88
             Usable on: filter
89
             Example for sensor: "Sensor1, opt:match_case"
90
             Example for question: "match_case"
91
92
         'match_any_value'
93
             Help: Make the filter match any value
94
             Usable on: filter
95
             Example for sensor: "Sensor1, opt:match_any_value"
96
             Example for question: "match_any_value"
97
98
         'match_all_values'
             Help: Make the filter match all values
100
             Usable on: filter
101
             Example for sensor: "Sensor1, opt:match_all_values"
102
             Example for question: "match_all_values"
103
104
         'max_data_age'
105
             Help: Re-fetch cached values older than N seconds
106
             Usable on: filter
107
             VALUE description and type: seconds, <type 'int'>
108
             Example for sensor: "Sensor1, opt:max_data_age:seconds"
109
             Example for question: "max_data_age:seconds"
110
111
         'value_type'
112
             Help: Make the filter consider the value type as VALUE_TYPE
113
             Usable on: filter
114
             VALUE description and type: value_type, <type 'str'>
115
             Example for sensor: "Sensor1, opt:value_type:value_type"
116
             Example for question: "value_type:value_type"
117
118
         'and'
119
             Help: Use 'and' for all of the filters supplied
120
             Usable on: group
121
             Example for sensor: "Sensor1, opt:and"
122
             Example for question: "and"
123
```

```
'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
5
    my_file = os.path.abspath(sys.argv[0])
6
   my_dir = os.path.dirname(my_file)
7
    # 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)
11
    lib_dir = os.path.join(pytan_root_dir, 'lib')
12
   path_adds = [lib_dir]
13
14
15
    for aa in path_adds:
        if aa not in sys.path:
16
             sys.path.append(aa)
17
18
19
    # connection info for Tanium Server
20
    USERNAME = "Tanium User"
21
    PASSWORD = "T@n!um"
22
    HOST = "172.16.31.128"
23
   PORT = "443"
24
25
    # Logging conrols
26
27
    LOGLEVEL = 2
   DEBUGFORMAT = False
28
29
30
    import tempfile
31
    import pytan
32
    handler = pytan.Handler(
33
        username=USERNAME,
34
35
        password=PASSWORD,
        host=HOST,
36
37
        port=PORT,
        loglevel=LOGLEVEL,
38
        debugformat=DEBUGFORMAT,
39
40
41
   print handler
42
43
```

```
# setup the arguments for the handler method
44
    kwargs = {}
45
    kwargs['report_dir'] = tempfile.gettempdir()
46
    kwargs["run"] = True
47
   kwargs["package"] = u''
48
49
50
    # call the handler with the deploy_action method, passing in kwargs for arguments
51
    # this should throw an exception: pytan.exceptions.HumanParserError
52
    import traceback
53
54
    try:
        handler.deploy_action(**kwargs)
55
    except Exception as e:
56
        traceback.print_exc(file=sys.stdout)
57
```

```
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
2
    sys.dont_write_bytecode = True
3
    # 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)
10
    pytan_root_dir = os.path.dirname(parent_dir)
    lib_dir = os.path.join(pytan_root_dir, 'lib')
12
    path_adds = [lib_dir]
13
14
    for aa in path_adds:
15
        if aa not in sys.path:
16
            sys.path.append(aa)
17
18
19
    # connection info for Tanium Server
20
    USERNAME = "Tanium User"
21
   PASSWORD = "T@n!um"
22
```

```
HOST = "172.16.31.128"
    PORT = "443"
24
25
    # Logging conrols
26
    LOGLEVEL = 2
27
    DEBUGFORMAT = False
28
29
    import tempfile
30
31
    import pytan
32
    handler = pytan.Handler(
33
        username=USERNAME,
34
        password=PASSWORD,
35
        host=HOST,
36
        port=PORT,
37
        loglevel=LOGLEVEL,
38
        debugformat=DEBUGFORMAT,
39
40
41
    print handler
42
43
    # setup the arguments for the handler method
44
    kwargs = {}
45
    kwargs['report_dir'] = tempfile.gettempdir()
46
    kwargs["filters_help"] = True
47
48
49
    # call the handler with the deploy_action method, passing in kwargs for arguments
50
    # this should throw an exception: pytan.exceptions.PytanHelp
51
    import traceback
52
    try:
53
        handler.deploy_action(**kwargs)
54
    except Exception as e:
55
        traceback.print_exc(file=sys.stdout)
```

```
Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
   Traceback (most recent call last):
2
     File "<string>", line 55, in <module>
3
     File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 377, in deploy_action
4
       raise pytan.exceptions.PytanHelp(pytan.help.help_filters())
   PytanHelp:
   Filters Help
    _____
8
   Filters are used generously throughout pytan. When used as part of a
10
   sensor string, they control what data is shown for the columns that
11
   the sensor returns. When filters are used for whole question filters,
12
   they control what rows will be returned. They are used by Groups to
   define group membership, deploy actions to determine which machines
14
   should have the action deployed to it, and more.
15
16
   A filter string is a human string that describes, a sensor followed
17
   by ', that FILTER: VALUE', where FILTER is a valid filter string,
18
   and VALUE is the string that you want FILTER to match on.
19
```

```
Valid Filters
21
22
23
         ' < '
24
             Help: Filter for less than VALUE
25
             Example: "Sensor1, that <: VALUE"
26
27
         'less'
28
             Help: Filter for less than VALUE
29
             Example: "Sensor1, that less:VALUE"
30
31
         '1t'
32
             Help: Filter for less than VALUE
33
             Example: "Sensor1, that lt:VALUE"
34
35
         'less than'
36
             Help: Filter for less than VALUE
37
             Example: "Sensor1, that less than: VALUE"
38
39
         11<1
40
             Help: Filter for not less than VALUE
41
             Example: "Sensor1, that !<:VALUE"
42
43
         'notless'
44
             Help: Filter for not less than VALUE
45
             Example: "Sensor1, that notless: VALUE"
46
47
         'not less'
48
             Help: Filter for not less than VALUE
49
             Example: "Sensor1, that not less:VALUE"
50
51
         'not less than'
52
             Help: Filter for not less than VALUE
53
             Example: "Sensor1, that not less than: VALUE"
54
55
         ' <= '
56
             Help: Filter for less than or equal to VALUE
57
             Example: "Sensor1, that <=: VALUE"
58
59
         'less equal'
60
             Help: Filter for less than or equal to VALUE
61
             Example: "Sensor1, that less equal: VALUE"
62
63
         'lessequal'
64
             Help: Filter for less than or equal to VALUE
65
             Example: "Sensor1, that lessequal: VALUE"
66
67
         'le'
68
             Help: Filter for less than or equal to VALUE
69
             Example: "Sensor1, that le:VALUE"
70
71
         '!<='
72
             Help: Filter for not less than or equal to VALUE
73
             Example: "Sensor1, that !<=:VALUE"
74
75
         'not less equal'
76
             Help: Filter for not less than or equal to VALUE
77
             Example: "Sensor1, that not less equal: VALUE"
```

```
79
         'not lessequal'
80
             Help: Filter for not less than or equal to VALUE
81
             Example: "Sensor1, that not lessequal: VALUE"
82
83
         1 > 1
84
             Help: Filter for greater than VALUE
85
             Example: "Sensor1, that >: VALUE"
86
87
         'greater'
88
             Help: Filter for greater than VALUE
89
             Example: "Sensor1, that greater: VALUE"
91
         'qt'
92
             Help: Filter for greater than VALUE
93
             Example: "Sensor1, that gt:VALUE"
94
95
         'greater than'
             Help: Filter for greater than VALUE
97
             Example: "Sensor1, that greater than: VALUE"
98
99
100
             Help: Filter for not greater than VALUE
101
             Example: "Sensor1, that !>:VALUE"
102
103
         'not greater'
104
             Help: Filter for not greater than VALUE
105
             Example: "Sensor1, that not greater: VALUE"
106
107
         'notgreater'
108
             Help: Filter for not greater than VALUE
109
             Example: "Sensor1, that notgreater: VALUE"
110
111
         'not greater than'
112
             Help: Filter for not greater than VALUE
113
             Example: "Sensor1, that not greater than: VALUE"
114
115
         ' => '
116
117
             Help: Filter for greater than or equal to VALUE
             Example: "Sensor1, that =>: VALUE"
118
119
         'greater equal'
120
             Help: Filter for greater than or equal to VALUE
121
             Example: "Sensor1, that greater equal: VALUE"
122
123
         'greaterequal'
124
             Help: Filter for greater than or equal to VALUE
125
             Example: "Sensor1, that greaterequal:VALUE"
126
127
         'ae'
128
129
             Help: Filter for greater than or equal to VALUE
130
             Example: "Sensor1, that ge:VALUE"
131
         '!=>'
132
             Help: Filter for not greater than VALUE
133
             Example: "Sensor1, that !=>:VALUE"
134
135
         'not greater equal'
```

```
Help: Filter for not greater than VALUE
137
             Example: "Sensor1, that not greater equal: VALUE"
138
139
         'notgreaterequal'
140
             Help: Filter for not greater than VALUE
141
             Example: "Sensor1, that notgreaterequal: VALUE"
142
143
144
             Help: Filter for equals to VALUE
145
             Example: "Sensor1, that =: VALUE"
146
147
         'equal'
148
149
             Help: Filter for equals to VALUE
             Example: "Sensor1, that equal: VALUE"
150
151
         'equals'
152
             Help: Filter for equals to VALUE
153
             Example: "Sensor1, that equals: VALUE"
154
155
         'eq'
156
             Help: Filter for equals to VALUE
157
             Example: "Sensor1, that eq:VALUE"
158
159
160
             Help: Filter for not equals to VALUE
             Example: "Sensor1, that !=: VALUE"
162
163
         'not equal'
164
             Help: Filter for not equals to VALUE
165
             Example: "Sensor1, that not equal: VALUE"
166
167
         'notequal'
168
             Help: Filter for not equals to VALUE
169
             Example: "Sensor1, that notequal: VALUE"
170
171
         'not equals'
172
             Help: Filter for not equals to VALUE
173
174
             Example: "Sensor1, that not equals: VALUE"
175
         'notequals'
176
             Help: Filter for not equals to VALUE
177
             Example: "Sensor1, that notequals: VALUE"
178
179
         'ne'
180
             Help: Filter for not equals to VALUE
181
             Example: "Sensor1, that ne:VALUE"
182
183
         'contains'
184
             Help: Filter for contains VALUE (adds .* before and after VALUE)
185
             Example: "Sensor1, that contains: VALUE"
186
187
         'does not contain'
188
             Help: Filter for does not contain VALUE (adds .* before and after VALUE)
189
             Example: "Sensor1, that does not contain: VALUE"
190
191
         'doesnotcontain'
192
             Help: Filter for does not contain VALUE (adds .* before and after VALUE)
193
             Example: "Sensor1, that doesnotcontain: VALUE"
```

```
195
         'not contains'
196
             Help: Filter for does not contain VALUE (adds .* before and after VALUE)
197
             Example: "Sensor1, that not contains: VALUE"
198
199
         'notcontains'
             Help: Filter for does not contain VALUE (adds .* before and after VALUE)
201
             Example: "Sensor1, that notcontains: VALUE"
202
203
         'starts with'
204
             Help: Filter for starts with VALUE (adds .* after VALUE)
205
             Example: "Sensor1, that starts with: VALUE"
207
         'startswith'
208
             Help: Filter for starts with VALUE (adds .* after VALUE)
209
             Example: "Sensor1, that startswith: VALUE"
210
211
         'does not start with'
212
             Help: Filter for does not start with VALUE (adds .* after VALUE)
213
             Example: "Sensor1, that does not start with: VALUE"
214
215
         'doesnotstartwith'
216
             Help: Filter for does not start with VALUE (adds .* after VALUE)
217
             Example: "Sensor1, that doesnotstartwith: VALUE"
218
219
         'not starts with'
220
             Help: Filter for does not start with VALUE (adds .* after VALUE)
221
             Example: "Sensor1, that not starts with: VALUE"
222
223
         'notstartswith'
224
             Help: Filter for does not start with VALUE (adds .* after VALUE)
225
             Example: "Sensor1, that notstartswith: VALUE"
226
227
         'ends with'
228
             Help: Filter for ends with VALUE (adds .* before VALUE)
229
             Example: "Sensor1, that ends with: VALUE"
230
231
         'endswith'
232
             Help: Filter for ends with VALUE (adds .* before VALUE)
233
             Example: "Sensor1, that endswith: VALUE"
234
235
         'does not end with'
236
             Help: Filter for does bit end with VALUE (adds .* before VALUE)
237
             Example: "Sensor1, that does not end with: VALUE"
238
239
         'doesnotendwith'
240
             Help: Filter for does bit end with VALUE (adds .* before VALUE)
241
             Example: "Sensor1, that doesnotendwith: VALUE"
242
243
         'not ends with'
244
             Help: Filter for does bit end with VALUE (adds .* before VALUE)
245
             Example: "Sensor1, that not ends with: VALUE"
246
247
         'notstartswith'
248
             Help: Filter for does bit end with VALUE (adds .* before VALUE)
249
             Example: "Sensor1, that notstartswith: VALUE"
250
251
         'is not'
252
```

```
Help: Filter for non regular expression match for VALUE
253
             Example: "Sensor1, that is not: VALUE"
254
255
         'not regex'
256
             Help: Filter for non regular expression match for VALUE
257
             Example: "Sensor1, that not regex: VALUE"
258
259
         'notregex'
260
             Help: Filter for non regular expression match for VALUE
261
             Example: "Sensor1, that notregex: VALUE"
262
263
         'not regex match'
             Help: Filter for non regular expression match for VALUE
265
             Example: "Sensor1, that not regex match: VALUE"
266
267
         'notregexmatch'
268
             Help: Filter for non regular expression match for VALUE
269
             Example: "Sensor1, that notregexmatch: VALUE"
270
271
         'nre'
272
             Help: Filter for non regular expression match for VALUE
273
             Example: "Sensor1, that nre: VALUE"
274
275
276
             Help: Filter for regular expression match for VALUE
             Example: "Sensor1, that is: VALUE"
278
279
         'regex'
280
             Help: Filter for regular expression match for VALUE
281
             Example: "Sensor1, that regex: VALUE"
282
283
         'regex match'
284
             Help: Filter for regular expression match for VALUE
285
             Example: "Sensor1, that regex match: VALUE"
286
287
         'regexmatch'
288
             Help: Filter for regular expression match for VALUE
289
             Example: "Sensor1, that regexmatch: VALUE"
290
291
         're'
292
             Help: Filter for regular expression match for VALUE
293
             Example: "Sensor1, that re:VALUE"
294
295
         ' < '
296
             Help: Filter for less than VALUE
297
             Example: "Sensor1, that <: VALUE"
298
299
         'less'
300
             Help: Filter for less than VALUE
301
             Example: "Sensor1, that less:VALUE"
302
         '1t'
             Help: Filter for less than VALUE
305
             Example: "Sensor1, that lt:VALUE"
306
307
         'less than'
308
             Help: Filter for less than VALUE
309
             Example: "Sensor1, that less than: VALUE"
```

```
311
         '!<'
312
             Help: Filter for not less than VALUE
313
             Example: "Sensor1, that !<: VALUE"
314
315
         'notless'
             Help: Filter for not less than VALUE
317
             Example: "Sensor1, that notless: VALUE"
318
319
         'not less'
320
             Help: Filter for not less than VALUE
321
             Example: "Sensor1, that not less: VALUE"
322
323
         'not less than'
324
             Help: Filter for not less than VALUE
325
             Example: "Sensor1, that not less than: VALUE"
326
327
         ' <= '
328
             Help: Filter for less than or equal to VALUE
329
             Example: "Sensor1, that <=: VALUE"
330
331
         'less equal'
332
             Help: Filter for less than or equal to VALUE
333
             Example: "Sensor1, that less equal: VALUE"
334
         'lessequal'
336
             Help: Filter for less than or equal to VALUE
337
             Example: "Sensor1, that lessequal: VALUE"
338
339
         'le'
340
             Help: Filter for less than or equal to VALUE
341
             Example: "Sensor1, that le:VALUE"
342
343
         '!<=!
344
             Help: Filter for not less than or equal to VALUE
345
             Example: "Sensor1, that !<=:VALUE"
346
347
348
         'not less equal'
349
             Help: Filter for not less than or equal to VALUE
             Example: "Sensor1, that not less equal: VALUE"
350
351
         'not lessequal'
352
             Help: Filter for not less than or equal to VALUE
353
             Example: "Sensor1, that not lessequal: VALUE"
354
355
         ' > '
356
             Help: Filter for greater than VALUE
357
             Example: "Sensor1, that >: VALUE"
358
359
         'greater'
360
             Help: Filter for greater than VALUE
             Example: "Sensor1, that greater: VALUE"
363
         'at'
364
             Help: Filter for greater than VALUE
365
             Example: "Sensor1, that gt:VALUE"
366
367
         'greater than'
```

```
Help: Filter for greater than VALUE
369
             Example: "Sensor1, that greater than: VALUE"
370
371
         '!>'
372
             Help: Filter for not greater than VALUE
373
             Example: "Sensor1, that !>: VALUE"
374
375
         'not greater'
376
             Help: Filter for not greater than VALUE
377
             Example: "Sensor1, that not greater: VALUE"
378
379
         'notgreater'
             Help: Filter for not greater than VALUE
381
             Example: "Sensor1, that notgreater: VALUE"
382
383
         'not greater than'
384
             Help: Filter for not greater than VALUE
385
             Example: "Sensor1, that not greater than: VALUE"
386
387
         !=>!
388
             Help: Filter for greater than or equal to VALUE
389
             Example: "Sensor1, that =>:VALUE"
390
391
         'greater equal'
392
             Help: Filter for greater than or equal to VALUE
             Example: "Sensor1, that greater equal: VALUE"
394
395
         'greaterequal'
396
             Help: Filter for greater than or equal to VALUE
397
             Example: "Sensor1, that greaterequal: VALUE"
398
399
         'ge'
400
             Help: Filter for greater than or equal to VALUE
401
             Example: "Sensor1, that ge:VALUE"
402
403
         '!=>'
404
             Help: Filter for not greater than VALUE
405
             Example: "Sensor1, that !=>:VALUE"
406
407
         'not greater equal'
408
             Help: Filter for not greater than VALUE
409
             Example: "Sensor1, that not greater equal: VALUE"
410
411
         'notgreaterequal'
412
             Help: Filter for not greater than VALUE
413
             Example: "Sensor1, that notgreaterequal: VALUE"
414
415
         1 _ 1
416
             Help: Filter for equals to VALUE
417
             Example: "Sensor1, that =: VALUE"
418
419
         'equal'
420
             Help: Filter for equals to VALUE
421
             Example: "Sensor1, that equal: VALUE"
422
423
         'equals'
424
             Help: Filter for equals to VALUE
425
             Example: "Sensor1, that equals: VALUE"
```

```
427
         'eq'
428
             Help: Filter for equals to VALUE
429
             Example: "Sensor1, that eq:VALUE"
430
431
         ' ! = '
432
             Help: Filter for not equals to VALUE
433
             Example: "Sensor1, that !=:VALUE"
434
435
         'not equal'
436
             Help: Filter for not equals to VALUE
437
             Example: "Sensor1, that not equal: VALUE"
439
         'notequal'
440
             Help: Filter for not equals to VALUE
441
             Example: "Sensor1, that notequal: VALUE"
442
443
444
         'not equals'
             Help: Filter for not equals to VALUE
445
             Example: "Sensor1, that not equals: VALUE"
446
447
         'notequals'
448
             Help: Filter for not equals to VALUE
449
             Example: "Sensor1, that notequals: VALUE"
450
451
         'ne'
452
             Help: Filter for not equals to VALUE
453
             Example: "Sensor1, that ne:VALUE"
454
455
         'contains'
456
             Help: Filter for contains VALUE (adds .* before and after VALUE)
457
             Example: "Sensor1, that contains: VALUE"
458
459
         'does not contain'
460
             Help: Filter for does not contain VALUE (adds .* before and after VALUE)
461
             Example: "Sensor1, that does not contain: VALUE"
462
463
         'doesnotcontain'
464
             Help: Filter for does not contain VALUE (adds .* before and after VALUE)
             Example: "Sensor1, that doesnotcontain: VALUE"
466
467
         'not contains'
468
             Help: Filter for does not contain VALUE (adds .* before and after VALUE)
469
             Example: "Sensor1, that not contains: VALUE"
470
471
         'notcontains'
472
             Help: Filter for does not contain VALUE (adds .* before and after VALUE)
473
             Example: "Sensor1, that notcontains: VALUE"
474
475
         'starts with'
476
             Help: Filter for starts with VALUE (adds .* after VALUE)
477
             Example: "Sensor1, that starts with: VALUE"
478
479
         'startswith'
480
             Help: Filter for starts with VALUE (adds .* after VALUE)
481
             Example: "Sensor1, that startswith: VALUE"
482
483
         'does not start with'
```

```
Help: Filter for does not start with VALUE (adds .* after VALUE)
485
             Example: "Sensor1, that does not start with: VALUE"
486
487
         'doesnotstartwith'
488
             Help: Filter for does not start with VALUE (adds .* after VALUE)
489
             Example: "Sensor1, that doesnotstartwith: VALUE"
491
         'not starts with'
492
             Help: Filter for does not start with VALUE (adds .* after VALUE)
493
             Example: "Sensor1, that not starts with: VALUE"
494
495
         'notstartswith'
             Help: Filter for does not start with VALUE (adds .* after VALUE)
497
             Example: "Sensor1, that notstartswith: VALUE"
498
499
         'ends with'
500
             Help: Filter for ends with VALUE (adds .* before VALUE)
501
             Example: "Sensor1, that ends with: VALUE"
502
503
         'endswith'
504
             Help: Filter for ends with VALUE (adds .* before VALUE)
505
             Example: "Sensor1, that endswith: VALUE"
506
507
         'does not end with'
             Help: Filter for does bit end with VALUE (adds .* before VALUE)
             Example: "Sensor1, that does not end with: VALUE"
510
511
         'doesnotendwith'
512
             Help: Filter for does bit end with VALUE (adds .* before VALUE)
513
             Example: "Sensor1, that doesnotendwith: VALUE"
514
515
         'not ends with'
             Help: Filter for does bit end with VALUE (adds .* before VALUE)
517
             Example: "Sensor1, that not ends with: VALUE"
518
519
         'notstartswith'
520
             Help: Filter for does bit end with VALUE (adds .* before VALUE)
521
             Example: "Sensor1, that notstartswith: VALUE"
522
523
         'is not'
524
             Help: Filter for non regular expression match for VALUE
525
             Example: "Sensor1, that is not: VALUE"
526
527
         'not regex'
528
             Help: Filter for non regular expression match for VALUE
529
             Example: "Sensor1, that not regex: VALUE"
530
531
         'notregex'
532
             Help: Filter for non regular expression match for VALUE
533
             Example: "Sensor1, that notregex: VALUE"
534
         'not regex match'
536
             Help: Filter for non regular expression match for VALUE
537
             Example: "Sensor1, that not regex match: VALUE"
538
539
         'notregexmatch'
540
             Help: Filter for non regular expression match for VALUE
541
             Example: "Sensor1, that notregexmatch: VALUE"
```

```
543
         'nre'
544
             Help: Filter for non regular expression match for VALUE
545
             Example: "Sensor1, that nre: VALUE"
546
547
         'is'
             Help: Filter for regular expression match for VALUE
549
             Example: "Sensor1, that is:VALUE"
550
551
         'regex'
552
             Help: Filter for regular expression match for VALUE
553
             Example: "Sensor1, that regex: VALUE"
555
         'regex match'
556
             Help: Filter for regular expression match for VALUE
557
             Example: "Sensor1, that regex match: VALUE"
558
559
         'regexmatch'
560
             Help: Filter for regular expression match for VALUE
             Example: "Sensor1, that regexmatch: VALUE"
562
563
564
             Help: Filter for regular expression match for VALUE
565
             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

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

```
25
    # Logging conrols
26
    LOGLEVEL = 2
27
    DEBUGFORMAT = False
28
29
    import tempfile
30
31
    import pytan
32
    handler = pytan.Handler(
33
        username=USERNAME,
34
        password=PASSWORD,
35
        host=HOST,
36
        port=PORT,
37
        loglevel=LOGLEVEL,
38
        debugformat=DEBUGFORMAT,
39
40
41
    print handler
42
43
    # setup the arguments for the handler method
44
    kwargs = \{\}
45
    kwargs['report_dir'] = tempfile.gettempdir()
46
    kwargs["run"] = True
47
    kwargs["package"] = u'Custom Tagging - Add Tags'
48
49
50
    # call the handler with the deploy_action method, passing in kwargs for arguments
51
    # this should throw an exception: pytan.exceptions.HandlerError
52
    import traceback
53
54
    try:
        handler.deploy_action(**kwargs)
55
    except Exception as e:
        traceback.print_exc(file=sys.stdout)
```

```
Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2
   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
4
        **kwargs
     File "/Users/jolsen/gh/pytan/lib/pytan/utils.py", line 2699, in wrap
6
        ret = f(*args, **kwargs)
     File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 1670, in _deploy_action
       empty_ok=False,
     File "/Users/jolsen/gh/pytan/lib/pytan/utils.py", line 2329, in build_param_objlist
10
        raise pytan.exceptions.HandlerError(err(obj_name, p_key, jsonify(obj_param)))
11
    HandlerError: PackageSpec, name: 'Custom Tagging - Add Tags', id: 27 parameter key '$1
12
                                                                                              requires a v
13
14
      "defaultValue": "",
      "helpString": "Enter tags space-delimited.",
15
      "key": "$1",
16
      "label": "Add tags (space-delimited)",
17
      "maxChars": 0,
18
      "model": "com.tanium.components.parameters::TextInputParameter",
19
      "parameterType": "com.tanium.components.parameters::TextInputParameter",
20
      "promptText": "e.g. PCI DMZ Decomm",
21
```

```
"restrict": null,
22
      "validationExpressions": [
23
24
          "expression": "\\S",
25
          "flags": "",
          "helpString": "You must enter a value",
27
          "model": "com.tanium.models::ValidationExpression",
28
          "parameterType": "com.tanium.models::ValidationExpression"
29
30
      ],
31
      "value": ""
32
```

PyTan API Valid Create Object Examples

Create user

Create a user called API Test User

Example Python Code

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

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

```
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
8
9
    print the object returned in JSON format:
10
11
      "_type": "user",
12
      "deleted_flag": 0,
13
      "group_id": 0,
14
      "id": 15,
15
      "last_login": "2001-01-01T00:00:00",
16
      "local_admin_flag": -1,
17
      "metadata": {
18
         "_type": "metadata",
19
         "item": [
20
21
             "_type": "item",
22
             "admin_flag": 0,
23
             "name": "TConsole.User.Property.property1",
24
             "value": "value1"
25
26
        ]
27
      },
28
      "name": "API Test User",
29
      "permissions": {
30
         "_type": "permissions",
31
         "permission": [
32
           "admin",
33
           "sensor_read",
34
           "sensor_write"
35
           "question_read",
36
           "question_write",
37
           "action_read",
38
           "action_write",
39
           "action_approval",
40
           "notification_write",
41
           "clients_read",
42
           "question_log_read",
43
           "content_admin"
44
        ]
45
46
      "roles": {
47
         "_type": "roles",
48
         "role": [
49
50
             "_type": "role",
51
             "description": "Administrators can perform all functions in the system, including creating of
52
             "id": 1,
53
             "name": "Administrator",
54
             "permissions": {
55
                "_type": "permissions",
56
                "permission": [
57
                  "admin",
58
                  "sensor_read",
59
                  "sensor_write"
60
                  "question_read",
61
                  "question_write",
62
                  "action_read",
63
                  "action_write",
64
                  "action_approval",
```

```
"notification_write",
66
                  "clients_read",
67
                 "question_log_read",
68
                  "content_admin"
69
70
71
             }
72
        ]
73
74
75
    2015-08-07 19:46:14,099 INFO
                                         pytan.handler: Deleted "User, name: 'API Test User', id: 15"
76
```

Create package

Create a package called package49

Example Python Code

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

```
loglevel=LOGLEVEL,
38
        debugformat=DEBUGFORMAT,
39
40
41
    print handler
42
43
    # setup the arguments for the delete method (to remove the package in case it exists)
44
    delete_kwargs = {}
45
    delete_kwarqs["objtype"] = 'package'
46
    delete_kwargs["name"] = 'package49'
47
48
    # setup the arguments for the handler method
50
51
    kwarqs = \{\}
    kwargs["expire_seconds"] = 1500
52
    kwargs["display_name"] = u'package49 API test'
53
    kwargs["name"] = u'package49'
54
    kwargs["parameters_json_file"] = u'../doc/example_of_all_package_parameters.json'
55
    kwargs["verify_expire_seconds"] = 3600
    kwargs["command"] = u'package49 $1 $2 $3 $4 $5 $6 $7 $8'
57
    kwargs["file_urls"] = [u'3600::testing.vbs||https://content.tanium.com/files/initialcontent/bundles/
58
    kwargs["verify_filter_options"] = [u'and']
59
    kwargs["verify_filters"] = [u'Custom Tags, that contains:tag']
60
    kwargs["command_timeout_seconds"] = 9999
61
62
    # delete the object in case it already exists
63
64
        handler.delete(**delete_kwargs)
65
    except Exception as e:
66
        print e
67
68
    # call the handler with the create_package method, passing in kwargs for arguments
69
    response = handler.create_package(**kwargs)
70
71
72
   print ""
73
   print "Type of response: ", type(response)
74
    print ""
76
    print "print of response:"
77
   print response
78
79
   print ""
80
   print "print the object returned in JSON format:"
81
   print response.to_json(response)
82
83
    # delete the object, we are done with it now
84
   try:
85
        handler.delete(**delete_kwargs)
86
    except Exception as e:
87
        print e
88
```

```
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:
```

```
Type of response: <class 'taniumpy.object_types.package_spec.PackageSpec'>
5
6
    print of response:
7
    PackageSpec, name: 'package49', id: 82
   print the object returned in JSON format:
10
11
      "_type": "package_spec",
12
      "available_time": "2001-01-01T00:00:00",
13
      "command": "package49 $1 $2 $3 $4 $5 $6 $7 $8",
14
      "command_timeout": 9999,
15
      "creation_time": "2001-01-01T00:00:00",
16
      "deleted_flag": 0,
17
      "display_name": "package49 API test",
18
      "expire_seconds": 1500,
19
      "files": {
20
        "_type": "package_files",
21
        "file": [
22
23
             "_type": "file",
24
             "bytes_downloaded": 0,
25
             "bytes_total": 0,
26
             "download_seconds": 3600,
27
             "file_status": {
28
               "_type": "file_status",
29
               "status": [
30
                 {
31
                   "_type": "status",
32
                   "bytes_downloaded": 0,
33
                   "bytes_total": 0,
34
                   "cache_status": "Processing",
35
                   "server_id": 1,
36
                   "server_name": "JTANIUM1.localdomain:17472",
37
                   "status": 0
38
                 }
39
               ]
40
41
             },
             "id": 184,
42
             "name": "testing.vbs",
43
             "size": 0,
44
             "source": "https://content.tanium.com/files/initialcontent/bundles/2014-10-01_1-32-15-7844/
45
             "status": 0
46
47
          }
        ]
48
      },
      "hidden_flag": 0,
50
      "id": 82,
51
      "last_update": "2001-01-01T00:00:00",
52
      "modification_time": "2001-01-01T00:00:00",
53
      "name": "package49",
54
      "parameter_definition": "{\"parameterType\": \"com.tanium.components.parameters::ParametersArray\"
55
      "skip_lock_flag": 0,
56
      "source_id": 0,
57
      "verify_expire_seconds": 3600,
58
      "verify_group": {
59
        "_type": "group",
60
        "id": 211
```

Create group

Create a group called All Windows Computers API Test

Example Python Code

```
import os
    import sys
2
    sys.dont_write_bytecode = True
3
    # Determine our script name, script dir
5
    my_file = os.path.abspath(sys.argv[0])
6
   my_dir = os.path.dirname(my_file)
7
    # determine the pytan lib dir and add it to the path
    parent_dir = os.path.dirname(my_dir)
10
   pytan_root_dir = os.path.dirname(parent_dir)
    lib_dir = os.path.join(pytan_root_dir, 'lib')
13
   path_adds = [lib_dir]
14
   for aa in path_adds:
15
        if aa not in sys.path:
16
            sys.path.append(aa)
17
18
19
    # connection info for Tanium Server
20
    USERNAME = "Tanium User"
21
    PASSWORD = "T@n!um"
22
    HOST = "172.16.31.128"
23
    PORT = "443"
24
25
    # Logging conrols
26
    LOGLEVEL = 2
27
   DEBUGFORMAT = False
28
29
    import tempfile
30
31
32
    import pytan
    handler = pytan.Handler(
33
        username=USERNAME,
34
        password=PASSWORD,
35
        host=HOST,
36
37
        port=PORT,
        loglevel=LOGLEVEL,
        debugformat=DEBUGFORMAT,
39
40
41
   print handler
42
43
    # setup the arguments for the delete method (to remove the package in case it exists)
```

```
delete_kwargs = {}
45
    delete_kwargs["objtype"] = 'group'
46
    delete_kwargs["name"] = 'All Windows Computers API Test'
47
48
49
    # setup the arguments for the handler method
50
    kwargs = \{\}
51
    kwargs["groupname"] = u'All Windows Computers API Test'
52
    kwarqs["filters"] = [u'Operating System, that contains:Windows']
53
    kwargs["filter_options"] = [u'and']
54
55
    # delete the object in case it already exists
56
57
        handler.delete(**delete_kwargs)
58
    except Exception as e:
59
        print e
60
61
    # call the handler with the create_group method, passing in kwargs for arguments
62
    response = handler.create_group(**kwargs)
63
64
65
    print ""
66
    print "Type of response: ", type(response)
67
68
    print ""
    print "print of response:"
70
71
    print response
72
    print ""
73
    print "print the object returned in JSON format:"
74
    print response.to_json(response)
75
76
    # delete the object, we are done with it now
77
78
    try:
        handler.delete(**delete_kwargs)
79
    except Exception as e:
80
        print e
81
```

```
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'!!
2
                                     pytan.handler: New group 'All Windows Computers API Test' created w
   2015-08-07 19:46:14,211 INFO
   Type of response: <class 'taniumpy.object_types.group.Group'>
5
6
   print of response:
7
8
   Group, name: 'All Windows Computers API Test', id: 212
9
10
   print the object returned in JSON format:
11
      "_type": "group",
12
      "and flag": 1,
13
      "deleted_flag": 0,
14
      "filters": {
15
        "_type": "filters",
16
        "filter": [
17
```

```
18
             "_type": "filter",
19
             "all_times_flag": 0,
20
             "all_values_flag": 0,
21
             "delimiter_index": 0,
22
             "ignore_case_flag": 1,
23
             "max_age_seconds": 0,
24
             "not_flag": 0,
25
             "operator": "RegexMatch",
26
             "sensor": {
27
               "_type": "sensor",
28
               "hash": 45421433
29
30
             "substring_flag": 0,
31
             "substring_length": 0,
32
             "substring_start": 0,
33
             "utf8_flag": 0,
34
             "value": ".*Windows.*",
35
             "value_type": "String"
           }
37
        ]
38
      },
39
      "id": 212,
40
      "name": "All Windows Computers API Test",
41
      "not_flag": 0,
42
43
      "sub_groups": {
        "_type": "groups",
44
        "group": []
45
46
      "text": " Operating System containing \"Windows\"",
47
      "type": 0
48
49
    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
3
   # Determine our script name, script dir
5
   my_file = os.path.abspath(sys.argv[0])
6
   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)
10
   pytan_root_dir = os.path.dirname(parent_dir)
11
   lib_dir = os.path.join(pytan_root_dir, 'lib')
12
   path_adds = [lib_dir]
13
14
   for aa in path_adds:
```

```
if aa not in sys.path:
16
             sys.path.append(aa)
17
18
19
    # connection info for Tanium Server
20
    USERNAME = "Tanium User"
21
    PASSWORD = "T@n!um"
22
    HOST = "172.16.31.128"
23
    PORT = "443"
24
25
    # Logging conrols
26
    LOGLEVEL = 2
27
    DEBUGFORMAT = False
28
29
    import tempfile
30
31
    import pytan
32
    handler = pytan.Handler(
33
34
        username=USERNAME,
        password=PASSWORD,
35
        host=HOST,
36
        port=PORT,
37
        loglevel=LOGLEVEL,
38
        debugformat=DEBUGFORMAT,
39
40
41
    print handler
42
43
    # setup the arguments for the delete method (to remove the package in case it exists)
44
    delete_kwargs = {}
45
    delete_kwargs["objtype"] = 'whitelisted_url'
46
    delete_kwargs["url_regex"] = 'regex:http://test.com/.*API_Test.*URL'
47
49
    # setup the arguments for the handler method
50
    kwargs = {}
51
    kwargs["url"] = u'http://test.com/.*API_Test.*URL'
52
    kwargs["regex"] = True
53
    kwargs["properties"] = [[u'property1', u'value1']]
54
    kwargs["download_seconds"] = 3600
55
56
    # delete the object in case it already exists
57
    try:
58
        handler.delete(**delete_kwargs)
59
    except Exception as e:
60
        print e
61
62
    # call the handler with the create_whitelisted_url method, passing in kwargs for arguments
63
    response = handler.create_whitelisted_url(**kwargs)
64
65
66
    print ""
67
    print "Type of response: ", type(response)
68
69
    print ""
70
    print "print of response:"
71
    print response
72
73
```

```
print ""
74
   print "print the object returned in JSON format:"
75
   print response.to_json(response)
76
77
    # delete the object, we are done with it now
78
79
        handler.delete(**delete_kwargs)
80
    except Exception as e:
81
        print e
82
```

```
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.
                                      pytan.handler: New Whitelisted URL 'regex:http://test.com/.*API_Tes
   2015-08-07 19:46:14,252 INFO
    Type of response: <class 'taniumpy.object_types.white_listed_url.WhiteListedUrl'>
5
6
    print of response:
7
   WhiteListedUrl, id: 52
8
Q
    print the object returned in JSON format:
10
11
      "_type": "white_listed_url",
12
      "download_seconds": 3600,
13
      "id": 52,
14
      "metadata": {
15
        "_type": "metadata",
16
        "item": [
17
18
            "_type": "item",
19
            "admin_flag": 0,
20
            "name": "TConsole.WhitelistedURL.property1",
21
            "value": "value1"
22
23
24
        ]
25
      },
      "url_regex": "regex:http://test.com/.*API_Test.*URL"
26
27
    2015-08-07 19:46:14,263 INFO
                                      pytan.handler: Deleted 'WhiteListedUrl, id: 52'
28
```

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

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

```
43
    # set the attribute name and value we want to add to the original object (if any)
44
    attr_name = "name"
45
    attr_add = " API TEST"
46
47
    # delete object before creating it?
48
    delete = True
49
50
    # setup the arguments for getting an object to export as json file
51
    get_kwargs = {}
52
    get_kwargs["objtype"] = u'package'
53
    get_kwargs["id"] = 31
54
55
56
    # get objects to use as an export to JSON file
57
    orig_objs = handler.get(**get_kwargs)
58
59
    # if attr_name and attr_add exists, modify the orig_objs to add attr_add to the attribute
60
    # attr_name
61
    if attr_name:
62
        for x in orig_objs:
63
            new_attr = getattr(x, attr_name)
64
            new_attr += attr_add
65
            setattr(x, attr_name, new_attr)
66
            if delete:
                 # delete the object in case it already exists
68
                 del_kwargs = {}
69
                 del_kwargs[attr_name] = new_attr
70
                 del_kwargs['objtype'] = u'package'
71
72
                 try:
                     handler.delete(**del_kwargs)
73
74
                 except Exception as e:
                     print e
75
76
    # export orig_objs to a json file
77
    json_file, results = handler.export_to_report_file(
78
        obj=orig_objs,
79
        export_format='json',
80
81
        report_dir=tempfile.gettempdir(),
82
83
    # create the object from the exported JSON file
84
    create_kwargs = {'objtype': u'package', 'json_file': json_file}
85
    response = handler.create_from_json(**create_kwargs)
86
87
88
89
    print "Type of response: ", type(response)
90
91
    print ""
92
    print "print of response:"
93
    print response
    print ""
    print "print the object returned in JSON format:"
   print response.to_json(response)
```

```
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'
2
    2015-08-07 19:46:14,305 INFO
                                       pytan.handler: Report file '/var/folders/dk/vjr1r_c53\folders2bbt_c
3
    2015-08-07 19:46:14,337 INFO
                                       pytan.handler: New PackageSpec, name: 'Disable Java Auto Update API
4
    Type of response: <class 'taniumpy.object_types.package_spec_list.PackageSpecList'>
6
    print of response:
8
   PackageSpecList, len: 1
10
    print the object returned in JSON format:
11
12
      "_type": "package_specs",
13
14
      "package_spec": [
15
          "_type": "package_spec",
16
          "available_time": "2015-08-07T13:22:40",
17
          "command": "cmd /c cscript //T:60 disable-java-auto-update.vbs",
18
          "command_timeout": 60,
19
          "creation_time": "2001-01-01T00:00:00",
20
          "deleted_flag": 0,
21
          "display_name": "Disable Java Auto Update",
22
          "expire_seconds": 660,
23
          "files": {
24
            "_type": "package_files",
25
            "file": [
26
27
               {
                 "_type": "file",
28
                 "bytes_downloaded": 0,
29
                 "bytes_total": 0,
30
                 "cache_status": "CACHED",
31
                 "download_seconds": 0,
32
                 "file_status": {
33
                   "_type": "file_status",
                   "status": [
35
36
                       "_type": "status",
37
                       "bytes_downloaded": 0,
38
                       "bytes_total": 0,
39
                       "cache_status": "Processing",
                       "server_id": 1,
41
                       "server_name": "JTANIUM1.localdomain:17472",
42
                       "status": 0
43
                     }
44
                   ]
45
46
                 "hash": "9e36208ce643c767ad76ef2ad6a69141fbb5a59a607b8eb8065db09e3a153c0d"
47
                 "id": 43,
48
                 "name": "disable-java-auto-update.vbs",
49
                 "size": 11377,
50
                 "source": "https://content.tanium.com/files/published/InitialContent/2015-06-04_18-59-45
51
                 "status": 0
52
53
54
            ]
55
          "hidden_flag": 0,
56
          "id": 83,
57
```

```
"last_update": "2001-01-01T00:00:00",
58
           "metadata": {
59
             "_type": "metadata",
60
             "item": [
61
                  "_type": "item",
                 "admin_flag": 0,
                 "name": "defined",
65
                  "value": "Tanium"
66
               },
67
68
                  "_type": "item",
                 "admin_flag": 0,
70
                  "name": "category",
71
                  "value": "Tanium"
72
73
             ]
74
75
           },
           "modification_time": "2001-01-01T00:00:00",
           "name": "Disable Java Auto Update API TEST",
77
           "skip_lock_flag": 0,
78
           "source_id": 0,
79
           "verify_expire_seconds": 600,
80
           "verify_group": {
81
             "_type": "group",
             "id": 0
83
84
           "verify_group_id": 0
85
86
      ]
87
```

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
2
   sys.dont_write_bytecode = True
3
   # 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)
10
   pytan_root_dir = os.path.dirname(parent_dir)
11
   lib_dir = os.path.join(pytan_root_dir, 'lib')
12
   path_adds = [lib_dir]
13
   for aa in path_adds:
15
        if aa not in sys.path:
```

```
sys.path.append(aa)
17
18
19
    # connection info for Tanium Server
20
    USERNAME = "Tanium User"
21
    PASSWORD = "T@n!um"
22
    HOST = "172.16.31.128"
23
    PORT = "443"
24
25
    # Logging conrols
26
    LOGLEVEL = 2
27
    DEBUGFORMAT = False
28
29
    import tempfile
30
31
    import pytan
32
    handler = pytan.Handler(
33
        username=USERNAME,
34
35
        password=PASSWORD,
        host=HOST,
36
        port=PORT,
37
        loglevel=LOGLEVEL,
38
        debugformat=DEBUGFORMAT,
39
40
41
42
    print handler
43
    # set the attribute name and value we want to add to the original object (if any)
44
    attr_name = "name"
45
    attr_add = " API TEST"
46
47
    # delete object before creating it?
48
    delete = True
50
    # setup the arguments for getting an object to export as json file
51
    qet_kwargs = {}
52
    get_kwargs["objtype"] = u'user'
53
    get_kwargs["id"] = 1
54
55
56
    # get objects to use as an export to JSON file
57
    orig_objs = handler.get(**get_kwargs)
58
59
    # if attr_name and attr_add exists, modify the orig_objs to add attr_add to the attribute
60
    # attr_name
61
    if attr_name:
62
        for x in orig_objs:
63
            new_attr = getattr(x, attr_name)
64
            new_attr += attr_add
65
             setattr(x, attr_name, new_attr)
66
             if delete:
67
                 # delete the object in case it already exists
68
                 del_kwargs = {}
69
                 del_kwargs[attr_name] = new_attr
70
                 del_kwargs['objtype'] = u'user'
71
                 try:
72.
                     handler.delete(**del_kwargs)
73
                 except Exception as e:
74
```

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

```
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 TE$T', id: 12"
2
    2015-08-07 19:46:14,359 INFO
                                       pytan.handler: Report file '/var/folders/dk/vjr1r_c53\px6k6gzp2bbt_c
3
    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'>
6
    print of response:
9
    UserList, len: 1
10
11
    print the object returned in JSON format:
12
      "_type": "users",
13
      "user": [
14
15
          "_type": "user",
          "deleted_flag": 0,
17
          "group_id": 0,
18
          "id": 16,
19
          "last_login": "2001-01-01T00:00:00",
20
21
          "local_admin_flag": -1,
          "name": "Jim Olsen API TEST",
22
23
          "permissions": {
            "_type": "permissions",
24
            "permission": [
25
              "admin",
26
              "sensor_read",
27
              "sensor_write",
28
               "question_read",
29
              "question_write",
```

```
"action_read",
31
                "action_write",
32
                "action_approval",
33
                "notification_write",
34
                "clients_read",
35
                "question_log_read",
                "content_admin"
37
             ]
38
           },
39
           "roles": {
40
              "_type": "roles",
41
             "role": [
42
43
                {
                  "_type": "role",
44
                  "description": "Administrators can perform all functions in the system, including creati
45
                  "id": 1,
46
                  "name": "Administrator",
47
                  "permissions": {
48
49
                    "_type": "permissions",
                    "permission": [
50
                       "admin",
51
                       "sensor_read",
52
                       "sensor_write",
53
                       "question_read",
54
                       "question_write",
55
                       "action_read",
56
                       "action_write",
57
                       "action_approval",
58
                       "notification_write",
59
                       "clients_read",
60
                       "question_log_read",
61
                       "content_admin"
62
                    ]
63
                  }
64
                }
65
             ]
66
67
68
         }
69
70
```

Create saved question from json

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

Example Python Code

```
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)
7
8
    # determine the pytan lib dir and add it to the path
    parent_dir = os.path.dirname(my_dir)
10
    pytan_root_dir = os.path.dirname(parent_dir)
11
    lib_dir = os.path.join(pytan_root_dir, 'lib')
12
    path_adds = [lib_dir]
13
14
    for aa in path_adds:
15
        if aa not in sys.path:
16
             sys.path.append(aa)
17
18
19
    # connection info for Tanium Server
20
    USERNAME = "Tanium User"
21
    PASSWORD = "T@n!um"
22
    HOST = "172.16.31.128"
23
    PORT = "443"
24
25
    # Logging conrols
26
    LOGLEVEL = 2
27
    DEBUGFORMAT = False
28
29
    import tempfile
30
31
32
    import pytan
    handler = pytan.Handler(
33
        username=USERNAME,
34
        password=PASSWORD,
35
        host=HOST,
36
        port=PORT,
37
        loglevel=LOGLEVEL,
38
        debugformat=DEBUGFORMAT,
39
40
41
    print handler
42
43
    # set the attribute name and value we want to add to the original object (if any)
44
    attr_name = "name"
45
    attr_add = " API TEST"
46
47
    # delete object before creating it?
48
    delete = True
49
50
    # setup the arguments for getting an object to export as json file
51
    get_kwargs = {}
52
    get_kwargs["objtype"] = u'saved_question'
53
    get_kwargs["id"] = 1
54
55
56
    # get objects to use as an export to JSON file
57
    orig_objs = handler.get(**get_kwargs)
58
59
    # if attr_name and attr_add exists, modify the oriq_objs to add attr_add to the attribute
60
    # attr_name
61
    if attr_name:
62
        for x in orig_objs:
63
            new_attr = getattr(x, attr_name)
```

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

```
Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
                                     pytan.handler: Deleted 'SavedQuestion, id: 109'
   2015-08-07 19:46:14,432 INFO
2
   2015-08-07 19:46:14,433 INFO
                                      pytan.handler: Report file '/var/folders/dk/vjr1r_c53\u03c7x6k6qzp2bbt_c
   2015-08-07 19:46:14,460 INFO
                                      pytan.handler: New SavedQuestion, name: 'Has Tanium Standard Utilit
4
   Type of response: <class 'taniumpy.object_types.saved_question_list.SavedQuestionList'>
   print of response:
8
   SavedQuestionList, len: 1
9
10
   print the object returned in JSON format:
11
12
13
      "_type": "saved_questions",
      "saved_question": [
14
15
          "_type": "saved_question",
16
          "action_tracking_flag": 0,
17
          "archive_enabled_flag": 0,
18
          "archive_owner": {
19
            "_type": "user"
```

```
21
           "expire_seconds": 600,
22
           "hidden_flag": 0,
23
           "id": 111,
24
           "issue_seconds": 120,
25
           "issue_seconds_never_flag": 0,
26
           "keep_seconds": 0,
27
           "mod_time": "2015-08-07T19:46:14",
28
           "mod_user": {
29
             "_type": "user",
30
             "name": "Tanium User"
31
32
33
           "most_recent_question_id": 1256,
           "name": "Has Tanium Standard Utilities API TEST",
34
           "packages": {
35
             "_type": "package_specs",
36
             "package_spec": [
37
38
                  "_type": "package_spec",
39
                 "id": 20,
40
                 "name": "Distribute Tanium Standard Utilities"
41
42
             ]
43
44
           "public_flag": 1,
45
           "query_text": "Get Has Tanium Standard Utilities from all machines",
46
           "question": {
47
             "_type": "question",
48
             "action_tracking_flag": 0,
49
             "expiration": "2015-08-07T19:32:37",
50
             "expire_seconds": 0,
51
52
             "force_computer_id_flag": 0,
             "hidden_flag": 0,
53
             "id": 1256,
54
             "management_rights_group": {
55
               "_type": "group",
56
               "id": 0
57
58
             },
             "query_text": "Get Has Tanium Standard Utilities from all machines",
59
             "saved_question": {
60
               "_type": "saved_question",
61
               "id": 110
62
63
             },
             "selects": {
64
               "_type": "selects",
65
               "select": [
66
67
                    "_type": "select",
68
                    "filter": {
69
                      "_type": "filter",
70
                      "all_times_flag": 0,
71
                      "all_values_flag": 0,
72
                      "delimiter_index": 0,
73
                      "end_time": "2001-01-01T00:00:00",
74
                      "ignore_case_flag": 1,
75
                      "max_age_seconds": 0,
76
                      "not_flag": 0,
77
                      "operator": "Less",
```

```
"start_time": "2001-01-01T00:00:00",
79
                     "substring_flag": 0,
80
                     "substring_length": 0,
81
                     "substring_start": 0,
82
                     "utf8_flag": 0,
83
                     "value_type": "String"
                   },
85
                   "sensor": {
86
                     "_type": "sensor",
87
                     "category": "Tanium",
88
                     "creation_time": "2015-08-07T13:22:09",
89
                     "delimiter": ",",
90
                     "description": "Returns whether a machine has the Tanium Standard Utilities\nExample
91
                     "exclude_from_parse_flag": 1,
92
                     "hash": 1782389954,
93
                     "hidden_flag": 0,
94
                     "id": 194,
95
                     "ignore_case_flag": 1,
                     "last_modified_by": "Jim Olsen",
97
                     "max_age_seconds": 900,
98
                     "modification_time": "2015-08-07T13:22:09",
99
                     "name": "Has Tanium Standard Utilities",
100
                     "queries": {
101
                       "_type": "queries",
102
                       "query": [
103
104
                         {
                            "_type": "query",
105
                            "platform": "Windows",
106
                            107
                            "script_type": "VBScript"
108
                         },
109
                          {
110
                           "_type": "query",
111
                           "platform": "Linux",
112
                            "script": "#!/bin/sh\n\n# THIS IS A STUB - NOT INTENDED AS FUNCTIONAL - NA\n#
113
                            "script_type": "UnixShell"
114
                         },
115
                          {
116
117
                            "_type": "query",
                            "platform": "Mac",
118
                            "script": "#!/bin/sh\n\n# THIS IS A STUB - NOT INTENDED AS FUNCTIONAL - NA\n#
119
                            "script_type": "UnixShell"
120
121
                         },
                          {
122
                            "_type": "query",
123
                            "platform": "Solaris",
124
                            "script": "#!/bin/sh\n\n# THIS IS A STUB - NOT INTENDED AS FUNCT#ONAL - NA\n#
125
                            "script_type": "UnixShell"
126
                         },
127
                          {
128
                            "_type": "query",
129
                            "platform": "AIX",
130
                            "script": "#!/bin/sh\n\n# THIS IS A STUB - NOT INTENDED AS FUNCT ONAL - NA\n#
131
                            "script_type": "UnixShell"
132
                          }
133
                       ]
134
135
                     },
                     "source_id": 0,
136
```

```
"string_count": 16,
137
                         "value_type": "String"
138
                       }
139
                    }
140
                  ]
141
               },
               "skip_lock_flag": 0,
143
               "user": {
144
                  "_type": "user",
145
                  "id": 1,
146
                  "name": "Jim Olsen"
147
148
149
             },
             "row_count_flag": 0,
150
             "sort_column": 0,
151
             "user": {
152
               "_type": "user",
153
               "id": 2,
154
               "name": "Tanium User"
155
156
          }
157
       ]
158
159
```

Create action from json

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

Example Python Code

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

```
25
    # Logging conrols
26
    LOGLEVEL = 2
27
    DEBUGFORMAT = False
28
29
    import tempfile
30
31
    import pytan
32
    handler = pytan.Handler(
33
        username=USERNAME,
34
        password=PASSWORD,
35
        host=HOST,
36
37
        port=PORT,
        loglevel=LOGLEVEL,
38
        debugformat=DEBUGFORMAT,
39
40
41
    print handler
42
43
    # set the attribute name and value we want to add to the original object (if any)
44
    attr_name = ""
45
    attr_add = ""
46
47
    # delete object before creating it?
48
    delete = False
49
50
    # setup the arguments for getting an object to export as json file
51
    get_kwargs = {}
52
    get_kwargs["objtype"] = u'action'
53
    get_kwargs["id"] = 1
54
55
56
    # get objects to use as an export to JSON file
57
    orig_objs = handler.get(**get_kwargs)
58
59
    # if attr_name and attr_add exists, modify the orig_objs to add attr_add to the attribute
60
    # attr_name
61
    if attr_name:
62
63
        for x in orig_objs:
             new_attr = getattr(x, attr_name)
64
             new_attr += attr_add
65
             setattr(x, attr_name, new_attr)
66
             if delete:
67
                 # delete the object in case it already exists
68
                 del_kwargs = {}
69
                 del_kwargs[attr_name] = new_attr
70
                 del_kwargs['objtype'] = u'action'
71
                 try:
72
                     handler.delete(**del_kwargs)
73
                 except Exception as e:
74
75
                     print e
76
    # export orig_objs to a json file
77
    json_file, results = handler.export_to_report_file(
78
        obj=orig_objs,
79
        export_format='json',
80
        report_dir=tempfile.gettempdir(),
81
82
```

```
83
    # create the object from the exported JSON file
84
    create_kwargs = {'objtype': u'action', 'json_file': json_file}
85
    response = handler.create_from_json(**create_kwargs)
86
87
   print ""
89
   print "Type of response: ", type(response)
90
91
   print ""
92
   print "print of response:"
93
   print response
   print ""
96
   print "print the object returned in JSON format:"
97
   print response.to_json(response)
```

```
Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
    2015-08-07 19:46:14,474 INFO
                                    pytan.handler: Report file '/var/folders/dk/vjr1r_c53yx6k6gzp2bbt_c
2
   2015-08-07 19:46:14,496 INFO
                                       pytan.handler: New Action, name: 'Distribute Tanium Standard Utilit
3
    Type of response: <class 'taniumpy.object_types.action_list.ActionList'>
6
   print of response:
7
   ActionList, len: 1
8
    print the object returned in JSON format:
10
11
      "_type": "actions",
12
      "action": [
13
14
          "_type": "action",
15
16
          "action_group": {
            "_type": "group",
17
            "id": 0,
18
            "name": "Default"
19
          },
20
          "approver": {
21
            "_type": "user",
22
            "id": 2,
23
            "name": "Tanium User"
24
25
          "comment": "Distribute Tanium Standard Utilities",
26
          "creation_time": "2015-08-07T19:46:14",
27
          "distribute_seconds": 3200,
28
          "expiration_time": "2015-08-07T20:41:16",
29
30
          "expire_seconds": 3300,
31
          "history_saved_question": {
            "_type": "saved_question",
32
            "id": 102
33
34
          }.
          "id": 60,
35
          "name": "Distribute Tanium Standard Utilities",
36
37
          "package_spec": {
            "_type": "package_spec",
```

```
"command": "cmd /c cscript install-standard-utils.vbs \"Tools\\StdUtils\"",
39
             "id": 20,
40
             "name": "Distribute Tanium Standard Utilities"
41
42
           },
           "saved_action": {
43
             "_type": "saved_action",
44
             "id": 46
45
           },
46
           "skip_lock_flag": 0,
47
           "start_time": "2015-08-07T19:46:16",
48
           "status": "Open",
49
           "stopped_flag": 0,
50
51
           "target_group": {
             "_type": "group",
52
             "id": 37,
53
             "name": "Default"
54
55
           },
           "user": {
56
             "_type": "user",
57
             "group_id": 0,
58
             "id": 2,
59
             "last_login": "2015-08-07T19:46:14",
60
             "name": "Tanium User"
61
62
         }
63
      ]
64
65
```

Create sensor from json

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

Example Python Code

```
import os
    import sys
2
    sys.dont_write_bytecode = True
3
    # Determine our script name, script dir
5
   my_file = os.path.abspath(sys.argv[0])
6
   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)
10
   pytan_root_dir = os.path.dirname(parent_dir)
11
    lib_dir = os.path.join(pytan_root_dir, 'lib')
12
   path_adds = [lib_dir]
13
14
    for aa in path_adds:
15
        if aa not in sys.path:
16
            sys.path.append(aa)
17
18
19
```

```
# connection info for Tanium Server
20
    USERNAME = "Tanium User"
21
    PASSWORD = "T@n!um"
22
    HOST = "172.16.31.128"
23
    PORT = "443"
24
25
    # Logging conrols
26
    LOGLEVEL = 2
27
    DEBUGFORMAT = False
28
29
    import tempfile
30
31
32
    import pytan
    handler = pytan.Handler(
33
        username=USERNAME,
34
        password=PASSWORD,
35
        host=HOST,
36
        port=PORT,
37
        loglevel=LOGLEVEL,
38
        debugformat=DEBUGFORMAT,
39
40
41
    print handler
42
43
    # set the attribute name and value we want to add to the original object (if any)
44
45
    attr_name = "name"
    attr_add = " API TEST"
46
47
    # delete object before creating it?
48
    delete = True
49
50
    # setup the arguments for getting an object to export as json file
51
    get_kwargs = {}
52
    get_kwargs["objtype"] = u'sensor'
53
    get_kwargs["id"] = 381
54
55
56
57
    # get objects to use as an export to JSON file
    orig_objs = handler.get(**get_kwargs)
58
59
    # if attr_name and attr_add exists, modify the orig_objs to add attr_add to the attribute
60
    # attr_name
61
    if attr_name:
62
        for x in orig_objs:
63
            new_attr = getattr(x, attr_name)
64
            new_attr += attr_add
65
            setattr(x, attr_name, new_attr)
66
             if delete:
67
                 # delete the object in case it already exists
68
                 del_kwargs = {}
69
70
                 del_kwargs[attr_name] = new_attr
71
                 del_kwargs['objtype'] = u'sensor'
72
                 try:
                     handler.delete(**del_kwargs)
73
                 except Exception as e:
74
                     print e
75
76
    # export orig_objs to a json file
```

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

```
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'
2
    2015-08-07 19:46:14,551 INFO
                                      pytan.handler: Report file '/var/folders/dk/vjr1r_c53\p/x6k6gzp2bbt_c
3
    2015-08-07 19:46:14,577 INFO
                                      pytan.handler: New Sensor, name: 'Is Mac API TEST', id: 642 (ID: 64
4
    Type of response: <class 'taniumpy.object_types.sensor_list.SensorList'>
6
    print of response:
8
    SensorList, len: 1
9
10
11
    print the object returned in JSON format:
12
      "_type": "sensors",
13
14
      "sensor": [
15
          "_type": "sensor",
16
          "category": "Operating System",
17
          "creation_time": "2015-08-07T19:46:14",
18
          "delimiter": ",",
19
          "description": "Returns whether the machine is a Mac. True if so, False if not.\nExample: Tru
20
21
          "exclude_from_parse_flag": 0,
          "hash": 2387245230,
22
          "hidden_flag": 0,
23
          "id": 642,
24
25
          "ignore_case_flag": 1,
          "last_modified_by": "Tanium User",
          "max_age_seconds": 86400,
27
          "modification_time": "2015-08-07T19:46:14",
28
          "name": "Is Mac API TEST",
29
          "queries": {
30
            "_type": "queries",
31
            "query": [
32
               {
```

```
"_type": "query",
34
                 "platform": "Windows",
35
                 "script": "'===================================\n' Is Mac\n'=======
36
                 "script_type": "VBScript"
37
               },
                 "_type": "query",
40
                 "platform": "Linux",
41
                 "script": "#!/bin/bash\necho False\n",
42
                 "script_type": "UnixShell"
43
               },
44
                 "_type": "query",
46
                 "platform": "Mac",
47
                 "script": "#!/bin/bash\necho True\n",
48
                 "script_type": "UnixShell"
49
50
               },
51
                 "_type": "query",
52
                 "platform": "Solaris",
53
                 "script": "#!/bin/sh\n\n# THIS IS A STUB - NOT INTENDED AS FUNCTIONAL - NA\n# \n# \n\nec
54
                 "script_type": "UnixShell"
55
              },
56
57
                 "_type": "query",
                 "platform": "AIX",
59
                 "script": "#!/bin/sh\n\n# THIS IS A STUB - NOT INTENDED AS FUNCTIONAL - NA\n# \n# \n\nec
60
                 "script_type": "UnixShell"
61
62
            ]
63
          },
64
          "source_id": 0,
          "string_count": 0,
66
          "value_type": "String"
67
68
      ]
69
```

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)
11
    lib_dir = os.path.join(pytan_root_dir, 'lib')
12
    path_adds = [lib_dir]
13
14
    for aa in path_adds:
15
        if aa not in sys.path:
             sys.path.append(aa)
17
18
19
    # connection info for Tanium Server
20
    USERNAME = "Tanium User"
21
    PASSWORD = "T@n!um"
22
    HOST = "172.16.31.128"
23
    PORT = "443"
24
25
    # Logging conrols
26
    LOGLEVEL = 2
27
    DEBUGFORMAT = False
28
29
    import tempfile
30
31
    import pytan
32
    handler = pytan.Handler(
33
        username=USERNAME,
34
35
        password=PASSWORD,
        host=HOST,
36
        port=PORT,
37
        loglevel=LOGLEVEL,
38
        debugformat=DEBUGFORMAT,
39
40
41
    print handler
42
43
    # set the attribute name and value we want to add to the original object (if any)
44
    attr_name = ""
45
    attr_add = ""
46
47
    # delete object before creating it?
48
    delete = False
49
50
    # setup the arguments for getting an object to export as json file
51
    get_kwargs = {}
52
    get_kwargs["objtype"] = u'question'
53
    get_kwargs["id"] = 1
54
55
56
    # get objects to use as an export to JSON file
57
    orig_objs = handler.get(**get_kwargs)
58
59
    # if attr_name and attr_add exists, modify the orig_objs to add attr_add to the attribute
60
    # attr_name
61
62
    if attr_name:
        for x in orig_objs:
63
            new_attr = getattr(x, attr_name)
64
            new_attr += attr_add
65
             setattr(x, attr_name, new_attr)
66
             if delete:
67
                 # delete the object in case it already exists
```

```
del_kwarqs = {}
69
                del_kwargs[attr_name] = new_attr
70
                del_kwargs['objtype'] = u'question'
71
72
                     handler.delete(**del_kwargs)
73
                except Exception as e:
74
                     print e
75
76
    # export orig_objs to a json file
77
    json_file, results = handler.export_to_report_file(
78
        obj=orig_objs,
79
        export_format='json',
80
        report_dir=tempfile.gettempdir(),
81
82
83
    # create the object from the exported JSON file
84
    create_kwargs = {'objtype': u'question', 'json_file': json_file}
85
    response = handler.create_from_json(**create_kwargs)
86
87
88
   print ""
89
   print "Type of response: ", type(response)
90
91
   print ""
92
   print "print of response:"
   print response
95
   print ""
   print "print the object returned in JSON format:"
97
   print response.to_json(response)
```

```
Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2
    2015-08-07 19:46:14,608 INFO
                                   pytan.handler: Report file '/var/folders/dk/vjr1r_c53yx6k6gzp2bbt_c
   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:
10
11
      "_type": "questions",
12
      "question": [
13
14
          "_type": "question",
15
          "action_tracking_flag": 0,
16
17
          "context_group": {
            "_type": "group",
18
            "id": 0
19
20
          "expiration": "2015-08-07T19:56:14",
21
          "expire_seconds": 0,
22
          "force_computer_id_flag": 1,
23
          "hidden_flag": 0,
```

```
"id": 1305,
25
           "management_rights_group": {
26
             "_type": "group",
27
             "id": 0
28
29
           "query_text": "Get Action Statuses matching \"Nil\" from all machines",
30
           "saved_question": {
31
             "_type": "saved_question",
32
             "id": 0
33
           },
34
           "selects": {
35
             "_type": "selects",
             "select": [
37
38
                 "_type": "select",
39
                 "filter": {
40
                    "_type": "filter",
41
                    "all_times_flag": 0,
42
                    "all_values_flag": 1,
43
                    "delimiter_index": 0,
44
                    "end_time": "2001-01-01T00:00:00",
45
                    "ignore_case_flag": 1,
46
                    "max_age_seconds": 0,
47
                    "not_flag": 0,
48
                    "operator": "RegexMatch",
49
                    "start_time": "2001-01-01T00:00:00",
50
                    "substring_flag": 0,
51
                    "substring_length": 0,
52
                    "substring_start": 0,
53
                    "utf8_flag": 0,
54
                    "value": "Nil",
55
                    "value_type": "String"
                 },
57
                 "sensor": {
58
                   "_type": "sensor",
59
                    "category": "Reserved",
60
                    "description": "The recorded state of each action a client has taken recently in the f
61
                    "exclude_from_parse_flag": 1,
62
                    "hash": 1792443391,
63
                    "hidden_flag": 0,
64
                    "id": 1,
65
                    "ignore_case_flag": 1,
66
                    "max_age_seconds": 3600,
67
                    "name": "Action Statuses",
68
69
                    "queries": {
                      "_type": "queries",
                      "query": [
71
72
                          "_type": "query",
73
                          "platform": "Windows",
74
                          "script": "Reserved",
75
                          "script_type": "WMIQuery"
76
77
                      ]
78
                    },
79
                    "source_id": 0,
80
                    "string_count": 238,
81
                    "value_type": "String"
82
```

```
83
84
              ]
85
86
            },
            "skip_lock_flag": 0,
87
            "user": {
              "_type": "user",
89
               "id": 2,
90
               "name": "Tanium User"
91
92
          }
93
94
95
```

Create whitelisted url from json

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

Example Python Code

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

```
username=USERNAME,
34
        password=PASSWORD,
35
        host=HOST,
36
        port=PORT,
37
        loglevel=LOGLEVEL,
38
        debugformat=DEBUGFORMAT,
39
40
41
   print handler
42
43
    # set the attribute name and value we want to add to the original object (if any)
44
    attr_name = "url_regex"
45
    attr_add = " API TEST"
46
47
    # delete object before creating it?
48
    delete = True
49
50
    # setup the arguments for getting an object to export as json file
51
    get_kwargs = {}
52
    get_kwargs["objtype"] = u'whitelisted_url'
53
    get_kwargs["url_regex"] = u'test1'
54
55
56
    # get objects to use as an export to JSON file
57
    orig_objs = handler.get(**get_kwargs)
58
59
    # if attr_name and attr_add exists, modify the orig_objs to add attr_add to the attribute
60
    # attr_name
61
    if attr_name:
62
        for x in orig_objs:
63
64
            new_attr = getattr(x, attr_name)
            new_attr += attr_add
65
            setattr(x, attr_name, new_attr)
66
            if delete:
67
                 # delete the object in case it already exists
68
                 del_kwargs = {}
69
                 del_kwargs[attr_name] = new_attr
70
                 del_kwargs['objtype'] = u'whitelisted_url'
71
72
                     handler.delete(**del_kwargs)
73
                 except Exception as e:
74
                     print e
75
76
    # export orig_objs to a json file
77
78
    json_file, results = handler.export_to_report_file(
        obj=orig_objs,
        export_format='json',
80
        report_dir=tempfile.gettempdir(),
81
82
83
84
    # create the object from the exported JSON file
    create_kwargs = {'objtype': u'whitelisted_url', 'json_file': json_file}
85
    response = handler.create_from_json(**create_kwargs)
86
87
88
   print ""
89
   print "Type of response: ", type(response)
91
```

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

print ""
print ""
print ""
print "print the object returned in JSON format:"
print response.to_json(response)
```

```
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'
2
   2015-08-07 19:46:14,686 INFO
                                      pytan.handler: Report file '/var/folders/dk/vjr1r_c53yx6k6gzp2bbt_c
3
    2015-08-07 19:46:14,695 INFO
                                      pytan.handler: New WhiteListedUrl, id: 53 (ID: 53) created successf
4
    Type of response: <class 'taniumpy.object_types.white_listed_url_list.WhiteListedUrlList'>
6
    print of response:
8
    WhiteListedUrlList, len: 1
9
10
    print the object returned in JSON format:
11
12
13
      "_type": "white_listed_urls",
14
      "white_listed_url": [
15
          "_type": "white_listed_url",
16
          "download_seconds": 86400,
17
          "id": 53,
18
          "url_regex": "test1 API TEST"
19
20
      1
21
22
```

Create group from json

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

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

```
14
    for aa in path_adds:
15
        if aa not in sys.path:
16
             sys.path.append(aa)
17
18
19
    # connection info for Tanium Server
20
    USERNAME = "Tanium User"
21
    PASSWORD = "T@n!um"
22
    HOST = "172.16.31.128"
23
    PORT = "443"
24
25
26
    # Logging conrols
    LOGLEVEL = 2
27
    DEBUGFORMAT = False
28
29
    import tempfile
30
31
32
    import pytan
    handler = pytan.Handler(
33
        username=USERNAME,
34
        password=PASSWORD,
35
        host=HOST,
36
        port=PORT,
37
        loglevel=LOGLEVEL,
38
39
        debugformat=DEBUGFORMAT,
40
41
    print handler
42
43
    # set the attribute name and value we want to add to the original object (if any)
44
45
    attr_name = "name"
    attr_add = " API TEST"
46
47
    # delete object before creating it?
48
    delete = True
49
50
    # setup the arguments for getting an object to export as json file
51
52
    get_kwargs = {}
    get_kwargs["objtype"] = u'group'
53
    get_kwargs["name"] = u'All Computers'
54
55
56
    # get objects to use as an export to JSON file
57
    orig_objs = handler.get(**get_kwargs)
58
    # if attr_name and attr_add exists, modify the orig_objs to add attr_add to the attribute
60
    # attr_name
61
    if attr_name:
62
        for x in orig_objs:
63
64
            new_attr = getattr(x, attr_name)
             new_attr += attr_add
65
             setattr(x, attr_name, new_attr)
66
             if delete:
67
                 # delete the object in case it already exists
68
                 del_kwargs = {}
69
                 del_kwargs[attr_name] = new_attr
70
71
                 del_kwargs['objtype'] = u'group'
```

```
72
                try:
                     handler.delete(**del_kwargs)
73
                except Exception as e:
74
                     print e
75
76
    # export orig_objs to a json file
77
    json_file, results = handler.export_to_report_file(
78
        obj=orig_objs,
79
        export_format='json',
80
        report_dir=tempfile.gettempdir(),
81
82
83
84
    # create the object from the exported JSON file
    create_kwargs = {'objtype': u'group', 'json_file': json_file}
85
    response = handler.create_from_json(**create_kwargs)
86
87
88
   print ""
89
   print "Type of response: ", type(response)
91
   print ""
92
   print "print of response:"
93
   print response
94
95
   print ""
   print "print the object returned in JSON format:"
   print response.to_json(response)
```

```
Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2
    2015-08-07 19:46:14,746 INFO
                                     pytan.handler: Deleted 'Group, id: 157'
    2015-08-07 19:46:14,747 INFO
                                      pytan.handler: Report file '/var/folders/dk/vjr1r_c53\px6k6gzp2bbt_c
3
                                     pytan.handler: New Group, name: 'All Computers API TEST', id: 213 (
   2015-08-07 19:46:14,769 INFO
   Type of response: <class 'taniumpy.object_types.group_list.GroupList'>
   print of response:
8
   GroupList, len: 1
10
    print the object returned in JSON format:
11
12
13
      "_type": "groups",
      "group": [
14
15
          "_type": "group",
16
          "and_flag": 0,
17
18
          "deleted_flag": 0,
          "filters": {
19
            "_type": "filters",
            "filter": []
21
22
          },
          "id": 213,
23
          "name": "All Computers API TEST",
24
          "not_flag": 0,
25
          "sub_groups": {
26
            "_type": "groups",
```

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

```
41
   print handler
42
43
    # setup the arguments for getting an object to export as json file
44
    get_kwargs = {}
45
    get_kwargs["objtype"] = u'saved_action'
    get_kwargs["name"] = u'Distribute Tanium Standard Utilities'
47
48
    # get objects to use as an export to JSON file
49
    orig_objs = handler.get(**get_kwargs)
50
51
    # export orig_objs to a json file
52
    json_file, results = handler.export_to_report_file(
53
        obj=oriq_objs,
54
        export_format='json',
55
        report_dir=tempfile.gettempdir(),
56
57
58
    # call the handler with the create_from_json method, passing in kwargs for arguments
    # this should throw an exception: pytan.exceptions.HandlerError
60
    import traceback
61
62
    # create the object from the exported JSON file
63
    create_kwargs = {'objtype': u'saved_action', 'json_file': json_file}
64
        response = handler.create_from_json(**create_kwargs)
66
67
    except Exception as e:
        traceback.print_exc(file=sys.stdout)
68
```

```
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!)

```
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)
11
    lib_dir = os.path.join(pytan_root_dir, 'lib')
12
    path_adds = [lib_dir]
13
14
    for aa in path_adds:
15
        if aa not in sys.path:
16
            sys.path.append(aa)
17
18
19
    # connection info for Tanium Server
20
    USERNAME = "Tanium User"
21
    PASSWORD = "T@n!um"
22
    HOST = "172.16.31.128"
23
    PORT = "443"
24
25
    # Logging conrols
26
    LOGLEVEL = 2
27
    DEBUGFORMAT = False
28
29
    import tempfile
30
31
    import pytan
32
    handler = pytan.Handler(
33
        username=USERNAME,
34
        password=PASSWORD,
35
        host=HOST,
36
        port=PORT,
37
        loglevel=LOGLEVEL,
38
        debugformat=DEBUGFORMAT,
39
40
41
    print handler
42
43
    # setup the arguments for getting an object to export as json file
44
    get_kwargs = {}
45
    get_kwargs["objtype"] = u'client'
46
    get_kwargs["status"] = u'Leader'
47
48
    # get objects to use as an export to JSON file
49
    orig_objs = handler.get(**get_kwargs)
50
51
    # export orig_objs to a json file
52
    json_file, results = handler.export_to_report_file(
53
        obj=orig_objs,
54
        export_format='json',
55
        report_dir=tempfile.gettempdir(),
56
57
58
    # call the handler with the create_from_json method, passing in kwargs for arguments
59
    # this should throw an exception: pytan.exceptions.HandlerError
60
61
    import traceback
62
    # create the object from the exported JSON file
63
    create_kwargs = {'objtype': u'client', 'json_file': json_file}
64
65
        response = handler.create_from_json(**create_kwargs)
66
   except Exception as e:
```

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

```
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!)

```
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
9
    parent_dir = os.path.dirname(my_dir)
10
    pytan_root_dir = os.path.dirname(parent_dir)
11
    lib_dir = os.path.join(pytan_root_dir, 'lib')
12
    path_adds = [lib_dir]
13
14
    for aa in path_adds:
15
        if aa not in sys.path:
16
            sys.path.append(aa)
17
18
19
    # connection info for Tanium Server
20
    USERNAME = "Tanium User"
21
   PASSWORD = "T@n!um"
22
   HOST = "172.16.31.128"
23
   PORT = "443"
24
25
    # Logging conrols
    LOGLEVEL = 2
27
    DEBUGFORMAT = False
28
29
    import tempfile
30
31
    import pytan
32
33
    handler = pytan.Handler(
        username=USERNAME,
34
        password=PASSWORD,
35
        host=HOST,
36
```

```
port=PORT,
37
        loglevel=LOGLEVEL,
38
        debugformat=DEBUGFORMAT,
39
40
41
    print handler
42
43
    # setup the arguments for getting an object to export as json file
44
    get_kwargs = {}
45
    get_kwargs["objtype"] = u'userrole'
46
    get_kwargs["name"] = u'Administrator'
47
49
    # get objects to use as an export to JSON file
    orig_objs = handler.get(**get_kwargs)
50
51
    # export orig_objs to a json file
52
    json_file, results = handler.export_to_report_file(
53
        obj=orig_objs,
54
        export_format='json',
55
        report_dir=tempfile.gettempdir(),
56
57
58
    # call the handler with the create_from_json method, passing in kwargs for arguments
59
    # this should throw an exception: pytan.exceptions.HandlerError
60
    import traceback
61
62
    # create the object from the exported JSON file
63
    create_kwargs = {'objtype': u'userrole', 'json_file': json_file}
64
65
        response = handler.create_from_json(**create_kwargs)
66
    except Exception as e:
67
        traceback.print_exc(file=sys.stdout)
```

```
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
Q
    parent_dir = os.path.dirname(my_dir)
10
    pytan_root_dir = os.path.dirname(parent_dir)
11
    lib_dir = os.path.join(pytan_root_dir, 'lib')
12
    path_adds = [lib_dir]
13
14
    for aa in path_adds:
15
        if aa not in sys.path:
16
17
             sys.path.append(aa)
18
19
    # connection info for Tanium Server
20
    USERNAME = "Tanium User"
21
    PASSWORD = "T@n!um"
22
    HOST = "172.16.31.128"
23
    PORT = "443"
24
25
    # Logging conrols
26
    LOGLEVEL = 2
27
    DEBUGFORMAT = False
28
29
30
    import tempfile
31
    import pytan
32
    handler = pytan.Handler(
33
        username=USERNAME,
34
        password=PASSWORD,
35
        host=HOST,
36
        port=PORT,
37
        loglevel=LOGLEVEL,
38
        debugformat=DEBUGFORMAT,
39
40
41
    print handler
42
43
44
    # setup the arguments for getting an object to export as json file
    get_kwargs = {}
45
    get_kwargs["objtype"] = u'setting'
46
    get_kwargs["id"] = 1
47
48
    # get objects to use as an export to JSON file
49
    orig_objs = handler.get(**get_kwargs)
50
51
    # export orig_objs to a json file
52
    json_file, results = handler.export_to_report_file(
53
        obj=orig_objs,
54
        export_format='json',
55
56
        report_dir=tempfile.gettempdir(),
57
58
    # call the handler with the create_from_json method, passing in kwargs for arguments
59
    # this should throw an exception: pytan.exceptions.HandlerError
60
    import traceback
61
62
    # 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)
```

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

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

```
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
2
   2015-08-07 19:46:14,930 DEBUG
                                     pytan.handler.QuestionPoller: ID 1306: expiration resolved to 2015-
3
   2015-08-07 19:46:14,930 DEBUG
4
                                    pytan.handler.QuestionPoller: ID 1306: query_text resolved to Get (
   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.OuestionPoller: 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:
10
                                     pytan.handler.QuestionPoller: ID 1306: Timing: Started: 2015-08-07
   2015-08-07 19:46:19,940 DEBUG
11
   2015-08-07 19:46:24,944 DEBUG
                                     pytan.handler.QuestionPoller: ID 1306: Progress: Tested: 0, Passed:
12
   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:
14
                                     pytan.handler.QuestionPoller: ID 1306: Timing: Started: 2015-08-07
    2015-08-07 19:46:29,951 DEBUG
15
    2015-08-07 19:46:34,955 DEBUG
                                     pytan.handler.QuestionPoller: ID 1306: Progress: Tested: 0, Passed:
16
                                     pytan.handler.QuestionPoller: ID 1306: Timing: Started: 2015-08-07
   2015-08-07 19:46:34,955 DEBUG
17
   2015-08-07 19:46:39,959 DEBUG
                                     pytan.handler.QuestionPoller: ID 1306: Progress: Tested: 1, Passed:
18
   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
20
                                     pytan.handler.QuestionPoller: ID 1306: Progress: Tested: 1, Passed:
    2015-08-07 19:46:44,964 DEBUG
21
   2015-08-07 19:46:44,965 DEBUG
                                     pytan.handler.QuestionPoller: ID 1306: Timing: Started: 2015-08-07
22
    2015-08-07 19:46:49,971 DEBUG
                                     pytan.handler.QuestionPoller: ID 1306: Progress: Tested: 2, Passed:
23
   2015-08-07 19:46:49,971 DEBUG
                                     pytan.handler.QuestionPoller: ID 1306: Timing: Started: 2015-08-07
24
    2015-08-07 19:46:49,971 INFO
                                     pytan.handler.QuestionPoller: ID 1306: Progress Changed 100% (2 of
25
    2015-08-07 19:46:49,971 INFO
                                     pytan.handler.QuestionPoller: ID 1306: Reached Threshold of 99% (2
26
27
    print the export_str returned from export_obj():
28
   Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
29
                                     pytan.handler: Report file '/var/folders/dk/vjr1r_c53vx6k6qzp2bbt_c
   2015-08-07 19:46:14,843 INFO
30
   Traceback (most recent call last):
31
     File "<string>", line 67, in <module>
32
      File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 546, in create_from_json
33
        raise pytan.exceptions.HandlerError(m(objtype, json_createable))
34
   HandlerError: setting is not a json createable object! Supported objects: user, whitelisted_url, sav
35
```

Export resultset csv expand false

Export a ResultSet from asking a question as CSV with false for expand_grouped_columns

Example Python Code

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

```
LOGLEVEL = 2
    DEBUGFORMAT = False
28
29
    import tempfile
30
31
    import pytan
32
    handler = pytan.Handler(
33
        username=USERNAME,
34
        password=PASSWORD,
35
        host=HOST,
36
        port=PORT,
37
        loglevel=LOGLEVEL,
38
        debugformat=DEBUGFORMAT,
39
40
41
    print handler
42
43
    # setup the export_obj kwargs for later
44
    export_kwargs = {}
45
    export_kwargs["export_format"] = u'csv'
46
    export_kwargs["expand_grouped_columns"] = False
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
```

```
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 2015-
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 Composition Poller: ID 1307: id resolved to 1307
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 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)
                                     pytan.handler.QuestionPoller: ID 1307: Progress: Tested: 1, Passed:
    2015-08-07 19:46:55,088 DEBUG
10
    2015-08-07 19:46:55,088 DEBUG
                                     pytan.handler.QuestionPoller: ID 1307: Timing: Started: 2015-08-07
11
                                     pytan.handler.QuestionPoller: ID 1307: Progress Changed 50% (1 of 2
   2015-08-07 19:46:55,088 INFO
12
   2015-08-07 19:47:00,092 DEBUG
                                     pytan.handler.QuestionPoller: ID 1307: Progress: Tested: 1, Passed:
13
   2015-08-07 19:47:00,092 DEBUG
                                     pytan.handler.QuestionPoller: ID 1307: Timing: Started: 2015-08-07
14
   2015-08-07 19:47:05,096 DEBUG
                                     pytan.handler.QuestionPoller: ID 1307: Progress: Tested: 1, Passed:
15
   2015-08-07 19:47:05,096 DEBUG
                                     pytan.handler.QuestionPoller: ID 1307: Timing: Started: 2015-08-07
16
   2015-08-07 19:47:10,102 DEBUG
                                     pytan.handler.QuestionPoller: ID 1307: Progress: Tested: 2, Passed:
17
                                     pytan.handler.QuestionPoller: ID 1307: Timing: Started: 2015-08-07
   2015-08-07 19:47:10,102 DEBUG
18
                                     pytan.handler.QuestionPoller: ID 1307: Progress Changed 100% (2 of
   2015-08-07 19:47:10,102 INFO
19
   2015-08-07 19:47:10,102 INFO
                                     pytan.handler.QuestionPoller: ID 1307: Reached Threshold of 99% (2
20
21
    print the export_str returned from export_obj():
22
    Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
23
    2015-08-07 19:46:14,930 DEBUG
                                     pytan.handler.QuestionPoller: ID 1306: id resolved to 1306
24
   2015-08-07 19:46:14,930 DEBUG
                                     pytan.handler.QuestionPoller: ID 1306: expiration resolved to 2015-
25
   2015-08-07 19:46:14,930 DEBUG
                                     pytan.handler.QuestionPoller: ID 1306: query_text resolved to Get C
26
   2015-08-07 19:46:14,930 DEBUG
                                     pytan.handler.QuestionPoller: ID 1306: id resolved to 1306
27
   2015-08-07 19:46:14,930 DEBUG
                                     pytan.handler.QuestionPoller: ID 1306: Object Info resolved to Ques
28
   2015-08-07 19:46:14,933 DEBUG
                                     pytan.handler.QuestionPoller: ID 1306: Progress: Tested: 0, Passed:
29
   2015-08-07 19:46:14,933 DEBUG
                                     pytan.handler.QuestionPoller: ID 1306: Timing: Started: 2015-08-07
30
   2015-08-07 19:46:14,933 INFO
                                     pytan.handler.QuestionPoller: ID 1306: Progress Changed 0% (0 of 2)
31
   2015-08-07 19:46:19,940 DEBUG
                                     pytan.handler.QuestionPoller: ID 1306: Progress: Tested: 0, Passed:
32
    2015-08-07 19:46:19,940 DEBUG
                                     pytan.handler.QuestionPoller: ID 1306: Timing: Started: 2015-08-07
33
    2015-08-07 19:46:24,944 DEBUG
                                     pytan.handler.QuestionPoller: ID 1306: Progress: Tested: 0, Passed:
34
    2015-08-07 19:46:24,944 DEBUG
                                     pytan.handler.QuestionPoller: ID 1306: Timing: Started: 2015-08-07
35
    2015-08-07 19:46:29,951 DEBUG
                                     pytan.handler.QuestionPoller: ID 1306: Progress: Tested: 0, Passed:
36
   2015-08-07 19:46:29,951 DEBUG
                                     pytan.handler.QuestionPoller: ID 1306: Timing: Started: 2015-08-07
37
    ..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

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

```
19
    # connection info for Tanium Server
20
    USERNAME = "Tanium User"
21
    PASSWORD = "T@n!um"
22
    HOST = "172.16.31.128"
23
   PORT = "443"
24
25
    # Logging conrols
26
    LOGLEVEL = 2
27
   DEBUGFORMAT = False
28
29
    import tempfile
30
31
    import pytan
32
    handler = pytan.Handler(
33
        username=USERNAME,
34
        password=PASSWORD,
35
        host=HOST,
36
37
        port=PORT,
        loglevel=LOGLEVEL,
38
        debugformat=DEBUGFORMAT,
39
40
41
   print handler
42
43
44
    # setup the export_obj kwargs for later
    export_kwargs = {}
45
    export_kwargs["export_format"] = u'csv'
46
    export_kwargs["expand_grouped_columns"] = True
47
48
    # ask the question that will provide the resultset that we want to use
49
50
    ask_kwargs = {
        '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
57
    response = handler.ask(**ask_kwargs)
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
   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
```

```
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
2
   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
6
    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:
10
11
   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:
12
   2015-08-07 19:47:20,239 DEBUG
                                     pytan.handler.QuestionPoller: ID 1309: Timing: Started: 2015-08-07
13
   2015-08-07 19:47:20,239 INFO
                                     pytan.handler.QuestionPoller: ID 1309: Progress Changed 50% (1 of 2
14
   2015-08-07 19:47:25,243 DEBUG
                                     pytan.handler.QuestionPoller: ID 1309: Progress: Tested: 1, Passed:
15
   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:
17
    2015-08-07 19:47:30,250 DEBUG
                                     pytan.handler.QuestionPoller: ID 1309: Timing: Started: 2015-08-07
18
    2015-08-07 19:47:35,255 DEBUG
                                     pytan.handler.QuestionPoller: ID 1309: Progress: Tested: 1, Passed:
19
    2015-08-07 19:47:35,255 DEBUG
                                     pytan.handler.QuestionPoller: ID 1309: Timing: Started: 2015-08-07
20
    2015-08-07 19:47:40,259 DEBUG
                                     pytan.handler.QuestionPoller: ID 1309: Progress: Tested: 2, Passed:
21
    2015-08-07 19:47:40,259 DEBUG
                                     pytan.handler.QuestionPoller: ID 1309: Timing: Started: 2015-08-07
22
23
   2015-08-07 19:47:40,259 INFO
                                     pytan.handler.QuestionPoller: ID 1309: Progress Changed 100% (2 of
                                     pytan.handler.QuestionPoller: ID 1309: Reached Threshold of 99% (2
24
    2015-08-07 19:47:40,259 INFO
25
   print the export_str returned from export_obj():
26
   Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
27
    2015-08-07 19:46:50,080 DEBUG
                                     pytan.handler.QuestionPoller: ID 1307: id resolved to 1307
28
    2015-08-07 19:46:50,080 DEBUG
                                     pytan.handler.QuestionPoller: ID 1307: expiration resolved to 2015-
29
    2015-08-07 19:46:50,080 DEBUG
                                     pytan.handler.QuestionPoller: ID 1307: query_text resolved to Get C
30
    2015-08-07 19:46:50,080 DEBUG
                                     pytan.handler.QuestionPoller: ID 1307: id resolved to 1307
31
    2015-08-07 19:46:50,080 DEBUG
                                     pytan.handler.QuestionPoller: ID 1307: Object Info resolved to Ques
32
    2015-08-07 19:46:50,083 DEBUG
                                     pytan.handler.QuestionPoller: ID 1307: Progress: Tested: 0, Passed:
33
    2015-08-07 19:46:50,083 DEBUG
                                     pytan.handler.QuestionPoller: ID 1307: Timing: Started: 2015-08-07
34
   2015-08-07 19:46:50,083 INFO
                                     pytan.handler.QuestionPoller: ID 1307: Progress Changed 0% (0 of 2)
35
   2015-08-07 19:46:55,088 DEBUG
                                     pytan.handler.QuestionPoller: ID 1307: Progress: Tested: 1, Passed:
36
37
   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:
39
   2015-08-07 19:47:00,092 DEBUG
                                     pytan.handler.QuestionPoller: ID 1307: Timing: Started: 2015-08-07
40
   2015-08-07 19:47:05,096 DEBUG
                                     pytan.handler.QuestionPoller: ID 1307: Progress: Tested: 1, Passed:
41
    ..trimmed for brevity..
42
```

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)
7
    # determine the pytan lib dir and add it to the path
Q
    parent_dir = os.path.dirname(my_dir)
10
    pytan_root_dir = os.path.dirname(parent_dir)
11
    lib_dir = os.path.join(pytan_root_dir, 'lib')
12
    path_adds = [lib_dir]
13
14
    for aa in path_adds:
15
        if aa not in sys.path:
16
17
             sys.path.append(aa)
18
19
    # connection info for Tanium Server
20
    USERNAME = "Tanium User"
21
    PASSWORD = "T@n!um"
22
    HOST = "172.16.31.128"
23
    PORT = "443"
24
25
    # Logging conrols
26
    LOGLEVEL = 2
27
    DEBUGFORMAT = False
28
29
    import tempfile
30
31
    import pytan
32
    handler = pytan.Handler(
33
        username=USERNAME,
34
        password=PASSWORD,
35
        host=HOST,
36
        port=PORT,
37
        loglevel=LOGLEVEL,
38
        debugformat=DEBUGFORMAT,
39
40
41
    print handler
42
43
    # setup the export_obj kwargs for later
44
    export_kwargs = {}
45
    export_kwargs["header_sort"] = True
46
    export_kwargs["export_format"] = u'csv'
47
    export_kwargs["header_add_type"] = True
48
    export_kwargs["expand_grouped_columns"] = True
49
    export_kwargs["header_add_sensor"] = True
50
51
    # ask the question that will provide the resultset that we want to use
52
    ask_kwargs = {
53
        'qtype': 'manual',
54
        'sensors': [
55
             "Computer Name", "IP Route Details", "IP Address",
56
             'Folder Name Search with RegEx Match{dirname=Program Files,regex=.*Shared.*}',
57
58
        ],
59
    response = handler.ask(**ask_kwargs)
60
61
    # export the object to a string
62
    # (we could just as easily export to a file using export_to_report_file)
```

```
export_kwarqs['obj'] = response['question_results']
    export_str = handler.export_obj(**export_kwargs)
65
66
67
    print ""
68
    print "print the export_str returned from export_obj():"
    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
```

```
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
2
    2015-08-07 19:47:40,405 DEBUG
                                     pytan.handler.QuestionPoller: ID 1310: expiration resolved to 2015-
3
    2015-08-07 19:47:40,405 DEBUG
                                     pytan.handler.QuestionPoller: ID 1310: query_text resolved to Get (
4
    2015-08-07 19:47:40,405 DEBUG
                                     pytan.handler.QuestionPoller: ID 1310: id resolved to 1310
5
    2015-08-07 19:47:40,405 DEBUG
                                     pytan.handler.QuestionPoller: ID 1310: Object Info resolved to Ques
6
    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:
10
    2015-08-07 19:47:45,417 DEBUG
                                     pytan.handler.QuestionPoller: ID 1310: Timing: Started: 2015-08-07
11
                                     pytan.handler.QuestionPoller: ID 1310: Progress: Tested: 0, Passed:
    2015-08-07 19:47:50,421 DEBUG
12
    2015-08-07 19:47:50,421 DEBUG
                                     pytan.handler.QuestionPoller: ID 1310: Timing: Started: 2015-08-07
13
    2015-08-07 19:47:55,425 DEBUG
                                     pytan.handler.QuestionPoller: ID 1310: Progress: Tested: 0, Passed:
14
    2015-08-07 19:47:55,425 DEBUG
                                     pytan.handler.QuestionPoller: ID 1310: Timing: Started: 2015-08-07
15
    2015-08-07 19:48:00,431 DEBUG
                                     pytan.handler.QuestionPoller: ID 1310: Progress: Tested: 0, Passed:
16
    2015-08-07 19:48:00,431 DEBUG
                                     pytan.handler.QuestionPoller: ID 1310: Timing: Started: 2015-08-07
17
    2015-08-07 19:48:05,435 DEBUG
                                     pytan.handler.QuestionPoller: ID 1310: Progress: Tested: 0, Passed:
18
    2015-08-07 19:48:05,435 DEBUG
                                     pytan.handler.QuestionPoller: ID 1310: Timing: Started: 2015-08-07
19
20
    2015-08-07 19:48:10,440 DEBUG
                                     pytan.handler.QuestionPoller: ID 1310: Progress: Tested: 0, Passed:
                                     pytan.handler.QuestionPoller: ID 1310: Timing: Started: 2015-08-07
21
   2015-08-07 19:48:10,440 DEBUG
   2015-08-07 19:48:15,444 DEBUG
                                     pytan.handler.QuestionPoller: ID 1310: Progress: Tested: 0, Passed:
22
   2015-08-07 19:48:15,444 DEBUG
                                     pytan.handler.QuestionPoller: ID 1310: Timing: Started: 2015-08-07
23
24
   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
25
   2015-08-07 19:48:25,453 DEBUG
                                     pytan.handler.QuestionPoller: ID 1310: Progress: Tested: 0, Passed:
26
   2015-08-07 19:48:25,453 DEBUG
                                     pytan.handler.QuestionPoller: ID 1310: Timing: Started: 2015-08-07
27
    2015-08-07 19:48:30,459 DEBUG
                                     pytan.handler.QuestionPoller: ID 1310: Progress: Tested: 0, Passed:
28
29
    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:
30
    2015-08-07 19:48:35,467 DEBUG
                                     pytan.handler.QuestionPoller: ID 1310: Timing: Started: 2015-08-07
31
    2015-08-07 19:48:40,473 DEBUG
                                     pytan.handler.QuestionPoller: ID 1310: Progress: Tested: 0, Passed:
32
33
    2015-08-07 19:48:40,473 DEBUG
                                     pytan.handler.QuestionPoller: ID 1310: Timing: Started: 2015-08-07
34
    2015-08-07 19:48:45,481 DEBUG
                                     pytan.handler.QuestionPoller: ID 1310: Progress: Tested: 0, Passed:
35
    2015-08-07 19:48:45,481 DEBUG
                                     pytan.handler.QuestionPoller: ID 1310: Timing: Started: 2015-08-07
    2015-08-07 19:48:50,489 DEBUG
                                     pytan.handler.QuestionPoller: ID 1310: Progress: Tested: 0, Passed:
36
   2015-08-07 19:48:50,489 DEBUG
                                     pytan.handler.QuestionPoller: ID 1310: Timing: Started: 2015-08-07
37
   2015-08-07 19:48:55,493 DEBUG
                                     pytan.handler.QuestionPoller: ID 1310: Progress: Tested: 1, Passed:
38
   2015-08-07 19:48:55,493 DEBUG
                                     pytan.handler.QuestionPoller: ID 1310: Timing: Started: 2015-08-07
39
   2015-08-07 19:48:55,493 INFO
                                     pytan.handler.QuestionPoller: ID 1310: Progress Changed 50% (1 of 2
40
    2015-08-07 19:49:00,497 DEBUG
                                     pytan.handler.QuestionPoller: ID 1310: Progress: Tested: 1, Passed:
41
                                     pytan.handler.QuestionPoller: ID 1310: Timing: Started: 2015-08-07
   2015-08-07 19:49:00,497 DEBUG
```

```
2015-08-07 19:49:05,502 DEBUG
                                     pytan.handler.QuestionPoller: ID 1310: Progress: Tested: 1, Passed:
43
                                     pytan.handler.QuestionPoller: ID 1310: Timing: Started: 2015-08-07
    2015-08-07 19:49:05,502 DEBUG
44
    2015-08-07 19:49:10,508 DEBUG
                                     pytan.handler.QuestionPoller: ID 1310: Progress: Tested: 2, Passed:
45
   2015-08-07 19:49:10,508 DEBUG
                                     pytan.handler.QuestionPoller: ID 1310: Timing: Started: 2015-08-07
46
   2015-08-07 19:49:10,508 INFO
                                     pytan.handler.QuestionPoller: ID 1310: Progress Changed 100% (2 of
47
   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():
50
   Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
51
   2015-08-07 19:47:10,227 DEBUG
                                     pytan.handler.QuestionPoller: ID 1309: id resolved to 1309
52
   2015-08-07 19:47:10,227 DEBUG
                                     pytan.handler.QuestionPoller: ID 1309: expiration resolved to 2015-
53
                                     pytan.handler.QuestionPoller: ID 1309: query_text resolved to Get C
    2015-08-07 19:47:10,227 DEBUG
54
    2015-08-07 19:47:10,227 DEBUG
                                     pytan.handler.QuestionPoller: ID 1309: id resolved to 1309
55
    2015-08-07 19:47:10,227 DEBUG
                                     pytan.handler.QuestionPoller: ID 1309: Object Info resolved to Ques
56
                                     pytan.handler.QuestionPoller: ID 1309: Progress: Tested: 0, Passed:
    2015-08-07 19:47:10,231 DEBUG
57
    2015-08-07 19:47:10,231 DEBUG
                                     pytan.handler.QuestionPoller: ID 1309: Timing: Started: 2015-08-07
58
   2015-08-07 19:47:10,231 INFO
                                     pytan.handler.QuestionPoller: ID 1309: Progress Changed 0% (0 of 2)
59
   2015-08-07 19:47:15,235 DEBUG
                                     pytan.handler.QuestionPoller: ID 1309: Progress: Tested: 0, Passed:
60
                                     pytan.handler.QuestionPoller: ID 1309: Timing: Started: 2015-08-07
   2015-08-07 19:47:15,235 DEBUG
61
   2015-08-07 19:47:20,239 DEBUG
                                     pytan.handler.QuestionPoller: ID 1309: Progress: Tested: 1, Passed:
62
                                     pytan.handler.QuestionPoller: ID 1309: Timing: Started: 2015-08-07
   2015-08-07 19:47:20,239 DEBUG
63
   2015-08-07 19:47:20,239 INFO
                                     pytan.handler.QuestionPoller: ID 1309: Progress Changed 50% (1 of 2
64
   2015-08-07 19:47:25,243 DEBUG
                                     pytan.handler.QuestionPoller: ID 1309: Progress: Tested: 1, Passed:
65
    ..trimmed for brevity..
```

Export resultset json

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

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

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

```
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 Company of the 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 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
                                     pytan.handler.QuestionPoller: ID 1311: Progress Changed 0% (0 of 2)
    2015-08-07 19:49:10,713 INFO
    2015-08-07 19:49:15,721 DEBUG
                                     pytan.handler.QuestionPoller: ID 1311: Progress: Tested: 0, Passed:
10
                                     pytan.handler.QuestionPoller: ID 1311: Timing: Started: 2015-08-07
   2015-08-07 19:49:15,721 DEBUG
11
                                     pytan.handler.QuestionPoller: ID 1311: Progress: Tested: 0, Passed:
   2015-08-07 19:49:20,725 DEBUG
12
                                     pytan.handler.QuestionPoller: ID 1311: Timing: Started: 2015-08-07
   2015-08-07 19:49:20,726 DEBUG
13
   2015-08-07 19:49:25,730 DEBUG
                                     pytan.handler.QuestionPoller: ID 1311: Progress: Tested: 0, Passed:
14
   2015-08-07 19:49:25,730 DEBUG
                                     pytan.handler.QuestionPoller: ID 1311: Timing: Started: 2015-08-07
15
   2015-08-07 19:49:30,739 DEBUG
                                     pytan.handler.QuestionPoller: ID 1311: Progress: Tested: 1, Passed:
16
   2015-08-07 19:49:30,739 DEBUG
                                     pytan.handler.QuestionPoller: ID 1311: Timing: Started: 2015-08-07
17
   2015-08-07 19:49:30,739 INFO
                                     pytan.handler.QuestionPoller: ID 1311: Progress Changed 50% (1 of 2
18
    2015-08-07 19:49:35,743 DEBUG
                                     pytan.handler.QuestionPoller: ID 1311: Progress: Tested: 1, Passed:
19
    2015-08-07 19:49:35,744 DEBUG
                                     pytan.handler.QuestionPoller: ID 1311: Timing: Started: 2015-08-07
20
    2015-08-07 19:49:40,751 DEBUG
                                     pytan.handler.QuestionPoller: ID 1311: Progress: Tested: 2, Passed:
21
    2015-08-07 19:49:40,751 DEBUG
                                     pytan.handler.QuestionPoller: ID 1311: Timing: Started: 2015-08-07
22
    2015-08-07 19:49:40,751 INFO
                                     pytan.handler.QuestionPoller: ID 1311: Progress Changed 100% (2 of
23
   2015-08-07 19:49:40,751 INFO
                                     pytan.handler.QuestionPoller: ID 1311: Reached Threshold of 99% (2
24
25
   print the export_str returned from export_obj():
26
   Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
27
   2015-08-07 19:47:40,405 DEBUG
                                     pytan.handler.QuestionPoller: ID 1310: id resolved to 1310
28
   2015-08-07 19:47:40,405 DEBUG
                                     pytan.handler.QuestionPoller: ID 1310: expiration resolved to 2015-
29
   2015-08-07 19:47:40,405 DEBUG
                                     pytan.handler.QuestionPoller: ID 1310: query_text resolved to Get C
30
   2015-08-07 19:47:40,405 DEBUG
                                     pytan.handler.QuestionPoller: ID 1310: id resolved to 1310
31
    2015-08-07 19:47:40,405 DEBUG
                                     pytan.handler.QuestionPoller: ID 1310: Object Info resolved to Ques
32
    2015-08-07 19:47:40,408 DEBUG
                                     pytan.handler.QuestionPoller: ID 1310: Progress: Tested: 0, Passed:
33
    2015-08-07 19:47:40,408 DEBUG
                                     pytan.handler.QuestionPoller: ID 1310: Timing: Started: 2015-08-07
34
    2015-08-07 19:47:40,408 INFO
                                     pytan.handler.QuestionPoller: ID 1310: Progress Changed 0% (0 of 2)
35
    2015-08-07 19:47:45,417 DEBUG
                                     pytan.handler.QuestionPoller: ID 1310: Progress: Tested: 0, Passed:
36
    2015-08-07 19:47:45,417 DEBUG
                                     pytan.handler.QuestionPoller: ID 1310: Timing: Started: 2015-08-07
37
   2015-08-07 19:47:50,421 DEBUG
                                     pytan.handler.QuestionPoller: ID 1310: Progress: Tested: 0, Passed:
38
   2015-08-07 19:47:50,421 DEBUG
                                     pytan.handler.QuestionPoller: ID 1310: Timing: Started: 2015-08-07
39
   2015-08-07 19:47:55,425 DEBUG
                                     pytan.handler.QuestionPoller: ID 1310: Progress: Tested: 0, Passed:
40
   2015-08-07 19:47:55,425 DEBUG
                                     pytan.handler.QuestionPoller: ID 1310: Timing: Started: 2015-08-07
41
    ..trimmed for brevity..
42
```

Export resultset csv sort empty

Export a ResultSet from asking a question as CSV with an empty list for header_sort

```
import os
1
2
   import sys
   sys.dont_write_bytecode = True
3
   # 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)
10
   pytan_root_dir = os.path.dirname(parent_dir)
11
   lib_dir = os.path.join(pytan_root_dir, 'lib')
12
   path_adds = [lib_dir]
```

```
14
    for aa in path_adds:
15
        if aa not in sys.path:
16
             sys.path.append(aa)
17
18
    # connection info for Tanium Server
20
    USERNAME = "Tanium User"
21
    PASSWORD = "T@n!um"
22
    HOST = "172.16.31.128"
23
    PORT = "443"
24
25
26
    # Logging conrols
    LOGLEVEL = 2
27
    DEBUGFORMAT = False
28
29
    import tempfile
30
31
32
    import pytan
    handler = pytan.Handler(
33
        username=USERNAME,
34
        password=PASSWORD,
35
        host=HOST,
36
        port=PORT,
37
        loglevel=LOGLEVEL,
38
        debugformat=DEBUGFORMAT,
39
40
41
    print handler
42
43
    # setup the export_obj kwargs for later
44
45
    export_kwargs = {}
    export_kwargs["export_format"] = u'csv'
46
    export_kwargs["header_sort"] = []
47
48
    # ask the question that will provide the resultset that we want to use
49
    ask_kwargs = {
50
        'qtype': 'manual',
51
52
        'sensors': [
             "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
    # (we could just as easily export to a file using export_to_report_file)
60
    export_kwarqs['obj'] = response['question_results']
61
    export_str = handler.export_obj(**export_kwargs)
62
63
64
    print ""
    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
```

```
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
2
    2015-08-07 19:49:40,850 DEBUG
                                     pytan.handler.QuestionPoller: ID 1312: expiration resolved to 2015-
3
    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
                                     pytan.handler.QuestionPoller: ID 1312: Object Info resolved to Ques
    2015-08-07 19:49:40,850 DEBUG
   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
                                     pytan.handler.QuestionPoller: ID 1312: Progress Changed 0% (0 of 2)
   2015-08-07 19:49:40,854 INFO
   2015-08-07 19:49:45,859 DEBUG
                                     pytan.handler.QuestionPoller: ID 1312: Progress: Tested: 0, Passed:
10
   2015-08-07 19:49:45,859 DEBUG
                                     pytan.handler.QuestionPoller: ID 1312: Timing: Started: 2015-08-07
11
   2015-08-07 19:49:50,863 DEBUG
                                     pytan.handler.QuestionPoller: ID 1312: Progress: Tested: 0, Passed:
12
    2015-08-07 19:49:50,863 DEBUG
                                     pytan.handler.QuestionPoller: ID 1312: Timing: Started: 2015-08-07
13
    2015-08-07 19:49:55,870 DEBUG
                                     pytan.handler.QuestionPoller: ID 1312: Progress: Tested: 0, Passed:
14
    2015-08-07 19:49:55,870 DEBUG
                                     pytan.handler.QuestionPoller: ID 1312: Timing: Started: 2015-08-07
15
    2015-08-07 19:50:00,877 DEBUG
                                     pytan.handler.QuestionPoller: ID 1312: Progress: Tested: 0, Passed:
16
    2015-08-07 19:50:00,877 DEBUG
                                     pytan.handler.QuestionPoller: ID 1312: Timing: Started: 2015-08-07
17
    2015-08-07 19:50:05,881 DEBUG
                                     pytan.handler.QuestionPoller: ID 1312: Progress: Tested: 0, Passed:
18
    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:
20
    2015-08-07 19:50:10,886 DEBUG
                                     pytan.handler.QuestionPoller: ID 1312: Timing: Started: 2015-08-07
21
   2015-08-07 19:50:15,891 DEBUG
                                     pytan.handler.QuestionPoller: ID 1312: Progress: Tested: 0, Passed:
22
    2015-08-07 19:50:15,891 DEBUG
                                     pytan.handler.QuestionPoller: ID 1312: Timing: Started: 2015-08-07
23
    2015-08-07 19:50:20,896 DEBUG
                                     pytan.handler.QuestionPoller: ID 1312: Progress: Tested: 0, Passed:
24
    2015-08-07 19:50:20,896 DEBUG
                                     pytan.handler.QuestionPoller: ID 1312: Timing: Started: 2015-08-07
25
                                     pytan.handler.QuestionPoller: ID 1312: Progress: Tested: 0, Passed:
    2015-08-07 19:50:25,901 DEBUG
26
    2015-08-07 19:50:25,901 DEBUG
                                     pytan.handler.QuestionPoller: ID 1312: Timing: Started: 2015-08-07
27
    2015-08-07 19:50:30,906 DEBUG
                                     pytan.handler.QuestionPoller: ID 1312: Progress: Tested: 0, Passed:
28
    2015-08-07 19:50:30,906 DEBUG
                                     pytan.handler.QuestionPoller: ID 1312: Timing: Started: 2015-08-07
29
    2015-08-07 19:50:35,910 DEBUG
                                     pytan.handler.QuestionPoller: ID 1312: Progress: Tested: 0, Passed:
30
                                     pytan.handler.QuestionPoller: ID 1312: Timing: Started: 2015-08-07
31
    2015-08-07 19:50:35,910 DEBUG
                                     pytan.handler.QuestionPoller: ID 1312: Progress: Tested: 0, Passed:
32
   2015-08-07 19:50:40,915 DEBUG
   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:
34
   2015-08-07 19:50:45,919 DEBUG
                                     pytan.handler.QuestionPoller: ID 1312: Timing: Started: 2015-08-07
35
   2015-08-07 19:50:50,923 DEBUG
                                     pytan.handler.QuestionPoller: ID 1312: Progress: Tested: 0, Passed:
36
   2015-08-07 19:50:50,923 DEBUG
                                     pytan.handler.QuestionPoller: ID 1312: Timing: Started: 2015-08-07
37
   2015-08-07 19:50:55,928 DEBUG
                                     pytan.handler.QuestionPoller: ID 1312: Progress: Tested: 0, Passed:
38
    2015-08-07 19:50:55,928 DEBUG
                                     pytan.handler.QuestionPoller: ID 1312: Timing: Started: 2015-08-07
40
    2015-08-07 19:51:00,934 DEBUG
                                     pytan.handler.QuestionPoller: ID 1312: Progress: Tested: 0, Passed:
    2015-08-07 19:51:00,935 DEBUG
                                     pytan.handler.QuestionPoller: ID 1312: Timing: Started: 2015-08-07
41
    2015-08-07 19:51:05,939 DEBUG
                                     pytan.handler.QuestionPoller: ID 1312: Progress: Tested: 0, Passed:
42.
    2015-08-07 19:51:05,939 DEBUG
                                     pytan.handler.QuestionPoller: ID 1312: Timing: Started: 2015-08-07
43
                                     pytan.handler.QuestionPoller: ID 1312: Progress: Tested: 2, Passed:
44
    2015-08-07 19:51:10,947 DEBUG
   2015-08-07 19:51:10,947 DEBUG
                                     pytan.handler.QuestionPoller: ID 1312: Timing: Started: 2015-08-07
45
                                     pytan.handler.QuestionPoller: ID 1312: Progress Changed 100% (2 of
    2015-08-07 19:51:10,947 INFO
   2015-08-07 19:51:10,947 INFO
                                     pytan.handler.QuestionPoller: ID 1312: Reached Threshold of 99% (2
47
48
   print the export_str returned from export_obj():
49
   Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
50
                                     pytan.handler.QuestionPoller: ID 1311: id resolved to 1311
   2015-08-07 19:49:10,709 DEBUG
51
    2015-08-07 19:49:10,709 DEBUG
                                     pytan.handler.QuestionPoller: ID 1311: expiration resolved to 2015-
52
```

pytan.handler.QuestionPoller: ID 1311: query_text resolved to Get (

2015-08-07 19:49:10,709 DEBUG

```
2015-08-07 19:49:10,709 DEBUG
                                     pytan.handler.QuestionPoller: ID 1311: id resolved to 1311
54
   2015-08-07 19:49:10,709 DEBUG
                                     pytan.handler.QuestionPoller: ID 1311: Object Info resolved to Ques
55
   2015-08-07 19:49:10,713 DEBUG
                                     pytan.handler.QuestionPoller: ID 1311: Progress: Tested: 0, Passed:
56
                                     pytan.handler.QuestionPoller: ID 1311: Timing: Started: 2015-08-07
   2015-08-07 19:49:10,713 DEBUG
57
                                     pytan.handler.QuestionPoller: ID 1311: Progress Changed 0% (0 of 2)
   2015-08-07 19:49:10,713 INFO
   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
60
   2015-08-07 19:49:20,725 DEBUG
                                     pytan.handler.QuestionPoller: ID 1311: Progress: Tested: 0, Passed:
61
   2015-08-07 19:49:20,726 DEBUG
                                     pytan.handler.QuestionPoller: ID 1311: Timing: Started: 2015-08-07
62
   2015-08-07 19:49:25,730 DEBUG
                                     pytan.handler.QuestionPoller: ID 1311: Progress: Tested: 0, Passed:
63
                                     pytan.handler.QuestionPoller: ID 1311: Timing: Started: 2015-08-07
   2015-08-07 19:49:25,730 DEBUG
64
   ..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

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

```
port=PORT,
37
        loglevel=LOGLEVEL,
38
        debugformat=DEBUGFORMAT,
39
40
41
    print handler
42
43
    # setup the export_obj kwargs for later
44
    export_kwarqs = {}
45
    export_kwargs["export_format"] = u'csv'
46
    export_kwarqs["header_sort"] = True
47
49
    # ask the question that will provide the resultset that we want to use
    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
72.
```

```
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
2
   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
6
   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
8
                                     pytan.handler.QuestionPoller: ID 1313: Progress Changed 0% (0 of 2)
   2015-08-07 19:51:11,066 INFO
   2015-08-07 19:51:16,074 DEBUG
                                     pytan.handler.QuestionPoller: ID 1313: Progress: Tested: 0, Passed:
10
   2015-08-07 19:51:16,074 DEBUG
                                     pytan.handler.QuestionPoller: ID 1313: Timing: Started: 2015-08-07
11
   2015-08-07 19:51:21,079 DEBUG
                                     pytan.handler.QuestionPoller: ID 1313: Progress: Tested: 0, Passed:
12
   2015-08-07 19:51:21,079 DEBUG
                                     pytan.handler.QuestionPoller: ID 1313: Timing: Started: 2015-08-07
13
   2015-08-07 19:51:26,083 DEBUG
                                     pytan.handler.QuestionPoller: ID 1313: Progress: Tested: 0, Passed:
14
   2015-08-07 19:51:26,083 DEBUG
                                     pytan.handler.QuestionPoller: ID 1313: Timing: Started: 2015-08-07
15
   2015-08-07 19:51:31,089 DEBUG
                                     pytan.handler.QuestionPoller: ID 1313: Progress: Tested: 0, Passed:
16
   2015-08-07 19:51:31,089 DEBUG
                                     pytan.handler.QuestionPoller: ID 1313: Timing: Started: 2015-08-07
17
   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:
20
    2015-08-07 19:51:41,099 DEBUG
                                     pytan.handler.QuestionPoller: ID 1313: Timing: Started: 2015-08-07
21
                                     pytan.handler.QuestionPoller: ID 1313: Progress: Tested: 0, Passed:
   2015-08-07 19:51:46,107 DEBUG
22
    2015-08-07 19:51:46,107 DEBUG
                                     pytan.handler.QuestionPoller: ID 1313: Timing: Started: 2015-08-07
23
   2015-08-07 19:51:51,112 DEBUG
                                     pytan.handler.QuestionPoller: ID 1313: Progress: Tested: 0, Passed:
24
                                     pytan.handler.QuestionPoller: ID 1313: Timing: Started: 2015-08-07
   2015-08-07 19:51:51,112 DEBUG
25
    2015-08-07 19:51:56,117 DEBUG
                                     pytan.handler.QuestionPoller: ID 1313: Progress: Tested: 0, Passed:
26
   2015-08-07 19:51:56,117 DEBUG
                                     pytan.handler.QuestionPoller: ID 1313: Timing: Started: 2015-08-07
27
   2015-08-07 19:52:01,121 DEBUG
                                     pytan.handler.QuestionPoller: ID 1313: Progress: Tested: 0, Passed:
28
   2015-08-07 19:52:01,122 DEBUG
                                     pytan.handler.QuestionPoller: ID 1313: Timing: Started: 2015-08-07
29
    2015-08-07 19:52:06,125 DEBUG
                                     pytan.handler.QuestionPoller: ID 1313: Progress: Tested: 0, Passed:
                                     pytan.handler.QuestionPoller: ID 1313: Timing: Started: 2015-08-07
    2015-08-07 19:52:06,126 DEBUG
31
    2015-08-07 19:52:11,132 DEBUG
                                      pytan.handler.QuestionPoller: ID 1313: Progress: Tested: 0, Passed:
32
    2015-08-07 19:52:11,133 DEBUG
                                     pytan.handler.QuestionPoller: ID 1313: Timing: Started: 2015-08-07
33
    2015-08-07 19:52:16,136 DEBUG
                                     pytan.handler.QuestionPoller: ID 1313: Progress: Tested: 0, Passed:
34
                                     pytan.handler.QuestionPoller: ID 1313: Timing: Started: 2015-08-07
   2015-08-07 19:52:16,136 DEBUG
35
                                     pytan.handler.QuestionPoller: ID 1313: Progress: Tested: 2, Passed:
   2015-08-07 19:52:21,142 DEBUG
36
   2015-08-07 19:52:21,142 DEBUG
                                     pytan.handler.QuestionPoller: ID 1313: Timing: Started: 2015-08-07
37
   2015-08-07 19:52:21,142 INFO
                                     pytan.handler.QuestionPoller: ID 1313: Progress Changed 100% (2 of
38
   2015-08-07 19:52:21,142 INFO
                                     pytan.handler.QuestionPoller: ID 1313: Reached Threshold of 99% (2
39
40
   print the export_str returned from export_obj():
41
   Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
42
   2015-08-07 19:49:40,850 DEBUG
                                     pytan.handler.QuestionPoller: ID 1312: id resolved to 1312
43
                                     pytan.handler.QuestionPoller: ID 1312: expiration resolved to 2015-
44
    2015-08-07 19:49:40,850 DEBUG
    2015-08-07 19:49:40,850 DEBUG
                                     pytan.handler.QuestionPoller: ID 1312: query_text resolved to Get C
45
    2015-08-07 19:49:40,850 DEBUG
                                     pytan.handler.QuestionPoller: ID 1312: id resolved to 1312
46
    2015-08-07 19:49:40,850 DEBUG
                                     pytan.handler.QuestionPoller: ID 1312: Object Info resolved to Ques
47
    2015-08-07 19:49:40,853 DEBUG
                                     pytan.handler.QuestionPoller: ID 1312: Progress: Tested: 0, Passed:
48
    2015-08-07 19:49:40,853 DEBUG
                                     pytan.handler.QuestionPoller: ID 1312: Timing: Started: 2015-08-07
49
   2015-08-07 19:49:40,854 INFO
                                     pytan.handler.QuestionPoller: ID 1312: Progress Changed 0% (0 of 2)
50
    2015-08-07 19:49:45,859 DEBUG
                                     pytan.handler.QuestionPoller: ID 1312: Progress: Tested: 0, Passed:
51
   2015-08-07 19:49:45,859 DEBUG
                                     pytan.handler.QuestionPoller: ID 1312: Timing: Started: 2015-08-07
52
   2015-08-07 19:49:50,863 DEBUG
                                     pytan.handler.QuestionPoller: ID 1312: Progress: Tested: 0, Passed:
53
   2015-08-07 19:49:50,863 DEBUG
                                     pytan.handler.QuestionPoller: ID 1312: Timing: Started: 2015-08-07
54
   2015-08-07 19:49:55,870 DEBUG
                                     pytan.handler.QuestionPoller: ID 1312: Progress: Tested: 0, Passed:
55
   2015-08-07 19:49:55,870 DEBUG
                                     pytan.handler.QuestionPoller: ID 1312: Timing: Started: 2015-08-07
56
    ..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)
10
    pytan root dir = os.path.dirname(parent dir)
11
    lib_dir = os.path.join(pytan_root_dir, 'lib')
12
    path_adds = [lib_dir]
13
14
    for aa in path_adds:
15
        if aa not in sys.path:
16
             sys.path.append(aa)
17
18
19
    # connection info for Tanium Server
20
    USERNAME = "Tanium User"
21
    PASSWORD = "T@n!um"
22
    HOST = "172.16.31.128"
23
    PORT = "443"
24
25
    # Logging conrols
26
    LOGLEVEL = 2
27
    DEBUGFORMAT = False
28
29
    import tempfile
30
31
    import pytan
32
    handler = pytan.Handler(
33
34
        username=USERNAME,
        password=PASSWORD,
35
        host=HOST,
36
        port=PORT,
37
        loglevel=LOGLEVEL,
38
        debugformat=DEBUGFORMAT,
39
40
41
    print handler
42
43
    # setup the export_obj kwargs for later
44
    export_kwargs = {}
45
    export_kwargs["export_format"] = u'csv'
46
    export_kwargs["header_sort"] = False
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():"
   if len(out.splitlines()) > 15:
```

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

print out
```

```
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
2
   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
6
    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:
10
    2015-08-07 19:52:26,251 DEBUG
                                     pytan.handler.QuestionPoller: ID 1315: Timing: Started: 2015-08-07
11
    2015-08-07 19:52:31,255 DEBUG
                                     pytan.handler.QuestionPoller: ID 1315: Progress: Tested: 0, Passed:
12
    2015-08-07 19:52:31,255 DEBUG
                                     pytan.handler.QuestionPoller: ID 1315: Timing: Started: 2015-08-07
13
    2015-08-07 19:52:36,264 DEBUG
                                     pytan.handler.QuestionPoller: ID 1315: Progress: Tested: 1, Passed:
14
   2015-08-07 19:52:36,264 DEBUG
                                     pytan.handler.QuestionPoller: ID 1315: Timing: Started: 2015-08-07
15
    2015-08-07 19:52:36,264 INFO
                                     pytan.handler.QuestionPoller: ID 1315: Progress Changed 50% (1 of 2
16
    2015-08-07 19:52:41,271 DEBUG
                                     pytan.handler.QuestionPoller: ID 1315: Progress: Tested: 1, Passed:
17
    2015-08-07 19:52:41,271 DEBUG
                                     pytan.handler.QuestionPoller: ID 1315: Timing: Started: 2015-08-07
18
19
    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
20
    2015-08-07 19:52:51,282 DEBUG
                                     pytan.handler.QuestionPoller: ID 1315: Progress: Tested: 2, Passed:
21
    2015-08-07 19:52:51,283 DEBUG
                                     pytan.handler.QuestionPoller: ID 1315: Timing: Started: 2015-08-07
22
    2015-08-07 19:52:51,283 INFO
                                     pytan.handler.QuestionPoller: ID 1315: Progress Changed 100% (2 of
23
   2015-08-07 19:52:51,283 INFO
                                     pytan.handler.QuestionPoller: ID 1315: Reached Threshold of 99% (2
24
25
   print the export_str returned from export_obj():
26
   Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
27
   2015-08-07 19:51:11,061 DEBUG
                                     pytan.handler.QuestionPoller: ID 1313: id resolved to 1313
28
   2015-08-07 19:51:11,061 DEBUG
                                     pytan.handler.QuestionPoller: ID 1313: expiration resolved to 2015-
29
    2015-08-07 19:51:11,061 DEBUG
                                     pytan.handler.QuestionPoller: ID 1313: query_text resolved to Get C
30
    2015-08-07 19:51:11,061 DEBUG
                                     pytan.handler.QuestionPoller: ID 1313: id resolved to 1313
31
32
    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:
33
    2015-08-07 19:51:11,066 DEBUG
                                     pytan.handler.QuestionPoller: ID 1313: Timing: Started: 2015-08-07
34
    2015-08-07 19:51:11,066 INFO
                                     pytan.handler.QuestionPoller: ID 1313: Progress Changed 0% (0 of 2)
35
    2015-08-07 19:51:16,074 DEBUG
                                     pytan.handler.QuestionPoller: ID 1313: Progress: Tested: 0, Passed:
36
    2015-08-07 19:51:16,074 DEBUG
                                     pytan.handler.QuestionPoller: ID 1313: Timing: Started: 2015-08-07
37
    2015-08-07 19:51:21,079 DEBUG
                                     pytan.handler.QuestionPoller: ID 1313: Progress: Tested: 0, Passed:
38
    2015-08-07 19:51:21,079 DEBUG
                                     pytan.handler.QuestionPoller: ID 1313: Timing: Started: 2015-08-07
39
   2015-08-07 19:51:26,083 DEBUG
                                     pytan.handler.QuestionPoller: ID 1313: Progress: Tested: 0, Passed:
40
   2015-08-07 19:51:26,083 DEBUG
                                     pytan.handler.QuestionPoller: ID 1313: Timing: Started: 2015-08-07
41
    ..trimmed for brevity..
42.
```

Export resultset csv sort list

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

```
import os
    import sys
2
    sys.dont_write_bytecode = True
3
    # Determine our script name, script dir
5
    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)
10
   pytan_root_dir = os.path.dirname(parent_dir)
11
    lib_dir = os.path.join(pytan_root_dir, 'lib')
12
    path_adds = [lib_dir]
13
14
    for aa in path_adds:
15
        if aa not in sys.path:
16
            sys.path.append(aa)
17
18
19
    # connection info for Tanium Server
20
    USERNAME = "Tanium User"
21
    PASSWORD = "T@n!um"
22
    HOST = "172.16.31.128"
23
    PORT = "443"
24
25
26
    # Logging conrols
27
    LOGLEVEL = 2
    DEBUGFORMAT = False
28
29
    import tempfile
30
31
    import pytan
32
33
    handler = pytan.Handler(
        username=USERNAME,
34
        password=PASSWORD,
35
        host=HOST,
36
        port=PORT,
37
        loglevel=LOGLEVEL,
38
39
        debugformat=DEBUGFORMAT,
40
41
    print handler
42
43
    # setup the export_obj kwargs for later
44
    export_kwargs = {}
45
    export_kwargs["export_format"] = u'csv'
46
    export_kwargs["header_sort"] = [u'Computer Name', u'IP Address']
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)
```

```
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
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..')
        out = ' \ n'. join (out)
70
71
   print out
72
```

```
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
4
   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
                                     pytan.handler.QuestionPoller: ID 1316: Progress Changed 0% (0 of 2)
    2015-08-07 19:52:51,392 INFO
    2015-08-07 19:52:56,396 DEBUG
                                     pytan.handler.QuestionPoller: ID 1316: Progress: Tested: 0, Passed:
10
    2015-08-07 19:52:56,396 DEBUG
                                     pytan.handler.QuestionPoller: ID 1316: Timing: Started: 2015-08-07
11
    2015-08-07 19:53:01,404 DEBUG
                                     pytan.handler.QuestionPoller: ID 1316: Progress: Tested: 0, Passed:
12
    2015-08-07 19:53:01,404 DEBUG
                                     pytan.handler.QuestionPoller: ID 1316: Timing: Started: 2015-08-07
13
   2015-08-07 19:53:06,413 DEBUG
                                     pytan.handler.QuestionPoller: ID 1316: Progress: Tested: 1, Passed:
14
                                     pytan.handler.QuestionPoller: ID 1316: Timing: Started: 2015-08-07
   2015-08-07 19:53:06,413 DEBUG
15
   2015-08-07 19:53:06,413 INFO
                                     pytan.handler.QuestionPoller: ID 1316: Progress Changed 50% (1 of 2
16
   2015-08-07 19:53:11,418 DEBUG
                                     pytan.handler.QuestionPoller: ID 1316: Progress: Tested: 2, Passed:
17
   2015-08-07 19:53:11,419 DEBUG
                                     pytan.handler.QuestionPoller: ID 1316: Timing: Started: 2015-08-07
18
   2015-08-07 19:53:11,419 INFO
                                     pytan.handler.QuestionPoller: ID 1316: Progress Changed 100% (2 of
19
   2015-08-07 19:53:11,419 INFO
                                     pytan.handler.QuestionPoller: ID 1316: Reached Threshold of 99% (2
20
21
22
    print the export_str returned from export_obj():
    Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
23
    2015-08-07 19:52:21,243 DEBUG
                                     pytan.handler.QuestionPoller: ID 1315: id resolved to 1315
24
    2015-08-07 19:52:21,243 DEBUG
                                     pytan.handler.QuestionPoller: ID 1315: expiration resolved to 2015-
25
    2015-08-07 19:52:21,243 DEBUG
                                     pytan.handler.QuestionPoller: ID 1315: query_text resolved to Get C
26
    2015-08-07 19:52:21,243 DEBUG
                                     pytan.handler.QuestionPoller: ID 1315: id resolved to 1315
27
    2015-08-07 19:52:21,243 DEBUG
                                     pytan.handler.QuestionPoller: ID 1315: Object Info resolved to Ques
28
    2015-08-07 19:52:21,246 DEBUG
                                     pytan.handler.QuestionPoller: ID 1315: Progress: Tested: 0, Passed:
29
    2015-08-07 19:52:21,246 DEBUG
                                     pytan.handler.QuestionPoller: ID 1315: Timing: Started: 2015-08-07
30
   2015-08-07 19:52:21,246 INFO
                                     pytan.handler.QuestionPoller: ID 1315: Progress Changed 0% (0 of 2)
31
    2015-08-07 19:52:26,251 DEBUG
                                     pytan.handler.QuestionPoller: ID 1315: Progress: Tested: 0, Passed:
32
    2015-08-07 19:52:26,251 DEBUG
                                     pytan.handler.QuestionPoller: ID 1315: Timing: Started: 2015-08-07
33
    2015-08-07 19:52:31,255 DEBUG
                                     pytan.handler.QuestionPoller: ID 1315: Progress: Tested: 0, Passed:
34
    2015-08-07 19:52:31,255 DEBUG
                                     pytan.handler.QuestionPoller: ID 1315: Timing: Started: 2015-08-07
35
    2015-08-07 19:52:36,264 DEBUG
                                     pytan.handler.QuestionPoller: ID 1315: Progress: Tested: 1, Passed:
36
   2015-08-07 19:52:36,264 DEBUG
                                     pytan.handler.QuestionPoller: ID 1315: Timing: Started: 2015-08-07
37
    ..trimmed for brevity..
```

Export resultset csv type false

Export a ResultSet from asking a question as CSV with false for header_add_type

```
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
    # determine the pytan lib dir and add it to the path
    parent_dir = os.path.dirname(my_dir)
10
    pytan_root_dir = os.path.dirname(parent_dir)
11
    lib_dir = os.path.join(pytan_root_dir, 'lib')
12
   path_adds = [lib_dir]
13
14
    for aa in path_adds:
15
        if aa not in sys.path:
16
            sys.path.append(aa)
17
18
19
    # connection info for Tanium Server
20
    USERNAME = "Tanium User"
21
    PASSWORD = "T@n!um"
22
    HOST = "172.16.31.128"
23
    PORT = "443"
24
25
    # Logging conrols
26
    LOGLEVEL = 2
27
   DEBUGFORMAT = False
28
29
   import tempfile
30
31
    import pytan
32
    handler = pytan.Handler(
33
        username=USERNAME,
34
35
        password=PASSWORD,
        host=HOST,
36
        port=PORT,
37
        loglevel=LOGLEVEL,
38
        debugformat=DEBUGFORMAT,
39
40
41
   print handler
42
43
    # setup the export_obj kwargs for later
44
    export_kwargs = {}
45
    export_kwargs["export_format"] = u'csv'
46
    export_kwargs["header_add_type"] = False
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
    # export the object to a string
59
    # (we could just as easily export to a file using export_to_report_file)
60
    export_kwarqs['obj'] = response['question_results']
61
    export_str = handler.export_obj(**export_kwargs)
62
63
    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
72
```

```
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
2
   2015-08-07 19:53:11,519 DEBUG
                                     pytan.handler.QuestionPoller: ID 1318: expiration resolved to 2015-
3
   2015-08-07 19:53:11,519 DEBUG
                                     pytan.handler.QuestionPoller: ID 1318: query_text resolved to Get C
4
   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)
Q
   2015-08-07 19:53:16,530 DEBUG
                                     pytan.handler.QuestionPoller: ID 1318: Progress: Tested: 0, Passed:
10
   2015-08-07 19:53:16,530 DEBUG
                                     pytan.handler.QuestionPoller: ID 1318: Timing: Started: 2015-08-07
11
12
   2015-08-07 19:53:21,538 DEBUG
                                     pytan.handler.QuestionPoller: ID 1318: Progress: Tested: 0, Passed:
                                     pytan.handler.QuestionPoller: ID 1318: Timing: Started: 2015-08-07
   2015-08-07 19:53:21,538 DEBUG
13
   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
15
   2015-08-07 19:53:31,548 DEBUG
                                     pytan.handler.QuestionPoller: ID 1318: Progress: Tested: 0, Passed:
16
   2015-08-07 19:53:31,548 DEBUG
                                     pytan.handler.QuestionPoller: ID 1318: Timing: Started: 2015-08-07
17
   2015-08-07 19:53:36,552 DEBUG
                                     pytan.handler.QuestionPoller: ID 1318: Progress: Tested: 0, Passed:
18
   2015-08-07 19:53:36,552 DEBUG
                                     pytan.handler.QuestionPoller: ID 1318: Timing: Started: 2015-08-07
19
   2015-08-07 19:53:41,559 DEBUG
                                     pytan.handler.QuestionPoller: ID 1318: Progress: Tested: 0, Passed:
20
21
   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:
22
   2015-08-07 19:53:46,566 DEBUG
                                     pytan.handler.QuestionPoller: ID 1318: Timing: Started: 2015-08-07
23
   2015-08-07 19:53:51,570 DEBUG
                                     pytan.handler.QuestionPoller: ID 1318: Progress: Tested: 0, Passed:
24
25
   2015-08-07 19:53:51,570 DEBUG
                                     pytan.handler.QuestionPoller: ID 1318: Timing: Started: 2015-08-07
26
   2015-08-07 19:53:56,577 DEBUG
                                     pytan.handler.QuestionPoller: ID 1318: Progress: Tested: 0, Passed:
27
   2015-08-07 19:53:56,577 DEBUG
                                     pytan.handler.QuestionPoller: ID 1318: Timing: Started: 2015-08-07
   2015-08-07 19:54:01,583 DEBUG
                                     pytan.handler.QuestionPoller: ID 1318: Progress: Tested: 0, Passed:
28
   2015-08-07 19:54:01,583 DEBUG
                                     pytan.handler.QuestionPoller: ID 1318: Timing: Started: 2015-08-07
29
   2015-08-07 19:54:06,590 DEBUG
                                     pytan.handler.QuestionPoller: ID 1318: Progress: Tested: 0, Passed:
30
   2015-08-07 19:54:06,591 DEBUG
                                     pytan.handler.QuestionPoller: ID 1318: Timing: Started: 2015-08-07
31
   2015-08-07 19:54:11,600 DEBUG
                                     pytan.handler.QuestionPoller: ID 1318: Progress: Tested: 0, Passed:
32
   2015-08-07 19:54:11,600 DEBUG
                                     pytan.handler.QuestionPoller: ID 1318: Timing: Started: 2015-08-07
33
                                     pytan.handler.QuestionPoller: ID 1318: Progress: Tested: 0, Passed:
   2015-08-07 19:54:16,608 DEBUG
```

```
2015-08-07 19:54:16,608 DEBUG
                                     pytan.handler.QuestionPoller: ID 1318: Timing: Started: 2015-08-07
35
                                     pytan.handler.QuestionPoller: ID 1318: Progress: Tested: 0, Passed:
    2015-08-07 19:54:21,612 DEBUG
36
    2015-08-07 19:54:21,612 DEBUG
                                     pytan.handler.QuestionPoller: ID 1318: Timing: Started: 2015-08-07
37
                                     pytan.handler.QuestionPoller: ID 1318: Progress: Tested: 0, Passed:
   2015-08-07 19:54:26,621 DEBUG
38
   2015-08-07 19:54:26,621 DEBUG
                                     pytan.handler.QuestionPoller: ID 1318: Timing: Started: 2015-08-07
39
                                     pytan.handler.QuestionPoller: ID 1318: Progress: Tested: 0, Passed:
   2015-08-07 19:54:31,629 DEBUG
   2015-08-07 19:54:31,629 DEBUG
                                     pytan.handler.QuestionPoller: ID 1318: Timing: Started: 2015-08-07
41
   2015-08-07 19:54:36,635 DEBUG
                                     pytan.handler.QuestionPoller: ID 1318: Progress: Tested: 2, Passed:
42
   2015-08-07 19:54:36,635 DEBUG
                                     pytan.handler.QuestionPoller: ID 1318: Timing: Started: 2015-08-07
43
   2015-08-07 19:54:36,635 INFO
                                     pytan.handler.QuestionPoller: ID 1318: Progress Changed 100% (2 of
44
   2015-08-07 19:54:36,635 INFO
                                     pytan.handler.QuestionPoller: ID 1318: Reached Threshold of 99% (2
45
47
    print the export_str returned from export_obj():
    Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
48
    2015-08-07 19:52:51,388 DEBUG
                                     pytan.handler.QuestionPoller: ID 1316: id resolved to 1316
49
    2015-08-07 19:52:51,388 DEBUG
                                     pytan.handler.QuestionPoller: ID 1316: expiration resolved to 2015-
50
   2015-08-07 19:52:51,388 DEBUG
                                     pytan.handler.QuestionPoller: ID 1316: query_text resolved to Get C
51
   2015-08-07 19:52:51,388 DEBUG
                                     pytan.handler.QuestionPoller: ID 1316: id resolved to 1316
52
   2015-08-07 19:52:51,388 DEBUG
                                     pytan.handler.QuestionPoller: ID 1316: Object Info resolved to Ques
53
   2015-08-07 19:52:51,392 DEBUG
                                     pytan.handler.QuestionPoller: ID 1316: Progress: Tested: 0, Passed:
54
                                     pytan.handler.QuestionPoller: ID 1316: Timing: Started: 2015-08-07
   2015-08-07 19:52:51,392 DEBUG
55
   2015-08-07 19:52:51,392 INFO
                                     pytan.handler.QuestionPoller: ID 1316: Progress Changed 0% (0 of 2)
56
   2015-08-07 19:52:56,396 DEBUG
                                     pytan.handler.QuestionPoller: ID 1316: Progress: Tested: 0, Passed:
57
   2015-08-07 19:52:56,396 DEBUG
                                     pytan.handler.QuestionPoller: ID 1316: Timing: Started: 2015-08-07
58
    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
60
    2015-08-07 19:53:06,413 DEBUG
                                     pytan.handler.QuestionPoller: ID 1316: Progress: Tested: 1, Passed:
61
                                     pytan.handler.QuestionPoller: ID 1316: Timing: Started: 2015-08-07
   2015-08-07 19:53:06,413 DEBUG
62
    ..trimmed for brevity..
```

Export resultset csv type true

Export a ResultSet from asking a question as CSV with true for header add type

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

```
# connection info for Tanium Server
20
    USERNAME = "Tanium User"
21
    PASSWORD = "T@n!um"
22
    HOST = "172.16.31.128"
23
    PORT = "443"
24
25
    # Logging conrols
26
    LOGLEVEL = 2
27
    DEBUGFORMAT = False
28
29
    import tempfile
30
31
32
    import pytan
    handler = pytan.Handler(
33
        username=USERNAME,
34
        password=PASSWORD,
35
        host=HOST,
36
        port=PORT,
37
        loglevel=LOGLEVEL,
38
        debugformat=DEBUGFORMAT,
39
40
41
    print handler
42
43
44
    # setup the export_obj kwargs for later
45
    export_kwargs = {}
    export_kwargs["export_format"] = u'csv'
46
    export_kwargs["header_add_type"] = True
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
    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
```

```
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
2
   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
6
    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:
10
                                     pytan.handler.QuestionPoller: ID 1319: Timing: Started: 2015-08-07
11
   2015-08-07 19:54:41,761 DEBUG
   2015-08-07 19:54:46,767 DEBUG
                                     pytan.handler.QuestionPoller: ID 1319: Progress: Tested: 0, Passed:
12
   2015-08-07 19:54:46,767 DEBUG
                                     pytan.handler.QuestionPoller: ID 1319: Timing: Started: 2015-08-07
13
   2015-08-07 19:54:51,771 DEBUG
                                     pytan.handler.QuestionPoller: ID 1319: Progress: Tested: 0, Passed:
14
   2015-08-07 19:54:51,771 DEBUG
                                     pytan.handler.QuestionPoller: ID 1319: Timing: Started: 2015-08-07
15
   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
17
    2015-08-07 19:54:56,775 INFO
                                     pytan.handler.QuestionPoller: ID 1319: Progress Changed 100% (2 of
18
    2015-08-07 19:54:56,775 INFO
                                     pytan.handler.QuestionPoller: ID 1319: Reached Threshold of 99% (2
19
20
    print the export_str returned from export_obj():
21
   Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
22
   2015-08-07 19:53:11,519 DEBUG
                                     pytan.handler.QuestionPoller: ID 1318: id resolved to 1318
23
24
    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
25
   2015-08-07 19:53:11,519 DEBUG
                                     pytan.handler.QuestionPoller: ID 1318: id resolved to 1318
26
   2015-08-07 19:53:11,519 DEBUG
                                     pytan.handler.QuestionPoller: ID 1318: Object Info resolved to Ques
27
   2015-08-07 19:53:11,522 DEBUG
                                     pytan.handler.QuestionPoller: ID 1318: Progress: Tested: 0, Passed:
28
   2015-08-07 19:53:11,522 DEBUG
                                     pytan.handler.QuestionPoller: ID 1318: Timing: Started: 2015-08-07
29
    2015-08-07 19:53:11,522 INFO
                                     pytan.handler.QuestionPoller: ID 1318: Progress Changed 0% (0 of 2)
30
    2015-08-07 19:53:16,530 DEBUG
                                     pytan.handler.QuestionPoller: ID 1318: Progress: Tested: 0, Passed:
31
    2015-08-07 19:53:16,530 DEBUG
                                     pytan.handler.QuestionPoller: ID 1318: Timing: Started: 2015-08-07
32
    2015-08-07 19:53:21,538 DEBUG
                                     pytan.handler.QuestionPoller: ID 1318: Progress: Tested: 0, Passed:
33
    2015-08-07 19:53:21,538 DEBUG
                                     pytan.handler.QuestionPoller: ID 1318: Timing: Started: 2015-08-07
34
   2015-08-07 19:53:26,543 DEBUG
                                     pytan.handler.QuestionPoller: ID 1318: Progress: Tested: 0, Passed:
35
   2015-08-07 19:53:26,543 DEBUG
                                     pytan.handler.QuestionPoller: ID 1318: Timing: Started: 2015-08-07
36
   ..trimmed for brevity..
```

Export resultset csv sensor false

Export a ResultSet from asking a question as CSV with false for header_add_sensor

```
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]
13
14
    for aa in path_adds:
15
        if aa not in sys.path:
16
17
            sys.path.append(aa)
18
19
    # connection info for Tanium Server
20
    USERNAME = "Tanium User"
21
    PASSWORD = "T@n!um"
22
    HOST = "172.16.31.128"
23
    PORT = "443"
24
25
    # Logging conrols
26
    LOGLEVEL = 2
27
    DEBUGFORMAT = False
28
29
    import tempfile
30
31
    import pytan
32
   handler = pytan.Handler(
33
        username=USERNAME,
34
        password=PASSWORD,
35
36
        host=HOST,
        port=PORT,
37
        loglevel=LOGLEVEL,
38
        debugformat=DEBUGFORMAT,
39
40
41
    print handler
42
43
    # setup the export_obj kwargs for later
44
    export_kwargs = {}
45
    export_kwargs["export_format"] = u'csv'
46
    export_kwargs["header_add_sensor"] = False
47
48
    # ask the question that will provide the resultset that we want to use
49
50
    ask_kwargs = {
        '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
    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..')
```

```
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:
10
    2015-08-07 19:55:01,930 DEBUG
                                     pytan.handler.QuestionPoller: ID 1320: Timing: Started: 2015-08-07
11
    2015-08-07 19:55:06,936 DEBUG
                                     pytan.handler.QuestionPoller: ID 1320: Progress: Tested: 0, Passed:
12
   2015-08-07 19:55:06,936 DEBUG
                                     pytan.handler.QuestionPoller: ID 1320: Timing: Started: 2015-08-07
13
    2015-08-07 19:55:11,944 DEBUG
                                     pytan.handler.QuestionPoller: ID 1320: Progress: Tested: 0, Passed:
14
    2015-08-07 19:55:11,944 DEBUG
                                     pytan.handler.QuestionPoller: ID 1320: Timing: Started: 2015-08-07
15
    2015-08-07 19:55:16,949 DEBUG
                                     pytan.handler.QuestionPoller: ID 1320: Progress: Tested: 0, Passed:
16
   2015-08-07 19:55:16,950 DEBUG
                                     pytan.handler.QuestionPoller: ID 1320: Timing: Started: 2015-08-07
17
    2015-08-07 19:55:21,957 DEBUG
                                     pytan.handler.QuestionPoller: ID 1320: Progress: Tested: 1, Passed:
18
    2015-08-07 19:55:21,957 DEBUG
                                     pytan.handler.QuestionPoller: ID 1320: Timing: Started: 2015-08-07
19
    2015-08-07 19:55:21,957 INFO
                                     pytan.handler.QuestionPoller: ID 1320: Progress Changed 50% (1 of 2
20
21
    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
22
    2015-08-07 19:55:26,962 INFO
                                     pytan.handler.QuestionPoller: ID 1320: Progress Changed 100% (2 of
23
    2015-08-07 19:55:26,962 INFO
                                     pytan.handler.QuestionPoller: ID 1320: Reached Threshold of 99% (2
24
25
   print the export_str returned from export_obj():
26
   Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
27
   2015-08-07 19:54:36,751 DEBUG
                                     pytan.handler.QuestionPoller: ID 1319: id resolved to 1319
28
   2015-08-07 19:54:36,751 DEBUG
                                     pytan.handler.QuestionPoller: ID 1319: expiration resolved to 2015-
29
   2015-08-07 19:54:36,751 DEBUG
                                     pytan.handler.QuestionPoller: ID 1319: query_text resolved to Get C
30
   2015-08-07 19:54:36,751 DEBUG
                                     pytan.handler.QuestionPoller: ID 1319: id resolved to 1319
31
    2015-08-07 19:54:36,751 DEBUG
                                     pytan.handler.QuestionPoller: ID 1319: Object Info resolved to Ques
32
   2015-08-07 19:54:36,754 DEBUG
                                     pytan.handler.QuestionPoller: ID 1319: Progress: Tested: 0, Passed:
33
34
    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)
35
    2015-08-07 19:54:41,761 DEBUG
                                     pytan.handler.QuestionPoller: ID 1319: Progress: Tested: 0, Passed:
36
    2015-08-07 19:54:41,761 DEBUG
                                     pytan.handler.QuestionPoller: ID 1319: Timing: Started: 2015-08-07
37
    2015-08-07 19:54:46,767 DEBUG
                                     pytan.handler.QuestionPoller: ID 1319: Progress: Tested: 0, Passed:
38
                                     pytan.handler.QuestionPoller: ID 1319: Timing: Started: 2015-08-07
    2015-08-07 19:54:46,767 DEBUG
39
    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
41
    ..trimmed for brevity..
42.
```

Export resultset csv sensor true

Export a ResultSet from asking a question as CSV with true for header_add_sensor

```
import os
    import sys
2
    sys.dont_write_bytecode = True
    # Determine our script name, script dir
5
    my_file = os.path.abspath(sys.argv[0])
6
    my_dir = os.path.dirname(my_file)
7
    # determine the pytan lib dir and add it to the path
10
    parent_dir = os.path.dirname(my_dir)
   pytan_root_dir = os.path.dirname(parent_dir)
    lib_dir = os.path.join(pytan_root_dir, 'lib')
12
   path_adds = [lib_dir]
13
14
   for aa in path_adds:
15
        if aa not in sys.path:
16
            sys.path.append(aa)
17
18
19
    # connection info for Tanium Server
20
    USERNAME = "Tanium User"
21
    PASSWORD = "T@n!um"
22
    HOST = "172.16.31.128"
23
    PORT = "443"
24
25
    # Logging conrols
26
    LOGLEVEL = 2
27
   DEBUGFORMAT = False
28
29
    import tempfile
30
31
    import pytan
32
    handler = pytan.Handler(
33
        username=USERNAME,
34
        password=PASSWORD,
35
36
        host=HOST,
37
        port=PORT,
        loglevel=LOGLEVEL,
        debugformat=DEBUGFORMAT,
39
40
41
   print handler
42
43
    # setup the export_obj kwargs for later
    export_kwargs = {}
45
    export_kwarqs["export_format"] = u'csv'
46
    export_kwargs["header_add_sensor"] = True
47
48
49
    # ask the question that will provide the resultset that we want to use
50
    ask_kwargs = {
51
        'qtype': 'manual',
        'sensors': [
52
             "Computer Name", "IP Route Details", "IP Address",
53
             'Folder Name Search with RegEx Match{dirname=Program Files,regex=.*Shared.*}',
54
55
        ],
56
57
    response = handler.ask(**ask_kwargs)
```

```
# 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
   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
72
   print out
```

```
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
2
    2015-08-07 19:55:27,070 DEBUG
                                     pytan.handler.QuestionPoller: ID 1321: expiration resolved to 2015-
3
    2015-08-07 19:55:27,070 DEBUG
                                     pytan.handler.QuestionPoller: ID 1321: query_text resolved to Get C
4
    2015-08-07 19:55:27,070 DEBUG
                                     pytan.handler.QuestionPoller: ID 1321: id resolved to 1321
5
    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:
10
    2015-08-07 19:55:32,077 DEBUG
                                     pytan.handler.QuestionPoller: ID 1321: Timing: Started: 2015-08-07
11
    2015-08-07 19:55:37,083 DEBUG
                                     pytan.handler.QuestionPoller: ID 1321: Progress: Tested: 0, Passed:
12
    2015-08-07 19:55:37,083 DEBUG
                                     pytan.handler.QuestionPoller: ID 1321: Timing: Started: 2015-08-07
13
    2015-08-07 19:55:42,089 DEBUG
                                     pytan.handler.QuestionPoller: ID 1321: Progress: Tested: 0, Passed:
14
    2015-08-07 19:55:42,089 DEBUG
                                     pytan.handler.QuestionPoller: ID 1321: Timing: Started: 2015-08-07
15
    2015-08-07 19:55:47,097 DEBUG
                                     pytan.handler.QuestionPoller: ID 1321: Progress: Tested: 0, Passed:
16
    2015-08-07 19:55:47,097 DEBUG
                                     pytan.handler.QuestionPoller: ID 1321: Timing: Started: 2015-08-07
17
18
    2015-08-07 19:55:52,105 DEBUG
                                     pytan.handler.QuestionPoller: ID 1321: Progress: Tested: 0, Passed:
                                     pytan.handler.QuestionPoller: ID 1321: Timing: Started: 2015-08-07
19
   2015-08-07 19:55:52,105 DEBUG
   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
21
   2015-08-07 19:56:02,119 DEBUG
                                     pytan.handler.QuestionPoller: ID 1321: Progress: Tested: 0, Passed:
22
   2015-08-07 19:56:02,119 DEBUG
                                     pytan.handler.QuestionPoller: ID 1321: Timing: Started: 2015-08-07
23
   2015-08-07 19:56:07,129 DEBUG
                                     pytan.handler.QuestionPoller: ID 1321: Progress: Tested: 0, Passed:
24
   2015-08-07 19:56:07,129 DEBUG
                                     pytan.handler.QuestionPoller: ID 1321: Timing: Started: 2015-08-07
25
    2015-08-07 19:56:12,138 DEBUG
                                     pytan.handler.QuestionPoller: ID 1321: Progress: Tested: 0, Passed:
26
27
    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:
28
    2015-08-07 19:56:17,145 DEBUG
                                     pytan.handler.QuestionPoller: ID 1321: Timing: Started: 2015-08-07
29
    2015-08-07 19:56:22,152 DEBUG
                                     pytan.handler.QuestionPoller: ID 1321: Progress: Tested: 0, Passed:
30
                                     pytan.handler.QuestionPoller: ID 1321: Timing: Started: 2015-08-07
31
    2015-08-07 19:56:22,152 DEBUG
   2015-08-07 19:56:27,160 DEBUG
                                     pytan.handler.QuestionPoller: ID 1321: Progress: Tested: 0, Passed:
32
                                     pytan.handler.QuestionPoller: ID 1321: Timing: Started: 2015-08-07
33
   2015-08-07 19:56:27,160 DEBUG
    2015-08-07 19:56:32,167 DEBUG
                                     pytan.handler.QuestionPoller: ID 1321: Progress: Tested: 2, Passed:
34
   2015-08-07 19:56:32,167 DEBUG
                                     pytan.handler.QuestionPoller: ID 1321: Timing: Started: 2015-08-07
35
    2015-08-07 19:56:32,167 INFO
                                     pytan.handler.OuestionPoller: ID 1321: Progress Changed 100% (2 of
36
   2015-08-07 19:56:32,167 INFO
                                     pytan.handler.QuestionPoller: ID 1321: Reached Threshold of 99% (2
37
38
   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
41
    2015-08-07 19:54:56,919 DEBUG
                                     pytan.handler.QuestionPoller: ID 1320: expiration resolved to 2015-
42
    2015-08-07 19:54:56,919 DEBUG
                                     pytan.handler.QuestionPoller: ID 1320: query_text resolved to Get C
43
   2015-08-07 19:54:56,919 DEBUG
                                     pytan.handler.QuestionPoller: ID 1320: id resolved to 1320
44
   2015-08-07 19:54:56,919 DEBUG
                                     pytan.handler.QuestionPoller: ID 1320: Object Info resolved to Ques
45
   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
47
   2015-08-07 19:54:56,923 INFO
                                     pytan.handler.QuestionPoller: ID 1320: Progress Changed 0% (0 of 2)
48
   2015-08-07 19:55:01,930 DEBUG
                                     pytan.handler.QuestionPoller: ID 1320: Progress: Tested: 0, Passed:
49
   2015-08-07 19:55:01,930 DEBUG
                                     pytan.handler.QuestionPoller: ID 1320: Timing: Started: 2015-08-07
50
   2015-08-07 19:55:06,936 DEBUG
                                     pytan.handler.QuestionPoller: ID 1320: Progress: Tested: 0, Passed:
51
   2015-08-07 19:55:06,936 DEBUG
                                     pytan.handler.QuestionPoller: ID 1320: Timing: Started: 2015-08-07
52
    2015-08-07 19:55:11,944 DEBUG
                                     pytan.handler.QuestionPoller: ID 1320: Progress: Tested: 0, Passed:
53
   2015-08-07 19:55:11,944 DEBUG
                                     pytan.handler.QuestionPoller: ID 1320: Timing: Started: 2015-08-07
54
    ..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

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

```
import pytan
32
    handler = pytan.Handler(
33
        username=USERNAME,
34
        password=PASSWORD,
35
        host=HOST,
        port=PORT,
37
        loglevel=LOGLEVEL,
38
        debugformat=DEBUGFORMAT,
39
40
41
    print handler
42
44
    # setup the export_obj kwargs for later
45
    export_kwargs = {}
    export_kwargs["export_format"] = u'csv'
46
    export_kwargs["header_sort"] = [[]]
47
48
    # ask the question that will provide the resultset that we want to use
49
    ask_kwargs = {
50
        'qtype': 'manual',
51
        'sensors': [
52
             "Computer Name"
53
        ],
54
55
    response = handler.ask(**ask_kwargs)
    export_kwargs['obj'] = response['question_results']
57
58
    # export the object to a string
59
    # this should throw an exception: pytan.exceptions.HandlerError
60
    import traceback
61
62
    try:
63
        handler.export_obj(**export_kwargs)
65
    except Exception as e:
        traceback.print_exc(file=sys.stdout)
66
```

```
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
2
   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
                                     pytan.handler.QuestionPoller: ID 1323: Object Info resolved to Ques
   2015-08-07 19:56:32,279 DEBUG
   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)
Q
10
   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
11
                                     pytan.handler.QuestionPoller: ID 1323: Timing: Started: 2015-08-07
12
   2015-08-07 19:56:37,290 INFO
                                     pytan.handler.QuestionPoller: ID 1323: Progress Changed 100% (2 of
   2015-08-07 19:56:37,290 INFO
                                     pytan.handler.QuestionPoller: ID 1323: Reached Threshold of 99% (2
13
   Traceback (most recent call last):
14
     File "<string>", line 65, in <module>
15
     File "/Users/jolsen/gh/pytan/lib/pytan/utils.py", line 2699, in wrap
16
17
       ret = f(*args, **kwargs)
     File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 1084, in export_obj
18
       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
5
    my_file = os.path.abspath(sys.argv[0])
6
    my_dir = os.path.dirname(my_file)
    # determine the pytan lib dir and add it to the path
9
    parent_dir = os.path.dirname(my_dir)
10
   pytan_root_dir = os.path.dirname(parent_dir)
11
    lib_dir = os.path.join(pytan_root_dir, 'lib')
12
   path_adds = [lib_dir]
13
14
    for aa in path_adds:
15
        if aa not in sys.path:
16
17
             sys.path.append(aa)
18
19
    # connection info for Tanium Server
20
    USERNAME = "Tanium User"
21
    PASSWORD = "T@n!um"
22
    HOST = "172.16.31.128"
23
   PORT = "443"
24
25
26
    # Logging conrols
27
    LOGLEVEL = 2
   DEBUGFORMAT = False
28
29
    import tempfile
30
31
32
    import pytan
33
    handler = pytan.Handler(
        username=USERNAME,
34
        password=PASSWORD,
35
        host=HOST.
36
        port=PORT,
37
        loglevel=LOGLEVEL,
38
        debugformat=DEBUGFORMAT,
40
41
   print handler
42
43
    # setup the export_obj kwargs for later
44
   export_kwargs = {}
```

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

```
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
2
    2015-08-07 19:56:37,342 DEBUG
                                     pytan.handler.QuestionPoller: ID 1324: expiration resolved to 2015-
3
    2015-08-07 19:56:37,342 DEBUG
                                     pytan.handler.QuestionPoller: ID 1324: query_text resolved to Get (
4
    2015-08-07 19:56:37,342 DEBUG
                                     pytan.handler.QuestionPoller: ID 1324: id resolved to 1324
5
    2015-08-07 19:56:37,342 DEBUG
                                     pytan.handler.QuestionPoller: ID 1324: Object Info resolved to Ques
6
   2015-08-07 19:56:37,345 DEBUG
                                     pytan.handler.QuestionPoller: ID 1324: Progress: Tested: 0, Passed:
                                     pytan.handler.QuestionPoller: ID 1324: Timing: Started: 2015-08-07
   2015-08-07 19:56:37,345 DEBUG
   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:
10
                                     pytan.handler.QuestionPoller: ID 1324: Timing: Started: 2015-08-07
   2015-08-07 19:56:42,353 DEBUG
11
   2015-08-07 19:56:47,361 DEBUG
                                     pytan.handler.QuestionPoller: ID 1324: Progress: Tested: 1, Passed:
12
   2015-08-07 19:56:47,361 DEBUG
                                     pytan.handler.QuestionPoller: ID 1324: Timing: Started: 2015-08-07
13
   2015-08-07 19:56:47,361 INFO
                                     pytan.handler.QuestionPoller: ID 1324: Progress Changed 50% (1 of 2
14
   2015-08-07 19:56:52,368 DEBUG
                                     pytan.handler.QuestionPoller: ID 1324: Progress: Tested: 2, Passed:
15
    2015-08-07 19:56:52,368 DEBUG
                                     pytan.handler.QuestionPoller: ID 1324: Timing: Started: 2015-08-07
16
    2015-08-07 19:56:52,368 INFO
                                     pytan.handler.QuestionPoller: ID 1324: Progress Changed 100% (2 of
17
    2015-08-07 19:56:52,368 INFO
                                     pytan.handler.QuestionPoller: ID 1324: Reached Threshold of 99% (2
18
   Traceback (most recent call last):
19
     File "<string>", line 65, in <module>
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 1084, in export_obj
23
        pytan.utils.check_dictkey(**check_args)
24
     File "/Users/jolsen/gh/pytan/lib/pytan/utils.py", line 2685, in check_dictkey
25
        raise pytan.exceptions.HandlerError(err(key, valid_types, k_type))
26
   HandlerError: 'header_sort' must be one of [<type 'bool'>, <type 'list'>, <type 'tuple'>], you suppl
27
```

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
2
    sys.dont_write_bytecode = True
3
    # Determine our script name, script dir
5
    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)
10
   pytan_root_dir = os.path.dirname(parent_dir)
11
    lib_dir = os.path.join(pytan_root_dir, 'lib')
12
    path_adds = [lib_dir]
13
14
    for aa in path_adds:
15
        if aa not in sys.path:
16
            sys.path.append(aa)
17
18
19
    # connection info for Tanium Server
20
    USERNAME = "Tanium User"
21
    PASSWORD = "T@n!um"
22
    HOST = "172.16.31.128"
23
    PORT = "443"
24
25
26
    # Logging conrols
    LOGLEVEL = 2
27
    DEBUGFORMAT = False
28
29
    import tempfile
30
31
    import pytan
32
    handler = pytan.Handler(
33
        username=USERNAME,
34
        password=PASSWORD,
35
        host=HOST,
36
        port=PORT,
37
        loglevel=LOGLEVEL,
38
39
        debugformat=DEBUGFORMAT,
41
    print handler
42
43
    # setup the export_obj kwargs for later
44
    export_kwargs = {}
45
    export_kwargs["export_format"] = u'csv'
46
    export_kwargs["expand_grouped_columns"] = u'bad'
47
48
    # ask the question that will provide the resultset that we want to use
49
    ask_kwargs = {
50
        'qtype': 'manual',
51
        'sensors': [
52
             "Computer Name"
53
54
        ],
55
    response = handler.ask(**ask_kwargs)
56
   export_kwargs['obj'] = response['question_results']
```

```
# 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)
```

```
Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
   2015-08-07 19:56:52,416 DEBUG
2
                                     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-
3
   2015-08-07 19:56:52,416 DEBUG
                                     pytan.handler.QuestionPoller: ID 1325: query_text resolved to Get C
4
   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:
                                     pytan.handler.QuestionPoller: ID 1325: Timing: Started: 2015-08-07
   2015-08-07 19:56:52,420 DEBUG
   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:
10
   2015-08-07 19:56:57,424 DEBUG
                                     pytan.handler.QuestionPoller: ID 1325: Timing: Started: 2015-08-07
11
   2015-08-07 19:56:57,424 INFO
                                     pytan.handler.QuestionPoller: ID 1325: Progress Changed 50% (1 of 2
12
   2015-08-07 19:57:02,429 DEBUG
                                     pytan.handler.QuestionPoller: ID 1325: Progress: Tested: 2, Passed:
13
   2015-08-07 19:57:02,429 DEBUG
                                     pytan.handler.QuestionPoller: ID 1325: Timing: Started: 2015-08-07
14
                                     pytan.handler.QuestionPoller: ID 1325: Progress Changed 100% (2 of
15
   2015-08-07 19:57:02,429 INFO
   2015-08-07 19:57:02,429 INFO
                                     pytan.handler.QuestionPoller: ID 1325: Reached Threshold of 99% (2
16
   Traceback (most recent call last):
17
     File "<string>", line 65, in <module>
18
     File "/Users/jolsen/gh/pytan/lib/pytan/utils.py", line 2699, in wrap
19
       ret = f(*args, **kwargs)
20
     File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 1084, in export_obj
21
       pytan.utils.check_dictkey(**check_args)
22
     File "/Users/jolsen/qh/pytan/lib/pytan/utils.py", line 2685, in check_dictkey
23
       raise pytan.exceptions.HandlerError(err(key, valid_types, k_type))
24
   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

```
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)
11
    lib_dir = os.path.join(pytan_root_dir, 'lib')
12
    path_adds = [lib_dir]
13
14
    for aa in path_adds:
15
        if aa not in sys.path:
             sys.path.append(aa)
17
18
19
    # connection info for Tanium Server
20
    USERNAME = "Tanium User"
21
    PASSWORD = "T@n!um"
22
    HOST = "172.16.31.128"
23
    PORT = "443"
24
25
    # Logging conrols
26
    LOGLEVEL = 2
27
    DEBUGFORMAT = False
28
29
    import tempfile
30
31
    import pytan
32
    handler = pytan.Handler(
33
        username=USERNAME,
34
35
        password=PASSWORD,
        host=HOST,
36
        port=PORT,
37
        loglevel=LOGLEVEL,
38
        debugformat=DEBUGFORMAT,
39
40
41
    print handler
42
43
    # setup the export_obj kwargs for later
44
    export_kwarqs = {}
45
    export_kwargs["export_format"] = u'csv'
46
    export_kwargs["sensors"] = [[]]
47
    export_kwargs["header_add_sensor"] = True
48
    # ask the question that will provide the resultset that we want to use
50
    ask_kwargs = {
51
         'qtype': 'manual',
52
         'sensors': [
53
             "Computer Name"
54
55
    response = handler.ask(**ask_kwargs)
    export_kwargs['obj'] = response['question_results']
58
59
    # export the object to a string
60
    # this should throw an exception: pytan.exceptions.HandlerError
61
62
    import traceback
63
    try:
64
        handler.export_obj(**export_kwargs)
65
    except Exception as e:
66
        traceback.print_exc(file=sys.stdout)
67
```

```
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
2
   2015-08-07 19:57:02,479 DEBUG
                                     pytan.handler.QuestionPoller: ID 1326: expiration resolved to 2015-
3
   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:
10
   2015-08-07 19:57:07,486 DEBUG
                                     pytan.handler.QuestionPoller: ID 1326: Timing: Started: 2015-08-07
11
   2015-08-07 19:57:12,490 DEBUG
                                     pytan.handler.QuestionPoller: ID 1326: Progress: Tested: 2, Passed:
12
   2015-08-07 19:57:12,490 DEBUG
                                     pytan.handler.QuestionPoller: ID 1326: Timing: Started: 2015-08-07
13
   2015-08-07 19:57:12,490 INFO
                                     pytan.handler.QuestionPoller: ID 1326: Progress Changed 100% (2 of
14
   2015-08-07 19:57:12,490 INFO
                                     pytan.handler.QuestionPoller: ID 1326: Reached Threshold of 99% (2
15
   Traceback (most recent call last):
16
     File "<string>", line 66, in <module>
17
     File "/Users/jolsen/gh/pytan/lib/pytan/utils.py", line 2699, in wrap
18
       ret = f(*args, **kwargs)
19
     File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 1084, in export_obj
20
       pytan.utils.check_dictkey(**check_args)
21
     File "/Users/jolsen/qh/pytan/lib/pytan/utils.py", line 2692, in check_dictkey
22
       raise pytan.exceptions.HandlerError(err(key, valid_list_types, list_types))
23
   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

```
import os
    import sys
2
    sys.dont_write_bytecode = True
3
    # Determine our script name, script dir
5
    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)
10
    pytan_root_dir = os.path.dirname(parent_dir)
11
    lib_dir = os.path.join(pytan_root_dir, 'lib')
12
    path_adds = [lib_dir]
13
14
    for aa in path_adds:
15
        if aa not in sys.path:
16
            sys.path.append(aa)
17
18
19
    # connection info for Tanium Server
20
21
    USERNAME = "Tanium User"
   PASSWORD = "T@n!um"
22
   HOST = "172.16.31.128"
```

```
PORT = "443"
24
25
26
    # Logging conrols
    LOGLEVEL = 2
27
    DEBUGFORMAT = False
28
29
    import tempfile
30
31
    import pytan
32
    handler = pytan.Handler(
33
        username=USERNAME,
34
        password=PASSWORD,
35
        host=HOST,
36
37
        port=PORT,
        loglevel=LOGLEVEL,
38
        debugformat=DEBUGFORMAT,
39
40
41
    print handler
42
43
    # setup the export_obj kwargs for later
44
    export_kwargs = {}
45
    export_kwargs["export_format"] = u'bad'
46
47
    # ask the question that will provide the resultset that we want to use
48
49
    ask_kwargs = {
         'qtype': 'manual',
50
         'sensors': [
51
             "Computer Name"
52
53
        ],
54
    response = handler.ask(**ask_kwargs)
55
    export_kwargs['obj'] = response['question_results']
56
57
    # export the object to a string
58
    # this should throw an exception: pytan.exceptions.HandlerError
59
    import traceback
60
61
62
        handler.export_obj(**export_kwargs)
63
    except Exception as e:
64
        traceback.print_exc(file=sys.stdout)
65
```

```
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
2
   2015-08-07 19:57:12,541 DEBUG
                                    pytan.handler.QuestionPoller: ID 1327: expiration resolved to 2015-
3
   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
6
   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:
10
   2015-08-07 19:57:17,551 DEBUG
                                     pytan.handler.QuestionPoller: ID 1327: Timing: Started: 2015-08-07
11
   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
13
    2015-08-07 19:57:22,556 INFO
                                     pytan.handler.QuestionPoller: ID 1327: Progress Changed 100% (2 of
14
   2015-08-07 19:57:22,556 INFO
                                     pytan.handler.QuestionPoller: ID 1327: Reached Threshold of 99% (2
15
   Traceback (most recent call last):
16
     File "<string>", line 64, in <module>
17
     File "/Users/jolsen/gh/pytan/lib/pytan/utils.py", line 2699, in wrap
18
       ret = f(*args, **kwargs)
19
     File "/Users/jolsen/gh/pytan/lib/pytan/handler.py", line 1078, in export_obj
20
       raise pytan.exceptions.HandlerError(err)
21
   HandlerError: u'bad' not a supported export format for ResultSet, must be one of: json, csv
22
```

PyTan API Valid Export BaseType Examples

Export basetype csv default options

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

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

```
port=PORT,
37
        loglevel=LOGLEVEL,
38
        debugformat=DEBUGFORMAT,
39
40
41
    print handler
42
43
    # setup the export_obj kwargs for later
44
    export_kwarqs = {}
45
    export_kwargs["export_format"] = u'csv'
46
47
    # get the objects that will provide the basetype that we want to use
48
49
    get_kwargs = {
         'name': [
50
             "Computer Name", "IP Route Details", "IP Address",
51
             'Folder Name Search with RegEx Match',
52
53
         'objtype': 'sensor',
54
55
    response = handler.get(**get_kwargs)
56
57
    # export the object to a string
58
    # (we could just as easily export to a file using export_to_report_file)
59
    export_kwargs['obj'] = response
60
    export_str = handler.export_obj(**export_kwargs)
61
62
63
    print ""
64
    print "print the export_str returned from export_obj():"
65
66
    out = export_str
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
73
    print out
```

```
Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2
   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.
5
   Example: workstation-1.company.com",0,3409330187,0,3,1,,86400,,,,,Computer Name,,Windows,select CSNa
6
   Network, 2015-08-07T13:22:12, |, "Returns IPv4 network routes, filtered to exclude noise. With Flags, N
7
   Example: 172.16.0.0|192.168.1.1|255.255.0.0|UG|100|eth0",1,435227963,0,552,1,Jim Olseh,60,0,defined
9
   Set objWMIService = GetObject(" winmgmts:" _
       & " {impersonationLevel=impersonate}!\\" & strComputer & " \root\cimv2&
11
   Set collip = objWMIService.ExecQuery(" select * from win32_networkadapterconfiguration where IPE
12
   dim ipaddrs()
13
   ipcount = 0
14
15
   for each ipItem in collip
       for each ipaddr in ipItem.IPAddress
16
           ipcount = ipcount + 1
17
```

```
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
2
    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
9
    parent_dir = os.path.dirname(my_dir)
10
    pytan_root_dir = os.path.dirname(parent_dir)
11
    lib_dir = os.path.join(pytan_root_dir, 'lib')
12
13
   path_adds = [lib_dir]
   for aa in path_adds:
15
        if aa not in sys.path:
16
            sys.path.append(aa)
17
18
19
    # connection info for Tanium Server
20
    USERNAME = "Tanium User"
21
    PASSWORD = "T@n!um"
22
    HOST = "172.16.31.128"
23
    PORT = "443"
24
25
    # Logging conrols
27
    LOGLEVEL = 2
   DEBUGFORMAT = False
28
29
   import tempfile
30
31
32
    import pytan
33
    handler = pytan.Handler(
34
        username=USERNAME,
        password=PASSWORD,
35
        host=HOST,
36
        port=PORT,
37
38
        loglevel=LOGLEVEL,
        debugformat=DEBUGFORMAT,
41
   print handler
42.
43
44
    # setup the export_obj kwargs for later
45
    export_kwargs = {}
```

export_kwargs["export_format"] = u'json'

```
export_kwarqs["include_type"] = False
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_kwarqs['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
74
```

```
Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2
   print the export_str returned from export_obj():
3
4
      "sensor": [
5
6
          "category": "Reserved",
          "description": "The assigned name of the client machine.\nExample: workstation-1.company.com",
          "exclude_from_parse_flag": 0,
          "hash": 3409330187,
10
          "hidden_flag": 0,
11
          "id": 3,
12
          "ignore_case_flag": 1,
13
          "max_age_seconds": 86400,
14
          "name": "Computer Name",
15
          "queries": {
16
            "query": [
17
18
    ..trimmed for brevity..
19
```

Export basetype json explode false

Export a BaseType from getting objects as JSON with false for explode_json_string_values

```
import os
    import sys
2
    sys.dont_write_bytecode = True
3
    # Determine our script name, script dir
5
   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)
10
   pytan_root_dir = os.path.dirname(parent_dir)
11
    lib_dir = os.path.join(pytan_root_dir, 'lib')
12
    path_adds = [lib_dir]
13
14
    for aa in path_adds:
15
        if aa not in sys.path:
16
            sys.path.append(aa)
17
18
19
    # connection info for Tanium Server
20
    USERNAME = "Tanium User"
21
    PASSWORD = "T@n!um"
22
    HOST = "172.16.31.128"
23
    PORT = "443"
24
25
26
    # Logging conrols
27
    LOGLEVEL = 2
    DEBUGFORMAT = False
28
29
    import tempfile
30
31
    import pytan
32
33
    handler = pytan.Handler(
        username=USERNAME,
34
        password=PASSWORD,
35
        host=HOST,
36
        port=PORT,
37
        loglevel=LOGLEVEL,
38
39
        debugformat=DEBUGFORMAT,
40
41
    print handler
42
43
    # setup the export_obj kwargs for later
44
    export_kwargs = {}
45
    export_kwargs["export_format"] = u'json'
46
    export_kwargs["explode_json_string_values"] = False
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)
```

```
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
64
    print ""
65
    print "print the export_str returned from export_obj():"
66
    out = export_str
68
    if len(out.splitlines()) > 15:
        out = out.splitlines()[0:15]
70
        out.append('..trimmed for brevity..')
71
        out = ' \ n'. join (out)
72
73
   print out
```

```
Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2
   print the export_str returned from export_obj():
3
4
      "_type": "sensors",
5
      "sensor": [
6
7
          "_type": "sensor",
8
          "category": "Reserved",
9
          "description": "The assigned name of the client machine.\nExample: workstation-1.company.com",
10
          "exclude_from_parse_flag": 0,
11
          "hash": 3409330187,
12
          "hidden_flag": 0,
13
          "id": 3,
          "ignore_case_flag": 1,
15
          "max_age_seconds": 86400,
16
          "name": "Computer Name",
17
          "queries": {
18
    ..trimmed for brevity..
```

Export basetype json explode true

Export a BaseType from getting objects as JSON with true for explode_ison_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)
10
    pytan_root_dir = os.path.dirname(parent_dir)
11
    lib_dir = os.path.join(pytan_root_dir, 'lib')
12
    path_adds = [lib_dir]
13
14
    for aa in path_adds:
15
        if aa not in sys.path:
16
            sys.path.append(aa)
17
18
19
20
    # connection info for Tanium Server
    USERNAME = "Tanium User"
21
    PASSWORD = "T@n!um"
22
    HOST = "172.16.31.128"
23
    PORT = "443"
24
25
    # Logging conrols
26
    LOGLEVEL = 2
27
    DEBUGFORMAT = False
28
29
    import tempfile
30
31
    import pytan
32
    handler = pytan.Handler(
33
34
        username=USERNAME,
        password=PASSWORD,
35
        host=HOST,
36
        port=PORT,
37
        loglevel=LOGLEVEL,
38
        debugformat=DEBUGFORMAT,
39
40
41
    print handler
42
43
    # setup the export_obj kwargs for later
44
    export_kwargs = {}
45
    export_kwargs["export_format"] = u'json'
46
47
    export_kwargs["explode_json_string_values"] = True
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
        'objtype': 'sensor',
55
56
    response = handler.get(**get_kwargs)
57
58
    # export the object to a string
59
60
    # (we could just as easily export to a file using export_to_report_file)
    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():"
```

```
out = export_str

if len(out.splitlines()) > 15:

out = out.splitlines()[0:15]

out.append('..trimmed for brevity..')

out = '\n'.join(out)

print out
```

```
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
      "_type": "sensors",
5
      "sensor": [
6
7
          "_type": "sensor",
          "category": "Reserved",
9
          "description": "The assigned name of the client machine.\nExample: workstation-1 company.com",
10
          "exclude_from_parse_flag": 0,
11
          "hash": 3409330187,
12
          "hidden_flag": 0,
13
          "id": 3,
14
          "ignore_case_flag": 1,
15
          "max_age_seconds": 86400,
16
          "name": "Computer Name",
17
          "queries": {
18
    ..trimmed for brevity..
19
```

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)
8
    # determine the pytan lib dir and add it to the path
   parent_dir = os.path.dirname(my_dir)
10
   pytan_root_dir = os.path.dirname(parent_dir)
11
   lib_dir = os.path.join(pytan_root_dir, 'lib')
12
   path_adds = [lib_dir]
13
14
   for aa in path_adds:
15
        if aa not in sys.path:
16
            sys.path.append(aa)
17
```

```
18
19
    # connection info for Tanium Server
20
    USERNAME = "Tanium User"
21
    PASSWORD = "T@n!um"
22
    HOST = "172.16.31.128"
23
    PORT = "443"
24
25
    # Logging conrols
26
    LOGLEVEL = 2
27
    DEBUGFORMAT = False
28
29
30
    import tempfile
31
    import pytan
32
    handler = pytan.Handler(
33
        username=USERNAME,
34
        password=PASSWORD,
35
36
        host=HOST,
        port=PORT,
37
        loglevel=LOGLEVEL,
38
        debugformat=DEBUGFORMAT,
39
40
41
    print handler
42
43
    # setup the export_obj kwargs for later
44
    export_kwargs = {}
45
    export_kwargs["export_format"] = u'xml'
46
47
    # get the objects that will provide the basetype that we want to use
48
49
    get_kwargs = {
        'name': [
50
             "Computer Name", "IP Route Details", "IP Address",
51
             'Folder Name Search with RegEx Match',
52
53
        'objtype': 'sensor',
54
55
    response = handler.get(**get_kwargs)
56
57
    # export the object to a string
58
    # (we could just as easily export to a file using export_to_report_file)
59
    export_kwargs['obj'] = response
60
    export_str = handler.export_obj(**export_kwargs)
61
62
63
    print "print the export_str returned from export_obj():"
65
66
    out = export_str
67
68
    if len(out.splitlines()) > 15:
69
        out = out.splitlines()[0:15]
        out.append('..trimmed for brevity..')
70
        out = ' \ n'. join (out)
71
72
    print out
```

```
Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2
        print the export_str returned from export_obj():
3
        <sensors><cache_info /><sensor><category>Reserved</category>y>category>category>category>category>category>category>category>category>category>category>category>category>category>category>category>category>category>category>category>category>category>category>category>category>category>category>category>category>category>category>category>category>category>category>category>category>category>category>category>category>category>category>category>category>category>category>category>category>category>category>category>category>category>category>category>category>category>category>category>category>category>category>category>category>category>category>category>category>category>category>category>category>category>category>category>category>category>category>category>category>category>category>category>category>category>category>category>category>category>category>category>category>category>category>category>category>category>category>category>category>category>category>category>category>category>category>category>category>category>category>category>category>category>category>category>category>category>category>category>category>category>category>category>category>category>category>category>category>category>category>category>category>category>category>category>category>category>category>category>category>cate
        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 />ksubcolumns><s
        Set objWMIService = GetObject("winmgmts:" _
                  & " {impersonationLevel=impersonate}!\\" & strComputer &amp
        Set collip = objWMIService.ExecQuery("select * from win32_networkadapterconfiguration where
10
        dim ipaddrs()
11
        ipcount = 0
12
        for each ipItem in collip
13
14
                  for each ipaddr in ipItem.IPAddress
                           ipcount = ipcount + 1
15
                  next
16
17
        next.
        redim ipaddrs(ipcount)
18
        ..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
5
    my_file = os.path.abspath(sys.argv[0])
6
    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)
10
    pytan_root_dir = os.path.dirname(parent_dir)
11
    lib_dir = os.path.join(pytan_root_dir, 'lib')
12
    path_adds = [lib_dir]
13
14
    for aa in path_adds:
15
        if aa not in sys.path:
16
17
            sys.path.append(aa)
18
19
    # connection info for Tanium Server
20
    USERNAME = "Tanium User"
21
    PASSWORD = "T@n!um"
22
    HOST = "172.16.31.128"
23
    PORT = "443"
24
25
    # Logging conrols
26
   LOGLEVEL = 2
27
   DEBUGFORMAT = False
```

```
29
    import tempfile
30
31
    import pytan
32
    handler = pytan.Handler(
33
        username=USERNAME,
34
        password=PASSWORD,
35
        host=HOST,
36
        port=PORT,
37
        loglevel=LOGLEVEL,
38
        debugformat=DEBUGFORMAT,
39
40
41
    print handler
42
43
    # setup the export_obj kwargs for later
44
    export_kwargs = {}
45
    export_kwargs["export_format"] = u'xml'
46
    export_kwargs["minimal"] = False
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():"
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
```

```
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</pre>
```

```
Set collip = objWMIService.ExecQuery("select * from win32_networkadapterconfiguration where
10
   dim ipaddrs()
11
   ipcount = 0
12
   for each ipItem in collip
13
       for each ipaddr in ipItem.IPAddress
            ipcount = ipcount + 1
15
       next
16
   next
17
   redim ipaddrs(ipcount)
18
   ..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
2
    sys.dont_write_bytecode = True
    # Determine our script name, script dir
   my_file = os.path.abspath(sys.argv[0])
6
   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)
10
    pytan_root_dir = os.path.dirname(parent_dir)
11
    lib_dir = os.path.join(pytan_root_dir, 'lib')
12
   path_adds = [lib_dir]
13
14
    for aa in path_adds:
15
        if aa not in sys.path:
16
            sys.path.append(aa)
17
18
19
    # connection info for Tanium Server
20
    USERNAME = "Tanium User"
21
    PASSWORD = "T@n!um"
22
   HOST = "172.16.31.128"
23
   PORT = "443"
24
25
    # Logging conrols
26
    LOGLEVEL = 2
27
   DEBUGFORMAT = False
28
29
30
    import tempfile
31
    import pytan
32
   handler = pytan.Handler(
33
        username=USERNAME,
34
        password=PASSWORD,
35
36
        host=HOST,
        port=PORT,
```

```
loglevel=LOGLEVEL,
38
        debugformat=DEBUGFORMAT,
39
40
41
    print handler
42
43
    # setup the export_obj kwargs for later
44
    export_kwargs = {}
45
    export_kwarqs["export_format"] = u'xml'
46
    export_kwargs["minimal"] = True
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
    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
74
```

```
Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2
   print the export_str returned from export_obj():
   <sensors><sensor><category>Reserved</category><hash>3409330187</hash><name>Computer Name</name><hido
   Example: workstation-1.company.com</description><queries><query><platform>Windows</platform><script_
5
   Example: 172.16.0.0|192.168.1.1|255.255.0.0|UG|100|eth0</description><subcolumns><subcolumn><index>
6
   Set objWMIService = GetObject("winmgmts:" _
7
       & " {impersonationLevel=impersonate}!\\" & strComputer &amp
10
   Set collip = objWMIService.ExecQuery("select * from win32_networkadapterconfiguration where
11
   dim ipaddrs()
12
   ipcount = 0
   for each ipItem in collip
13
       for each ipaddr in ipItem.IPAddress
14
           ipcount = ipcount + 1
15
16
       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
2
    sys.dont_write_bytecode = True
    # Determine our script name, script dir
5
   my_file = os.path.abspath(sys.argv[0])
    my_dir = os.path.dirname(my_file)
7
8
    # determine the pytan lib dir and add it to the path
9
    parent_dir = os.path.dirname(my_dir)
10
    pytan_root_dir = os.path.dirname(parent_dir)
11
    lib_dir = os.path.join(pytan_root_dir, 'lib')
12
13
   path_adds = [lib_dir]
15
   for aa in path_adds:
        if aa not in sys.path:
16
            sys.path.append(aa)
17
18
19
    # connection info for Tanium Server
20
    USERNAME = "Tanium User"
21
    PASSWORD = "T@n!um"
22
    HOST = "172.16.31.128"
23
    PORT = "443"
24
25
    # Logging conrols
    LOGLEVEL = 2
27
28
   DEBUGFORMAT = False
29
   import tempfile
30
31
32
    import pytan
33
    handler = pytan.Handler(
34
        username=USERNAME,
        password=PASSWORD,
35
        host=HOST,
36
        port=PORT,
37
38
        loglevel=LOGLEVEL,
        debugformat=DEBUGFORMAT,
41
   print handler
42.
43
44
    # setup the export_obj kwargs for later
45
    export_kwargs = {}
   export_kwargs["export_format"] = u'csv'
```

```
export_kwarqs["explode_json_string_values"] = False
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
74
```

```
Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2
   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, N
   Example: 172.16.0.0|192.168.1.1|255.255.0.0|UG|100|eth0",1,435227963,0,552,1,Jim Olseh,60,0,defined
   Set objWMIService = GetObject(" winmgmts:" _
       & " {impersonationLevel=impersonate}!\\" & strComputer & "\root\cimv2&
10
11
   Set collip = objWMIService.ExecQuery(" select * from win32_networkadapterconfiguration where IPE
12
   dim ipaddrs()
13
   ipcount = 0
14
   for each ipItem in collip
15
       for each ipaddr in ipItem.IPAddress
16
           ipcount = ipcount + 1
17
       next
18
   ..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
2
    sys.dont_write_bytecode = True
3
    # Determine our script name, script dir
5
    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)
10
   pytan_root_dir = os.path.dirname(parent_dir)
11
    lib_dir = os.path.join(pytan_root_dir, 'lib')
12
    path_adds = [lib_dir]
13
14
    for aa in path_adds:
15
        if aa not in sys.path:
16
            sys.path.append(aa)
17
18
19
    # connection info for Tanium Server
20
    USERNAME = "Tanium User"
21
    PASSWORD = "T@n!um"
22
    HOST = "172.16.31.128"
23
    PORT = "443"
24
25
26
    # Logging conrols
27
    LOGLEVEL = 2
    DEBUGFORMAT = False
28
29
    import tempfile
30
31
    import pytan
32
    handler = pytan.Handler(
33
        username=USERNAME,
34
        password=PASSWORD,
35
        host=HOST,
36
        port=PORT,
37
        loglevel=LOGLEVEL,
38
39
        debugformat=DEBUGFORMAT,
40
41
    print handler
42
43
    # setup the export_obj kwargs for later
44
    export_kwargs = {}
45
    export_kwargs["export_format"] = u'csv'
46
    export_kwargs["explode_json_string_values"] = True
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)
```

```
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
64
    print ""
65
    print "print the export_str returned from export_obj():"
66
67
68
    out = export_str
    if len(out.splitlines()) > 15:
69
        out = out.splitlines()[0:15]
70
        out.append('..trimmed for brevity..')
71
        out = ' \ n'. join (out)
72
73
74
   print out
```

```
Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2
   print the export_str returned from export_obj():
3
   category, creation_time, delimiter, description, exclude_from_parse_flag, hash, hidden_flag, id, ignore_case
   Reserved, , , "The assigned name of the client machine.
   Example: workstation-1.company.com",0,3409330187,0,3,1,,86400,,,,,Computer Name,,,,,,
   Network, 2015-08-07T13:22:12, |, "Returns IPv4 network routes, filtered to exclude noise. | With Flags, N
   Example: 172.16.0.0|192.168.1.1|255.255.0.0|UG|100|eth0",1,435227963,0,552,1,Jim Olseh,60,0,defined
8
   Set objWMIService = GetObject(" winmgmts:" _
9
       & " {impersonationLevel=impersonate}!\\" & strComputer & "\root\cimv2&
10
11
   Set collip = objWMIService.ExecQuery(" select * from win32_networkadapterconfiguration where IPE
12
   dim ipaddrs()
13
   ipcount = 0
14
   for each ipItem in collip
15
       for each ipaddr in ipItem.IPAddress
16
           ipcount = ipcount + 1
17
       next
18
   ..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

```
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)
10
    pytan_root_dir = os.path.dirname(parent_dir)
11
    lib_dir = os.path.join(pytan_root_dir, 'lib')
12
    path_adds = [lib_dir]
13
14
    for aa in path_adds:
15
        if aa not in sys.path:
16
            sys.path.append(aa)
17
18
19
20
    # connection info for Tanium Server
    USERNAME = "Tanium User"
21
    PASSWORD = "T@n!um"
22
    HOST = "172.16.31.128"
23
    PORT = "443"
24
25
    # Logging conrols
26
    LOGLEVEL = 2
27
    DEBUGFORMAT = False
28
29
    import tempfile
30
31
    import pytan
32
    handler = pytan.Handler(
33
34
        username=USERNAME,
        password=PASSWORD,
35
        host=HOST,
36
        port=PORT,
37
        loglevel=LOGLEVEL,
38
        debugformat=DEBUGFORMAT,
39
40
41
    print handler
42
43
    # setup the export_obj kwargs for later
44
    export_kwargs = {}
45
    export_kwargs["export_format"] = u'csv'
46
    export_kwargs["header_sort"] = []
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
        '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():"
```

```
out = export_str

if len(out.splitlines()) > 15:

out = out.splitlines()[0:15]

out.append('..trimmed for brevity..')

out = '\n'.join(out)

print out
```

```
Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2
   print the export_str returned from export_obj():
3
   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, N
   Example: 172.16.0.0|192.168.1.1|255.255.0.0|UG|100|eth0",1,435227963,0,552,1,Jim Olseh,60,0,defined
   Set objWMIService = GetObject(" winmgmts:" _
       & " {impersonationLevel=impersonate}!\\" & strComputer & " \root\cimv2&
10
11
   Set collip = objWMIService.ExecQuery(" select * from win32_networkadapterconfiguration where IPE
12
   dim ipaddrs()
13
   ipcount = 0
14
   for each ipItem in collip
15
16
       for each ipaddr in ipItem.IPAddress
           ipcount = ipcount + 1
17
       next
18
   ..trimmed for brevity..
19
```

Export basetype csv with sort true

Export a BaseType from getting objects as CSV with true for header_sort

```
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
    # determine the pytan lib dir and add it to the path
   parent_dir = os.path.dirname(my_dir)
10
   pytan_root_dir = os.path.dirname(parent_dir)
11
   lib_dir = os.path.join(pytan_root_dir, 'lib')
12
   path_adds = [lib_dir]
13
14
   for aa in path_adds:
15
        if aa not in sys.path:
16
            sys.path.append(aa)
17
```

```
18
19
    # connection info for Tanium Server
20
    USERNAME = "Tanium User"
21
    PASSWORD = "T@n!um"
22
    HOST = "172.16.31.128"
23
    PORT = "443"
24
25
    # Logging conrols
26
    LOGLEVEL = 2
27
    DEBUGFORMAT = False
28
29
30
    import tempfile
31
    import pytan
32
    handler = pytan.Handler(
33
        username=USERNAME,
34
        password=PASSWORD,
35
        host=HOST,
36
        port=PORT,
37
        loglevel=LOGLEVEL,
38
        debugformat=DEBUGFORMAT,
39
40
41
    print handler
42
43
    # setup the export_obj kwargs for later
44
    export_kwargs = {}
45
    export_kwargs["export_format"] = u'csv'
46
    export_kwargs["header_sort"] = True
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
62
    export_str = handler.export_obj(**export_kwargs)
63
    print ""
65
    print "print the export_str returned from export_obj():"
66
67
68
    out = export_str
69
    if len(out.splitlines()) > 15:
        out = out.splitlines()[0:15]
70
        out.append('..trimmed for brevity..')
71
        out = ' \ n'.join(out)
72
73
    print out
74
```

```
Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2
   print the export_str returned from export_obj():
3
   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, N
   Example: 172.16.0.0|192.168.1.1|255.255.0.0|UG|100|eth0",1,435227963,0,552,1,Jim Olseh,60,0,defined
   Set objWMIService = GetObject(" winmgmts:" _
       & " {impersonationLevel=impersonate}!\\" & strComputer & "\root\cimv2&
10
11
   Set collip = objWMIService.ExecQuery(" select * from win32_networkadapterconfiguration where IPE
12
   dim ipaddrs()
13
14
   ipcount = 0
   for each ipItem in collip
15
       for each ipaddr in ipItem.IPAddress
16
           ipcount = ipcount + 1
17
18
       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

```
import os
    import sys
    sys.dont_write_bytecode = True
    # Determine our script name, script dir
    my_file = os.path.abspath(sys.argv[0])
6
    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)
10
    pytan_root_dir = os.path.dirname(parent_dir)
11
    lib_dir = os.path.join(pytan_root_dir, 'lib')
12
    path_adds = [lib_dir]
13
14
    for aa in path_adds:
15
16
        if aa not in sys.path:
            sys.path.append(aa)
19
    # connection info for Tanium Server
20
    USERNAME = "Tanium User"
21
    PASSWORD = "T@n!um"
22
    HOST = "172.16.31.128"
23
    PORT = "443"
24
25
    # Logging conrols
26
   LOGLEVEL = 2
27
   DEBUGFORMAT = False
```

```
29
    import tempfile
30
31
    import pytan
32
    handler = pytan.Handler(
33
        username=USERNAME,
34
        password=PASSWORD,
35
        host=HOST,
36
        port=PORT,
37
        loglevel=LOGLEVEL,
38
        debugformat=DEBUGFORMAT,
39
40
41
    print handler
42
43
    # setup the export_obj kwargs for later
44
    export_kwargs = {}
45
    export_kwargs["export_format"] = u'csv'
46
    export_kwargs["header_sort"] = [u'name', u'description']
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():"
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
```

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

print the export_str returned from export_obj():
name, description, category, creation_time, delimiter, exclude_from_parse_flag, hash, hidden_flag, id, ignores
Computer Name, "The assigned name of the client machine.
Example: workstation-1.company.com", Reserved,,,0,3409330187,0,3,1,,86400,,,,,,Windows, select CSName
IP Route Details, "Returns IPv4 network routes, filtered to exclude noise. With Flags, Metric, Interf
Example: 172.16.0.0|192.168.1.1|255.255.0.0|UG|100|eth0", Network, 2015-08-07T13:22:12, |,1,435227963,
```

```
Set objWMIService = GetObject(" winmqmts:" __
       & " {impersonationLevel=impersonate}!\\" & strComputer & "\root\cimv2&
10
11
   Set collip = objWMIService.ExecQuery(" select * from win32_networkadapterconfiguration where IPE
12
   dim ipaddrs()
13
   ipcount = 0
   for each ipItem in collip
15
       for each ipaddr in ipItem.IPAddress
16
           ipcount = ipcount + 1
17
       next.
18
   ..trimmed for brevity..
```

Export basetype json default options

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

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

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

```
Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
1
2
3
    print the export_str returned from export_obj():
4
5
      "_type": "sensors",
      "sensor": [
6
7
          "_type": "sensor",
8
          "category": "Reserved",
9
          "description": "The assigned name of the client machine.\nExample: workstation-1.company.com",
10
11
          "exclude_from_parse_flag": 0,
          "hash": 3409330187,
12
          "hidden_flag": 0,
13
          "id": 3,
14
          "ignore_case_flag": 1,
15
          "max_age_seconds": 86400,
16
          "name": "Computer Name",
17
          "queries": {
```

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

Export basetype json type true

Export a BaseType from getting objects as JSON with true for include_type

Example Python Code

```
import os
    import sys
    sys.dont_write_bytecode = True
    # Determine our script name, script dir
5
    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
9
    parent_dir = os.path.dirname(my_dir)
10
    pytan_root_dir = os.path.dirname(parent_dir)
11
    lib_dir = os.path.join(pytan_root_dir, 'lib')
12
   path_adds = [lib_dir]
13
14
    for aa in path_adds:
15
        if aa not in sys.path:
16
            sys.path.append(aa)
17
18
19
    # connection info for Tanium Server
20
    USERNAME = "Tanium User"
21
    PASSWORD = "T@n!um"
22
    HOST = "172.16.31.128"
23
    PORT = "443"
24
25
    # Logging conrols
26
27
    LOGLEVEL = 2
   DEBUGFORMAT = False
28
29
   import tempfile
30
31
32
    import pytan
33
    handler = pytan.Handler(
34
        username=USERNAME,
        password=PASSWORD,
35
        host=HOST,
36
        port=PORT,
37
        loglevel=LOGLEVEL,
38
39
        debugformat=DEBUGFORMAT,
40
41
   print handler
42
43
    # setup the export_obj kwargs for later
44
   export_kwargs = {}
45
   export_kwargs["export_format"] = u'json'
46
   export_kwargs["include_type"] = True
```

```
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_kwarqs['obj'] = response
61
    export_str = handler.export_obj(**export_kwargs)
62
63
64
   print ""
65
   print "print the export_str returned from export_obj():"
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
```

```
Handler for Session to 172.16.31.128:443, Authenticated: True, Version: Not yet determined!
2
   print the export_str returned from export_obj():
      "_type": "sensors",
5
      "sensor": [
6
          "_type": "sensor",
          "category": "Reserved",
          "description": "The assigned name of the client machine.\nExample: workstation-1 company.com",
10
          "exclude_from_parse_flag": 0,
11
          "hash": 3409330187,
12
          "hidden_flag": 0,
13
          "id": 3,
14
          "ignore_case_flag": 1,
15
          "max_age_seconds": 86400,
          "name": "Computer Name",
17
          "queries": {
18
    ..trimmed for brevity..
19
```

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
5
   my_file = os.path.abspath(sys.argv[0])
6
   my_dir = os.path.dirname(my_file)
9
    # determine the pytan lib dir and add it to the path
    parent_dir = os.path.dirname(my_dir)
10
    pytan_root_dir = os.path.dirname(parent_dir)
11
    lib_dir = os.path.join(pytan_root_dir, 'lib')
12
   path_adds = [lib_dir]
13
14
    for aa in path_adds:
15
        if aa not in sys.path:
16
            sys.path.append(aa)
17
18
19
    # connection info for Tanium Server
20
    USERNAME = "Tanium User"
21
    PASSWORD = "T@n!um"
22
    HOST = "172.16.31.128"
23
    PORT = "443"
24
25
    # Logging conrols
26
    LOGLEVEL = 2
27
   DEBUGFORMAT = False
28
29
   import tempfile
30
31
    import pytan
32
    handler = pytan.Handler(
33
        username=USERNAME,
34
35
        password=PASSWORD,
36
        host=HOST,
        port=PORT,
37
        loglevel=LOGLEVEL,
38
        debugformat=DEBUGFORMAT,
39
40
41
42
   print handler
43
    # setup the export_obj kwargs for later
44
    export_kwargs = {}
45
    export_kwargs["export_format"] = u'csv'
46
47
    export_kwargs["explode_json_string_values"] = u'bad'
    # get the objects that will provide the basetype that we want to use
```

```
get_kwargs = {
50
         'name': [
51
             "Computer Name", "IP Route Details", "IP Address",
52
             'Folder Name Search with RegEx Match',
53
54
         'objtype': 'sensor',
56
    response = handler.get(**get_kwargs)
57
    export_kwarqs['obj'] = response
58
59
    # export the object to a string
60
    # this should throw an exception: pytan.exceptions.HandlerError
61
    import traceback
62
63
    try:
64
        handler.export_obj(**export_kwargs)
65
    except Exception as e:
66
        traceback.print_exc(file=sys.stdout)
67
```

```
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 'unice
```

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
2
   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)
7
8
    # determine the pytan lib dir and add it to the path
9
   parent_dir = os.path.dirname(my_dir)
10
   pytan_root_dir = os.path.dirname(parent_dir)
11
   lib_dir = os.path.join(pytan_root_dir, 'lib')
12
   path_adds = [lib_dir]
13
14
   for aa in path_adds:
15
        if aa not in sys.path:
16
```

```
sys.path.append(aa)
17
18
19
    # connection info for Tanium Server
20
    USERNAME = "Tanium User"
21
    PASSWORD = "T@n!um"
22
    HOST = "172.16.31.128"
23
    PORT = "443"
24
25
    # Logging conrols
26
    LOGLEVEL = 2
27
    DEBUGFORMAT = False
28
29
    import tempfile
30
31
    import pytan
32
    handler = pytan.Handler(
33
        username=USERNAME,
34
        password=PASSWORD,
35
        host=HOST,
36
        port=PORT,
37
        loglevel=LOGLEVEL,
38
        debugformat=DEBUGFORMAT,
39
40
41
42
    print handler
43
    # setup the export_obj kwargs for later
44
    export_kwargs = {}
45
    export_kwargs["export_format"] = u'csv'
46
    export_kwargs["header_sort"] = [[]]
47
48
    # get the objects that will provide the basetype that we want to use
49
    get_kwargs = {
50
        'name': [
51
             "Computer Name", "IP Route Details", "IP Address",
52
             'Folder Name Search with RegEx Match',
53
54
        'objtype': 'sensor',
55
56
    response = handler.get(**get_kwargs)
57
    export_kwargs['obj'] = response
58
59
    # export the object to a string
60
    # this should throw an exception: pytan.exceptions.HandlerError
61
    import traceback
62
63
    try:
64
        handler.export_obj(**export_kwargs)
65
    except Exception as e:
66
        traceback.print_exc(file=sys.stdout)
67
```

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

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

```
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 supplements and the properties of t
```

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)
10
    pytan_root_dir = os.path.dirname(parent_dir)
11
    lib_dir = os.path.join(pytan_root_dir, 'lib')
12
    path_adds = [lib_dir]
13
14
    for aa in path_adds:
15
        if aa not in sys.path:
16
             sys.path.append(aa)
17
18
19
    # connection info for Tanium Server
20
    USERNAME = "Tanium User"
21
    PASSWORD = "T@n!um"
22
    HOST = "172.16.31.128"
23
    PORT = "443"
24
25
    # Logging conrols
26
    LOGLEVEL = 2
27
    DEBUGFORMAT = False
28
29
    import tempfile
30
31
    import pytan
32
    handler = pytan.Handler(
33
34
        username=USERNAME,
        password=PASSWORD,
35
        host=HOST,
36
        port=PORT,
37
        loglevel=LOGLEVEL,
38
        debugformat=DEBUGFORMAT,
39
40
41
    print handler
42
43
    # setup the export_obj kwargs for later
44
    export_kwargs = {}
45
    export_kwargs["export_format"] = u'xml'
46
    export_kwargs["minimal"] = u'bad'
47
48
    # get the objects that will provide the basetype that we want to use
49
    get_kwargs = {
50
         'name': [
51
             "Computer Name", "IP Route Details", "IP Address",
52
             'Folder Name Search with RegEx Match',
53
        'objtype': 'sensor',
55
56
    response = handler.get(**get_kwargs)
57
    export_kwargs['obj'] = response
58
59
60
    # export the object to a string
    # this should throw an exception: pytan.exceptions.HandlerError
61
    import traceback
62
63
    try:
64
        handler.export_obj(**export_kwargs)
65
   except Exception as e:
```

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

```
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
2
    sys.dont_write_bytecode = True
    # Determine our script name, script dir
   my file = os.path.abspath(sys.argv[0])
6
   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)
11
    lib dir = os.path.join(pytan root dir, 'lib')
12
   path_adds = [lib_dir]
13
14
   for aa in path_adds:
15
        if aa not in sys.path:
16
            sys.path.append(aa)
17
18
19
    # connection info for Tanium Server
20
    USERNAME = "Tanium User"
21
    PASSWORD = "T@n!um"
22
    HOST = "172.16.31.128"
23
    PORT = "443"
24
25
    # Logging conrols
26
    LOGLEVEL = 2
27
   DEBUGFORMAT = False
28
29
    import tempfile
30
31
   import pytan
32
   handler = pytan.Handler(
```

```
username=USERNAME,
34
        password=PASSWORD,
35
        host=HOST,
36
        port=PORT,
37
        loglevel=LOGLEVEL,
        debugformat=DEBUGFORMAT,
40
41
   print handler
42
43
    # setup the export_obj kwargs for later
44
    export_kwarqs = {}
45
    export_kwarqs["export_format"] = u'json'
46
    export_kwarqs["include_type"] = u'bad'
47
48
    # get the objects that will provide the basetype that we want to use
49
    get_kwargs = {
50
        'name': [
51
             "Computer Name", "IP Route Details", "IP Address",
52
             'Folder Name Search with RegEx Match',
53
        1,
54
        'objtype': 'sensor',
55
56
    response = handler.get(**get_kwargs)
57
    export_kwargs['obj'] = response
59
    # export the object to a string
60
    # this should throw an exception: pytan.exceptions.HandlerError
61
    import traceback
62
63
64
   try:
        handler.export_obj(**export_kwargs)
65
    except Exception as e:
66
        traceback.print_exc(file=sys.stdout)
```

```
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_ison_string_values

Example Python Code

```
import os
    import sys
2
    sys.dont_write_bytecode = True
    # Determine our script name, script dir
5
    my_file = os.path.abspath(sys.argv[0])
6
    my_dir = os.path.dirname(my_file)
7
    # determine the pytan lib dir and add it to the path
10
    parent_dir = os.path.dirname(my_dir)
   pytan_root_dir = os.path.dirname(parent_dir)
    lib_dir = os.path.join(pytan_root_dir, 'lib')
12
   path_adds = [lib_dir]
13
14
   for aa in path_adds:
15
        if aa not in sys.path:
16
            sys.path.append(aa)
17
18
19
    # connection info for Tanium Server
20
    USERNAME = "Tanium User"
21
    PASSWORD = "T@n!um"
22
    HOST = "172.16.31.128"
23
    PORT = "443"
24
25
    # Logging conrols
26
    LOGLEVEL = 2
27
   DEBUGFORMAT = False
28
29
    import tempfile
30
31
    import pytan
32
    handler = pytan.Handler(
33
        username=USERNAME,
34
        password=PASSWORD,
35
36
        host=HOST,
37
        port=PORT,
        loglevel=LOGLEVEL,
        debugformat=DEBUGFORMAT,
39
40
41
   print handler
42
43
    # setup the export_obj kwargs for later
    export_kwargs = {}
45
    export_kwarqs["export_format"] = u'json'
46
    export_kwargs["explode_json_string_values"] = u'bad'
47
48
49
    # get the objects that will provide the basetype that we want to use
50
    get_kwargs = {
51
        'name': [
            "Computer Name", "IP Route Details", "IP Address",
52
            'Folder Name Search with RegEx Match',
53
54
        'objtype': 'sensor',
55
56
57
    response = handler.get(**get_kwargs)
   export_kwarqs['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)
```

```
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 'unice
```

Invalid export basetype bad format

Export a BaseType from getting objects using a bad export_format

Example Python Code

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

```
# Logging conrols
26
    LOGLEVEL = 2
27
    DEBUGFORMAT = False
28
29
    import tempfile
30
31
    import pytan
32
    handler = pytan.Handler(
33
        username=USERNAME,
34
        password=PASSWORD,
35
        host=HOST,
36
        port=PORT,
37
        loglevel=LOGLEVEL,
38
        debugformat=DEBUGFORMAT,
39
40
41
    print handler
42
43
    # setup the export_obj kwargs for later
44
    export_kwargs = {}
45
    export_kwargs["export_format"] = u'bad'
46
47
    # get the objects that will provide the basetype that we want to use
48
    get_kwargs = {
49
        'name': [
50
             "Computer Name", "IP Route Details", "IP Address",
51
             'Folder Name Search with RegEx Match',
52
        ],
53
         'objtype': 'sensor',
54
55
    response = handler.get(**get_kwargs)
56
    export_kwargs['obj'] = response
57
58
    # export the object to a string
59
    # this should throw an exception: pytan.exceptions.HandlerError
60
    import traceback
61
62
    try:
63
64
        handler.export_obj(**export_kwargs)
    except Exception as e:
65
        traceback.print_exc(file=sys.stdout)
66
```

```
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 from SOAPElement (el)
```

```
static from_jsonable (jsonable)
```

Inverse of to_jsonable, with explode_json_string_values=False.

This can be used to import objects from serialized JSON. This JSON should come from Base-Type.to_jsonable(explode_json_string_values=False, include+type=True)

Examples

```
>>> with open('question_list.json') as fd:
...     questions = json.loads(fd.read())
...     # is a list of serialized questions
...     question_objects = BaseType.from_jsonable(questions)
...     # will return a list of api.Question
```

```
toSOAPBody (minimal=False)
```

```
toSOAPElement (minimal=False)
```

```
to_flat_dict (prefix='', explode_json_string_values=False)
```

Convert the object to a dict, flattening any lists or nested types

```
to_flat_dict_explode_json(val, prefix='')
```

see if the value is json. If so, flatten it out into a dict

```
static to_json (jsonable, **kwargs)
```

Convert to a json string.

jsonable can be a single BaseType instance of a list of BaseType

 $\verb"to_jsonable" (explode_json_string_values = False, include_type = True)$

```
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, ac-
tual)
```

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 from SOAPElement (el)
class taniumpy.object_types.column_set.ColumnSet
    Bases: object
    classmethod from SOAPElement (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'
{\bf class} \; {\tt 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.pluqin_sql_result.PluqinSqlResult
    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 from SOAPElement (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 from SOAPElement (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'
{\bf class} \; {\tt taniumpy.object\_types.sensor\_subcolumn\_list.} {\bf 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

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>
... <b>2</b>
```

1.4. xmltodict module 379

```
... """)
>>> 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:

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:

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 <code>@file_data('test_data.json')</code>, 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

1.5. ddt module 381

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

PYTHON MODULE INDEX

d	taniumpy.object_types.computer_group,
ddt, 381	373
р	taniumpy.object_types.computer_group_list, 373
pytan,3	taniumpy.object_types.computer_group_spec,
pytan.binsupport,72	373
pytan.constants, 58	<pre>taniumpy.object_types.computer_spec_list,</pre>
pytan.exceptions, 34	373
pytan.handler,3	taniumpy.object_types.error_list,373
pytan.pollers,52	taniumpy.object_types.filter,373
pytan.sessions, 35	taniumpy.object_types.filter_list,373
pytan.utils, 60	taniumpy.object_types.group,373
pytan.xml_clean, 81	taniumpy.object_types.group_list,373
	taniumpy.object_types.metadata_item,374
r	taniumpy.object_types.metadata_list,374
requests, 382	taniumpy.object_types.object_list,374
	taniumpy.object_types.object_list_types,
t	374
taniumpy, 371	taniumpy.object_types.options,374
taniumpy.object_types,371	taniumpy.object_types.package_file,374
taniumpy.object_types.action, 371	taniumpy.object_types.package_file_list,
taniumpy.object_types.action_list,371	374
taniumpy.object_types.action_list_info,	taniumpy.object_types.package_file_status, 374
taniumpy.object_types.action_stop, 371	<pre>taniumpy.object_types.package_file_status_list,</pre>
taniumpy.object_types.action_stop_list,	374
371 taniumpy.object_types.all_objects,371	taniumpy.object_types.package_file_template, 374
$\verb taniumpy.object_types.archived_question ,$	3/4
371	taniumpy.object_types.package_spec,374
taniumpy.object_types.archived_question_ 371	taniumpy.object_types.package_spec_list,
taniumpy.object_types.audit_data,371	taniumpy.object_types.parameter,374
taniumpy.object_types.base,371	taniumpy.object_types.parameter_list,
taniumpy.object_types.cache_filter,372	375
<pre>taniumpy.object_types.cache_filter_list,</pre>	taniumpy.object_types.parse_job,375
372	taniumpy.object_types.parse_job_list,
taniumpy.object_types.cache_info,373	375
taniumpy.object_types.client_count,373	taniumpy.object_types.parse_result,375
<pre>taniumpy.object_types.client_status,373 taniumpy.object_types.column,373</pre>	taniumpy.object_types.parse_result_group,
taniumpy.object_types.column,373 taniumpy.object_types.column_set,373	375
carringy.object_types.corumi_set, 3/3	

```
taniumpy.object_types.parse_result_grouptainstmpy.object_types.sensor_types,378
                                         taniumpy.object_types.soap_error,378
       375
taniumpy.object_types.parse_result_list, taniumpy.object_types.string_hint_list,
                                                378
taniumpy.object_types.permission_list,
                                         taniumpy.object_types.system_setting,
       375
taniumpy.object_types.plugin, 375
                                         taniumpy.object_types.system_setting_list,
taniumpy.object_types.plugin_argument,
       375
                                         taniumpy.object_types.system_status_aggregate,
taniumpy.object_types.plugin_argument_list,
                                                378
                                         taniumpy.object_types.system_status_list,
                                                378
taniumpy.object_types.plugin_command_list,
                                         taniumpy.object_types.upload_file,378
taniumpy.object_types.plugin_list, 375
                                         taniumpy.object_types.upload_file_list,
taniumpy.object_types.plugin_schedule,
                                                378
       375
                                         taniumpy.object_types.upload_file_status,
taniumpy.object_types.plugin_schedule_list,
                                                378
       376
                                          taniumpy.object_types.user, 378
taniumpy.object_types.plugin_sql, 376
                                         taniumpy.object_types.user_list, 378
taniumpy.object_types.plugin_sql_column, taniumpy.object_types.user_role, 379
                                         taniumpy.object_types.user_role_list,
taniumpy.object_types.plugin_sql_result,
                                                379
       376
                                         taniumpy.object_types.version_aggregate,
taniumpy.object_types.question, 376
taniumpy.object_types.question_list,376 taniumpy.object_types.version_aggregate_list,
taniumpy.object_types.question_list_info,
                                                379
       376
                                         taniumpy.object_types.white_listed_url,
taniumpy.object_types.result_info,376
taniumpy.object_types.result_set, 376
                                          taniumpy.object_types.white_listed_url_list,
taniumpy.object_types.row, 376
                                                379
taniumpy.object_types.saved_action, 377
                                         taniumpy.object_types.xml_error, 379
taniumpy.object_types.saved_action_approtest_pytan_invalid_server_tests,91
                                         test_pytan_unit,83
taniumpy.object_types.saved_action_list, test_pytan_valid_server_tests, 87
                                         threaded http, 381
       377
taniumpy.object_types.saved_action_policy,
taniumpy.object_types.saved_action_row_idmltsdict, 379
       377
taniumpy.object_types.saved_question,
taniumpy.object_types.saved_question_list,
taniumpy.object_types.select,377
taniumpy.object_types.select_list,377
taniumpy.object_types.sensor, 377
taniumpy.object_types.sensor_list,377
taniumpy.object_types.sensor_query,377
taniumpy.object_types.sensor_query_list,
       378
taniumpy.object_types.sensor_subcolumn,
taniumpy.object_types.sensor_subcolumn_list,
       378
```

386 Python Module Index

Symbols	method), 53
_author (in module pytan), 3	_derive_result_map() (pytan.pollers.ActionPoller method), 53
_copyright (in module pytan), 3	_derive_status() (pytan.pollers.ActionPoller method), 53
_license (in module pytan), 3	_derive_stopped_flag() (pytan.pollers.ActionPoller
_version (in module pytan), 3	method), 53
_add() (pytan.handler.Handler method), 6	_derive_target_group() (pytan.pollers.ActionPoller
_ask_manual() (pytan.handler.Handler method), 7	method), 53
_build_body() (pytan.sessions.Session method), 37	_derive_verify_enabled() (pytan.pollers.ActionPoller
_check_auth() (pytan.sessions.Session method), 37	method), 53
_check_sse_crash_prevention() (pytan.handler.Handler method), 9	_export_class_BaseType() (pytan.handler.Handler
_check_sse_empty_rs() (pytan.handler.Handler method),	method), 11
check_ssc_empty_1s() (pytan.nandici.frandici inctiou),	_export_class_ResultSet() (pytan.handler.Handler
_check_sse_format_support() (pytan.handler.Handler	method), 12
method), 9	_export_format_csv() (pytan.handler.Handler method),
check_sse_timing() (pytan.handler.Handler method), 9	12
_check_sse_version() (pytan.handler.Handler method), 9	_export_format_json() (pytan.handler.Handler method),
_clean_headers() (pytan.sessions.Session method), 37	12
_create_add_object_body() (pytan.sessions.Session	_export_format_xml() (pytan.handler.Handler method),
method), 38	12
_create_delete_object_body() (pytan.sessions.Session	_extract_resultxml() (pytan.sessions.Session method), 39
method), 38	_find() (pytan.handler.Handler method), 12
_create_get_object_body() (pytan.sessions.Session	_find_stat_target() (pytan.sessions.Session method), 39
method), 38	_fix_group() (pytan.pollers.ActionPoller method), 53
_create_get_result_data_body() (pytan.sessions.Session	_flatten_server_info() (pytan.sessions.Session method),
method), 38	40
_create_get_result_info_body() (pytan.sessions.Session	_from_json() (taniumpy.object_types.base.BaseType
method), 39	class method), 371
_create_run_plugin_object_body() (py-	_full_url() (pytan.sessions.Session method), 40
tan.sessions.Session method), 39	_get_multi() (pytan.handler.Handler method), 13
_create_update_object_body() (pytan.sessions.Session	_get_package_def() (pytan.handler.Handler method), 13
method), 39	_get_percentage() (pytan.sessions.Session method), 40
_deploy_action() (pytan.handler.Handler method), 9	_get_response() (pytan.sessions.Session method), 40
_derive_attribute() (pytan.pollers.QuestionPoller	_get_sensor_defs() (pytan.handler.Handler method), 13
method), 55	_get_single() (pytan.handler.Handler method), 13
_derive_expiration() (pytan.pollers.QuestionPoller	_http_get() (pytan.sessions.Session method), 41
method), 55	_http_post() (pytan.sessions.Session method), 42
_derive_object_info() (pytan.pollers.ActionPoller	_parse_versioning() (pytan.handler.Handler method), 13
method), 53	_platform_is_6_2() (pytan.handler.Handler method), 13
_derive_object_info() (pytan.pollers.QuestionPoller	_post_init() (pytan.pollers.ActionPoller method), 53
method), 55	_post_init() (pytan.pollers.QuestionPoller method), 56
_derive_package_spec() (pytan.pollers.ActionPoller	_post_init() (pytan.pollers.SSEPoller method), 57

```
_refetch_obj() (pytan.pollers.QuestionPoller method), 56
                                                                                                    attribute), 374
_regex_body_for_element()
                                                (pytan.sessions.Session
                                                                                     soap tag (taniumpy.object types.metadata item.MetadataItem
              method), 44
                                                                                                    attribute), 374
_replace_auth() (pytan.sessions.Session method), 44
                                                                                      _soap_tag (taniumpy.object_types.metadata_list.MetadataList
resolve sse format() (pytan.handler.Handler method),
                                                                                                    attribute), 374
                                                                                      soap tag (taniumpy.object types.object list.ObjectList
resolve stat target() (pytan.sessions.Session method),
                                                                                                    attribute), 374
                                                                                                               (taniumpy.object_types.options.Options
                                                                                      _soap_tag
_single_find() (pytan.handler.Handler method), 14
                                                                                                    attribute), 374
                                                                                     _soap_tag (taniumpy.object_types.package_file.PackageFile
                  (taniumpy.object_types.action.Action
_soap_tag
                                                                              at-
              tribute), 371
                                                                                                    attribute), 374
_soap_tag (taniumpy.object_types.action_list.ActionList
                                                                                     _soap_tag (taniumpy.object_types.package_file_list.PackageFileList
              attribute), 371
                                                                                                    attribute), 374
_soap_tag (taniumpy.object_types.action_list_info.ActionListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonListlonLi
              attribute), 371
                                                                                                    attribute), 374
_soap_tag (taniumpy.object_types.action_stop.ActionStop _soap_tag (taniumpy.object_types.package_file_status_list.PackageFileStat
              attribute), 371
                                                                                                    attribute), 374
_soap_tag (taniumpy.object_types.action_stop_list.ActionStop&apt_tag (taniumpy.object_types.package file template.PackageFileTemp
              attribute), 371
                                                                                                    attribute), 374
_soap_tag (taniumpy.object_types.archived_question.Archivestapetaipotaniumpy.object_types.package_file_template_list.PackageFileT
              attribute), 371
                                                                                                    attribute), 374
_soap_tag (taniumpy.object_types.archived_question_list.ArcstoxpdQpg(stionInisty.object_types.package_spec.PackageSpec
              attribute), 371
                                                                                                    attribute), 374
_soap_tag (taniumpy.object_types.audit_data.AuditData _soap_tag (taniumpy.object_types.package_spec_list.PackageSpecList
                                                                                                    attribute), 374
              attribute), 371
_soap_tag (taniumpy.object_types.base.BaseType at-
                                                                                     _soap_tag (taniumpy.object_types.parameter.Parameter
              tribute), 371
                                                                                                    attribute), 374
_soap_tag (taniumpy.object_types.cache_filter.CacheFilter _soap_tag (taniumpy.object_types.parameter_list.ParameterList
                                                                                                    attribute), 375
              attribute), 372
_soap_tag (taniumpy.object_types.cache_filter_list.CacheFilterlaist_tag (taniumpy.object_types.parse_job.ParseJob at-
                                                                                                    tribute), 375
              attribute), 372
_soap_tag (taniumpy.object_types.cache_info.CacheInfo _soap_tag (taniumpy.object_types.parse_job_list.ParseJobList
              attribute), 373
                                                                                                    attribute), 375
_soap_tag (taniumpy.object_types.client_count.ClientCount_soap_tag (taniumpy.object_types.parse_result.ParseResult
              attribute), 373
                                                                                                    attribute), 375
_soap_tag (taniumpy.object_types.client_status.ClientStatus_soap_tag (taniumpy.object_types.parse_result_group.ParseResultGroup
              attribute), 373
                                                                                                    attribute), 375
_soap_tag (taniumpy.object_types.computer_group.Computers6aputag (taniumpy.object_types.parse_result_group_list.ParseResultGrou
              attribute), 373
                                                                                                    attribute), 375
_soap_tag (taniumpy.object_types.computer_group_list.ComputerGagu(pdristmpy.object_types.parse_result_list.ParseResultList
                                                                                                    attribute), 375
              attribute), 373
_soap_tag (taniumpy.object_types.computer_group_spec.Computert@gotupSpecpy.object_types.permission_list.PermissionList
              attribute), 373
                                                                                                    attribute), 375
_soap_tag (taniumpy.object_types.computer_spec_list.ComputerspetagList(taniumpy.object_types.plugin.Plugin
              attribute), 373
                                                                                                    tribute), 375
_soap_tag (taniumpy.object_types.error_list.ErrorList at- _soap_tag (taniumpy.object_types.plugin_argument.PluginArgument
              tribute), 373
                                                                                                    attribute), 375
_soap_tag (taniumpy.object_types.filter.Filter attribute),
                                                                                     _soap_tag(taniumpy.object_types.plugin_argument_list.PluginArgumentLi
                                                                                                    attribute), 375
_soap_tag (taniumpy.object_types.filter_list.FilterList at-
                                                                                     _soap_tag(taniumpy.object_types.plugin_command_list.PluginCommandL
                                                                                                    attribute), 375
              tribute), 373
                                                                                      _soap_tag (taniumpy.object_types.plugin_list.PluginList
                   (taniumpy.object_types.group.Group
_soap_tag
                                                                              at-
                                                                                                    attribute), 375
              tribute), 373
_soap_tag (taniumpy.object_types.group_list.GroupList _soap_tag (taniumpy.object_types.plugin_schedule.PluginSchedule
```

```
attribute), 376
                                                                      attribute), 378
soap tag (taniumpy.object types.plugin schedule list.Plugin Saphetdugle laisitumpy.object types.system status list.System Status List
         attribute), 376
                                                                      attribute), 378
_soap_tag (taniumpy.object_types.plugin_sql.PluginSql _soap_tag (taniumpy.object_types.upload_file.UploadFile
          attribute), 376
                                                                      attribute), 378
soap tag (taniumpy.object types.plugin sql column.Plugin Sqd Cotang (taniumpy.object types.upload file list.UploadFileList
          attribute), 376
                                                                      attribute), 378
soap tag (taniumpy.object types.plugin sql result.PluginSql Papsultag (taniumpy.object types.upload file status.UploadFileStatus
          attribute), 376
                                                                      attribute), 378
_soap_tag (taniumpy.object_types.question.Question at- _soap_tag (taniumpy.object_types.user.User attribute),
          tribute), 376
_soap_tag (taniumpy.object_types.question_list.QuestionList_soap_tag (taniumpy.object_types.user_list.UserList at-
          attribute), 376
                                                                      tribute), 379
_soap_tag (taniumpy.object_types.question_list_info.QuestionbastIttfg (taniumpy.object_types.user_role.UserRole at-
          attribute), 376
                                                                      tribute), 379
_soap_tag (taniumpy.object_types.saved_action.SavedActionsoap_tag (taniumpy.object_types.user_role_list.UserRoleList
          attribute), 377
                                                                      attribute), 379
soap tag (taniumpy.object types.saved action approval.SavedActar(Ampinoval.object types.version aggregate.VersionAggregate
          attribute), 377
                                                                      attribute), 379
soap tag (taniumpy.object types.saved action list.SavedActionpListg (taniumpy.object types.version aggregate list.VersionAggregate
          attribute), 377
                                                                      attribute), 379
soap tag (taniumpy.object types.saved action policy.Saveds\\approxes\text{apion}\) action policy. Saveds\\alpha\text{apion}\) action \(\text{Policy}\) (taniumpy.object types.white listed url. White Listed Url.)
          attribute), 377
                                                                      attribute), 379
soap tag (taniumpy.object types.saved action row id list. Soved Augi (taniumpy.object types.white listed url list. White Listed Url List
                                                                      attribute), 379
          attribute), 377
_soap_tag (taniumpy.object_types.saved_question.SavedQuestoap_tag
                                                                         (taniumpy.object types.xml error.XmlError
          attribute), 377
                                                                      attribute), 379
_soap_tag (taniumpy.object_types.saved_question_list.Saved@nestistats_isthread() (pytan.sessions.Session method), 45
                                                            _stats_loop() (pytan.sessions.Session method), 45
         attribute), 377
_soap_tag (taniumpy.object_types.select.Select attribute),
                                                            stop (pytan.pollers.QuestionPoller attribute), 56
                                                            _version_support_check()
                                                                                               (pytan.handler.Handler
_soap_tag (taniumpy.object_types.select_list.SelectList
                                                                      method), 14
         attribute), 377
             (taniumpy.object_types.sensor.Sensor
                                                       at-
_soap_tag
         tribute), 377
                                                            Action (class in taniumpy.object types.action), 371
soap tag (taniumpy.object types.sensor list.SensorList
                                                            ACTION DONE KEY (pytan.pollers.ActionPoller at-
         attribute), 377
                                                                      tribute), 53
_soap_tag (taniumpy.object_types.sensor_query.SensorQuerxctionList (class in taniumpy.object_types.action_list),
          attribute), 377
                                                                      371
_soap_tag (taniumpy.object_types.sensor_query_list.SensorQuery_liststInfo
                                                                                     (class
                                                                                                     in
                                                                                                                 tani-
          attribute), 378
                                                                      umpy.object_types.action_list_info), 371
_soap_tag (taniumpy.object_types.sensor_subcolumn.Sensor_subsaturation (class in pytan.pollers), 52
          attribute), 378
                                                            ActionStop (class in taniumpy.object_types.action_stop),
_soap_tag (taniumpy.object_types.sensor_subcolumn_list.SensorSubcohumnList
          attribute), 378
                                                            ActionStopList
                                                                                     (class
                                                                                                                 tani-
_soap_tag (taniumpy.object_types.soap_error.SoapError
                                                                      umpy.object types.action stop list), 371
          attribute), 378
                                                            add() (pytan.sessions.Session method), 45
_soap_tag (taniumpy.object_types.string_hint_list.StringHintHistask_report_argparser()
                                                                                             (in
                                                                                                      module
                                                                                                                   ру-
          attribute), 378
                                                                      tan.binsupport), 73
_soap_tag (taniumpy.object_types.system_setting.SystemSetting_file_log() (in module pytan.binsupport), 73
          attribute), 378
                                                            add get object report argparser() (in module
_soap_tag (taniumpy.object_types.system_setting_list.SystemSettingList.binsupport), 73
          attribute), 378
                                                            add report file_options() (in module pytan.binsupport),
soap tag (taniumpy.object types.system status aggregate.SystemStatusAggregate
```

ALL_REQUESTS_RESPONSES (pytan.sessions.Session attribute), 36	<pre>check_for_help() (in module pytan.utils), 62 chew_csv() (in module test_pytan_valid_server_tests), 91</pre>
append() (taniumpy.object_types.base.BaseType method), 371	chk_def_key() (in module pytan.utils), 63 clean_kwargs() (in module pytan.utils), 63
apply_options_obj() (in module pytan.utils), 60	ClientCount (class in tani-
approve_saved_action() (pytan.handler.Handler method),	umpy.object_types.client_count), 373
14	ClientStatus (class in tani-
ArchivedQuestion (class in tani-	umpy.object_types.client_status), 373
umpy.object_types.archived_question), 371	Column (class in taniumpy.object_types.column), 373
ArchivedQuestionList (class in tani-	ColumnSet (class in taniumpy.object_types.column_set),
umpy.object_types.archived_question_list),	373
371	COMPLETE_PCT_DEFAULT (py-
ask() (pytan.handler.Handler method), 14	tan.pollers.ActionPoller attribute), 53
ask_manual() (pytan.handler.Handler method), 15	COMPLETE_PCT_DEFAULT (py-
ask_parsed() (pytan.handler.Handler method), 18	tan.pollers.QuestionPoller attribute), 55
ask_saved() (pytan.handler.Handler method), 19	ComputerGroup (class in tani-
AuditData (class in taniumpy.object_types.audit_data), 371	umpy.object_types.computer_group), 373 ComputerGroupList (class in tani-
AUTH_CONNECT_TIMEOUT_SEC (py-	umpy.object_types.computer_group_list),
tan.sessions.Session attribute), 36 AUTH_FAIL_CODES (pytan.sessions.Session attribute),	373 ComputerGroupSpec (class in tani-
36	umpy.object_types.computer_group_spec),
AUTH_RES (pytan.sessions.Session attribute), 36	373
AUTH_RESPONSE_TIMEOUT_SEC (py-	ComputerSpecList (class in tani-
tan.sessions.Session attribute), 36	umpy.object_types.computer_spec_list),
authenticate() (pytan.sessions.Session method), 45	373
authlog (pytan.sessions.Session attribute), 46	copy_obj() (in module pytan.utils), 63
AuthorizationError, 34	copy_package_obj_for_action() (in module pytan.utils), 63
В	create_dashboard() (pytan.handler.Handler method), 20
BAD_RESPONSE_CMD_PRUNES (py-	create_from_json() (pytan.handler.Handler method), 21
tan.sessions.Session attribute), 36	create_group() (pytan.handler.Handler method), 21
BAD_SERVER_VERSIONS (pytan.sessions.Session at-	create_package() (pytan.handler.Handler method), 22
tribute), 36	create_report_file() (pytan.handler.Handler method), 23
BadResponseError, 34	create_sensor() (pytan.handler.Handler method), 24
BaseType (class in taniumpy.object_types.base), 371	create_user() (pytan.handler.Handler method), 24 create_whitelisted_url() (pytan.handler.Handler method),
bodyhttplog (pytan.sessions.Session attribute), 46 build_group_obj() (in module pytan.utils), 60	24
build_manual_q() (in module pytan.utils), 60	csvdictwriter() (in module pytan.binsupport), 73
build_metadatalist_obj() (in module pytan.utils), 61	CustomArgFormat (class in pytan.binsupport), 72
build_param_obj() (in module pytan.utils), 61	CustomArgParse (class in pytan.binsupport), 73 CustomHTTPHandler (class in threaded_http), 381
build_param_objlist() (in module pytan.utils), 61	Customiri i r Handiei (ciass in uneaded_intp), 381
build_selectlist_obj() (in module pytan.utils), 62	D
C	data() (in module ddt), 381
	datetime_to_timestr() (in module pytan.utils), 64
CacheFilter (class in taniumpy.object_types.cache_filter), 372	ddt (module), 381
CacheFilterList (class in tani-	ddt() (in module ddt), 381
umpy.object_types.cache_filter_list), 372	DEBUG_FORMAT (in module pytan.constants), 58
CacheInfo (class in taniumpy.object_types.cache_info),	debug_list() (in module pytan.binsupport), 73 debug_obj() (in module pytan.binsupport), 73
373 calc_percent() (in module pytan.utils), 62	DEFAULT_REPLACEMENT (in module py-
change_console_format() (in module pytan.utils), 62	tan.xml_clean), 81
check dictkey() (in module pytan.utils), 62	DefinitionParserError, 34

dehumanize_package() (in module pytan.utils), 64	find() (pytan.sessions.Session method), 47
dehumanize_question_filters() (in module pytan.utils), 64	$finished_eq_passed_loop() \qquad (pytan.pollers.ActionPoller$
dehumanize_question_options() (in module pytan.utils),	method), 53
64	flatten_jsonable() (taniumpy.object_types.base.BaseType
dehumanize_sensors() (in module pytan.utils), 64	method), 371
delete() (pytan.handler.Handler method), 25	from_jsonable() (taniumpy.object_types.base.BaseType
delete() (pytan.sessions.Session method), 46	static method), 372
delete_dashboard() (pytan.handler.Handler method), 25	fromSOAPBody() (tani-
deploy_action() (pytan.handler.Handler method), 25	umpy.object_types.base.BaseType class
derive_param_default() (in module pytan.utils), 65	method), 371
disable_stats_loop() (pytan.sessions.Session method), 46	fromSOAPElement() (tani-
do_GET() (threaded_http.CustomHTTPHandler	umpy.object_types.base.BaseType class
method), 382	method), 372
do_POST() (threaded_http.CustomHTTPHandler	fromSOAPElement() (tani- umpy.object_types.column.Column class
method), 382	umpy.object_types.column.Column class method), 373
E	fromSOAPElement() (tani-
	umpy.object_types.column_set.ColumnSet
ELEMENT_RE_TXT (pytan.sessions.Session attribute),	class method), 373
36	fromSOAPElement() (tani-
emit() (pytan.utils.SplitStreamHandler method), 60	umpy.object_types.result_info.ResultInfo
empty_obj() (in module pytan.utils), 65 ENABLE_LOGGING (threaded_http.CustomHTTPHandle	
attribute), 382	fromSOAPElement() (tani-
enable_stats_loop() (pytan.sessions.Session method), 47	umpy.object_types.result_set.ResultSet class
error() (pytan.binsupport.CustomArgParse method), 73	method), 376
ErrorList (class in taniumpy.object_types.error_list), 373	fromSOAPElement() (taniumpy.object_types.row.Row
eval_timing() (in module pytan.utils), 65	class method), 377
EXPIRATION_ATTR (pytan.pollers.ActionPoller at-	func_timing() (in module pytan.utils), 66
tribute), 53	G
EXPIRATION_ATTR (pytan.pollers.QuestionPoller at-	
tribute), 55	get() (pytan.handler.Handler method), 30
EXPIRY_FALLBACK_SECS (py-	get_all() (pytan.handler.Handler method), 31
tan.pollers.QuestionPoller attribute), 55 explode_json() (taniumpy.object_types.base.BaseType	get_all_headers() (in module pytan.binsupport), 74 get_all_loggers() (in module pytan.utils), 66
method), 371	get_dashboards() (pytan.handler.Handler method), 31
export_id (pytan.pollers.SSEPoller attribute), 57	get_dict_list_len() (in module pytan.utils), 66
EXPORT_MAPS (in module pytan.constants), 58	get_filter_obj() (in module pytan.utils), 66
export_obj() (pytan.handler.Handler method), 27	get_grp_opts() (in module pytan.binsupport), 74
export_to_report_file() (pytan.handler.Handler method),	get_kwargs_int() (in module pytan.utils), 66
29	get_now() (in module pytan.utils), 67
extract_filter() (in module pytan.utils), 65	GET_OBJ_MAP (in module pytan.constants), 58
extract_options() (in module pytan.utils), 65	get_obj_map() (in module pytan.utils), 67
extract_params() (in module pytan.utils), 65	get_obj_params() (in module pytan.utils), 67
extract_selector() (in module pytan.utils), 66	
	get percentage() (in module pytan.utils), 67
	get_percentage() (in module pytan.utils), 67 get_q_obj_map() (in module pytan.utils), 67
F	get_q_obj_map() (in module pytan.utils), 67
file_data() (in module ddt), 381	get_q_obj_map() (in module pytan.utils), 67 get_result_data() (pytan.handler.Handler method), 31
	get_q_obj_map() (in module pytan.utils), 67 get_result_data() (pytan.handler.Handler method), 31 get_result_data() (pytan.pollers.QuestionPoller method),
file_data() (in module ddt), 381 Filter (class in taniumpy.object_types.filter), 373 filter_filename() (in module pytan.binsupport), 73	get_q_obj_map() (in module pytan.utils), 67 get_result_data() (pytan.handler.Handler method), 31 get_result_data() (pytan.pollers.QuestionPoller method), 56
file_data() (in module ddt), 381 Filter (class in taniumpy.object_types.filter), 373	get_q_obj_map() (in module pytan.utils), 67 get_result_data() (pytan.handler.Handler method), 31 get_result_data() (pytan.pollers.QuestionPoller method), 56 get_result_data() (pytan.sessions.Session method), 47
file_data() (in module ddt), 381 Filter (class in taniumpy.object_types.filter), 373 filter_filename() (in module pytan.binsupport), 73 FILTER_MAPS (in module pytan.constants), 58	get_q_obj_map() (in module pytan.utils), 67 get_result_data() (pytan.handler.Handler method), 31 get_result_data() (pytan.pollers.QuestionPoller method), 56 get_result_data() (pytan.sessions.Session method), 47 get_result_data_sse() (pytan.handler.Handler method), 32 get_result_data_sse() (pytan.sessions.Session method), 47
file_data() (in module ddt), 381 Filter (class in taniumpy.object_types.filter), 373 filter_filename() (in module pytan.binsupport), 73 FILTER_MAPS (in module pytan.constants), 58 FILTER_RE (in module pytan.constants), 58	get_q_obj_map() (in module pytan.utils), 67 get_result_data() (pytan.handler.Handler method), 31 get_result_data() (pytan.pollers.QuestionPoller method), 56 get_result_data() (pytan.sessions.Session method), 47 get_result_data_sse() (pytan.handler.Handler method), 32 get_result_data_sse() (pytan.sessions.Session method),

get_result_info() (pytan.pollers.QuestionPoller method), 56	is_num() (in module pytan.utils), 68 is_str() (in module pytan.utils), 68
56 get_result_info() (pytan.sessions.Session method), 48 get_server_info() (pytan.sessions.Session method), 48 get_server_stats() (pytan.sessions.Session method), 48 get_server_version() (pytan.handler.Handler method), 33 get_server_version() (pytan.sessions.Session method), 48 get_server_version() (pytan.sessions.Session method), 48 get_ses_data() (pytan.pollers.SSEPoller method), 57 get_sse_status() (pytan.pollers.SSEPoller method), 58 get_taniumpy_obj() (in module pytan.utils), 67 Group (class in taniumpy.object_types.group), 373 GroupList (class in taniumpy.object_types.group_list),	is_str() (in module pytan.utils), 68 J jsonify() (in module pytan.utils), 68 L LAST_REQUESTS_RESPONSE (pytan.sessions.Session attribute), 36 LAST_RESPONSE_INFO (pytan.sessions.Session attribute), 36 load_param_json_file() (in module pytan.utils), 68 load_taniumpy_from_json() (in module pytan.utils), 68 LOG_LEVEL_MAPS (in module pytan.constants), 59 log_message() (threaded_http.CustomHTTPHandler method), 382 log_session_communication() (in module pytan.utils), 69 logout() (pytan.sessions.Session method), 51
HistoryConsole (class in pytan.binsupport), 73 host (pytan.sessions.Session attribute), 49 HTTP_AUTH_RETRY (pytan.sessions.Session attribute), 36 HTTP_DEBUG (pytan.sessions.Session attribute), 36 http_get() (pytan.sessions.Session method), 49 http_post() (pytan.sessions.Session method), 50 HTTP_RETRY_COUNT (pytan.sessions.Session attribute), 36 HttpError, 34 httplog (pytan.sessions.Session attribute), 51 human_time() (in module pytan.utils), 68 HumanParserError, 34	map_filter() (in module pytan.utils), 69 map_option() (in module pytan.utils), 69 map_options() (in module pytan.utils), 69 MetadataItem (class in taniumpy.object_types.metadata_item), 374 MetadataList (class in taniumpy.object_types.metadata_list), 374 mk_test_name() (in module ddt), 381 mylog (pytan.pollers.QuestionPoller attribute), 56 mylog (pytan.sessions.Session attribute), 51
	N
IncorrectTypeException, 372 NFO_CONNECT_TIMEOUT_SEC (pytan.sessions.Session attribute), 36	NotFoundError, 34
NFO_FORMAT (in module pytan.constants), 59 NFO_RES (pytan.sessions.Session attribute), 36 NFO_RESPONSE_TIMEOUT_SEC (pytan.sessions.Session attribute), 36 nit_history() (pytan.binsupport.HistoryConsole method), 73 nput_prompts() (in module pytan.binsupport), 74 ntrospect() (in module pytan.binsupport), 74	obj (pytan.pollers.QuestionPoller attribute), 56 OBJECT_TYPE (pytan.pollers.ActionPoller attribute), 53 OBJECT_TYPE (pytan.pollers.QuestionPoller attribute), 55 ObjectList (class in taniumpy.object_types.object_list), 374 OPTION_MAPS (in module pytan.constants), 59
NVALID_UNICODE_RAW_RE (in module py-tan.xml_clean), 81 NVALID_UNICODE_RE (in module pytan.xml_clean), 81	OPTION_RE (in module pytan.constants), 59 Options (class in taniumpy.object_types.options), 374 OVERRIDE_TIMEOUT_SECS_DEFAULT (pytan.pollers.QuestionPoller attribute), 55
InvalidServerTests (class in test_pytan_invalid_server_tests), 91	P
s_auth (pytan.sessions.Session attribute), 51 s_dict() (in module pytan.utils), 68 s_hash_randomized() (in module ddt), 381	PackageFile (class in tani- umpy.object_types.package_file), 374 PackageFileList (class in tani-
s list() (in module pytan utils) 68	umpy.object_types.package_file_list), 374

PackageFileStatus (class in tani- umpy.object_types.package_file_status), 374	PluginCommandList (class in tani- umpy.object_types.plugin_command_list), 375
PackageFileStatusList (class in tani-	PluginList (class in taniumpy.object_types.plugin_list),
umpy.object_types.package_file_status_list),	375
374	PluginSchedule (class in tani-
PackageFileTemplate (class in tani-	umpy.object_types.plugin_schedule), 375
umpy.object_types.package_file_template),	PluginScheduleList (class in tani-
374	umpy.object_types.plugin_schedule_list),
PackageFileTemplateList (class in tani-	376
umpy.object_types.package_file_template_list),	PluginSql (class in taniumpy.object_types.plugin_sql),
374	376
PackageSpec (class in tani-	PluginSqlColumn (class in tani-
umpy.object_types.package_spec), 374	umpy.object_types.plugin_sql_column),
PackageSpecList (class in tani-	376
umpy.object_types.package_spec_list), 374	PluginSqlResult (class in tani-
PARAM_DELIM (in module pytan.constants), 59	umpy.object_types.plugin_sql_result), 376
PARAM_KEY_SPLIT (in module pytan.constants), 59	POLLING_SECS_DEFAULT (py-
PARAM_RE (in module pytan.constants), 59	tan.pollers.QuestionPoller attribute), 55
PARAM_SPLIT_RE (in module pytan.constants), 59	POLLING_SECS_DEFAULT (pytan.pollers.SSEPoller
Parameter (class in taniumpy.object_types.parameter),	attribute), 57
374	PollingError, 34
ParameterList (class in tani-	port (pytan.sessions.Session attribute), 51
umpy.object_types.parameter_list), 375	port_check() (in module pytan.utils), 70
parse() (in module xmltodict), 379	print_help() (pytan.binsupport.CustomArgParse method),
parse_defs() (in module pytan.utils), 69	73
parse_query() (pytan.handler.Handler method), 33	print_log_levels() (in module pytan.utils), 70
parse_sensor_platforms() (in module pytan.binsupport),	print_obj() (in module pytan.binsupport), 74
74	process_ask_manual_args() (in module py-
ParseJob (class in taniumpy.object_types.parse_job), 375	tan.binsupport), 74
ParseJobList (class in tani-	process_ask_saved_args() (in module pytan.binsupport),
umpy.object_types.parse_job_list), 375	74
ParseResult (class in tani-	process_create_group_args() (in module py-
umpy.object_types.parse_result), 375	tan.binsupport), 75
ParseResultGroup (class in tani-	process_create_json_object_args() (in module py-
umpy.object_types.parse_result_group),	tan.binsupport), 75
375	process_create_package_args() (in module py-
ParseResultGroupList (class in tani-	tan.binsupport), 75
umpy.object_types.parse_result_group_list),	process_create_sensor_args() (in module py-
375	tan.binsupport), 75
ParseResultList (class in tani-	process_create_user_args() (in module pytan.binsupport),
umpy.object_types.parse_result_list), 375	76
passed_eq_est_total_loop() (pytan.pollers.QuestionPoller	process_create_whitelisted_url_args() (in module py-
method), 56	tan.binsupport), 76
PermissionList (class in tani-	process_delete_object_args() (in module py-
umpy.object_types.permission_list), 375	tan.binsupport), 76
PickerError, 34	process_deploy_action_args() (in module py-
Plugin (class in taniumpy.object_types.plugin), 375	tan.binsupport), 77
plugin_zip() (in module pytan.utils), 70	process_get_object_args() (in module pytan.binsupport),
PluginArgument (class in tani-	77
umpy.object_types.plugin_argument), 375	process_get_results_args() (in module pytan.binsupport),
PluginArgumentList (class in tani-	77
umpy.object_types.plugin_argument_list),	process_handler_args() (in module pytan.binsupport), 77
375	process_print_sensors_args() (in module py-

tan.binsupport), 78 process_print_server_info_args() (in module pytan.binsupport), 78 process_pytan_shell_args() (in module pytan.binsupport), 78 process_stop_action_args() (in module pytan.binsupport), 78 process_tsat_args() (in module pytan.binsupport), 79 progresslog (pytan.pollers.QuestionPoller attribute), 56	run() (pytan.pollers.ActionPoller method), 53 run() (pytan.pollers.QuestionPoller method), 56 run() (pytan.pollers.SSEPoller method), 58 run_callback() (pytan.pollers.QuestionPoller method), 57 run_plugin() (pytan.handler.Handler method), 33 run_plugin() (pytan.sessions.Session method), 51 RunFalse, 35 RUNNING_STATUSES (pytan.pollers.ActionPoller attribute), 53
pytan (module), 3	
pytan.binsupport (module), 72	S
pytan.constants (module), 58	save() (pytan.sessions.Session method), 52
pytan.exceptions (module), 34	save_history() (pytan.binsupport.HistoryConsole static
pytan.handler (module), 3	method), 73
pytan.pollers (module), 52	SavedAction (class in tani-
pytan.sessions (module), 35	umpy.object_types.saved_action), 377
pytan.utils (module), 60	SavedActionApproval (class in tani-
pytan.xml_clean (module), 81 PytanHelp, 35	umpy.object_types.saved_action_approval), 377
	SavedActionList (class in tani-
Q	umpy.object_types.saved_action_list), 377
Q_OBJ_MAP (in module pytan.constants), 59	SavedActionPolicy (class in tani-
Question (class in taniumpy.object_types.question), 376	umpy.object_types.saved_action_policy), 377
QuestionList (class in tani- umpy.object_types.question_list), 376	SavedActionRowIdList (class in tani-
QuestionListInfo (class in tani-	umpy.object_types.saved_action_row_id_list), 377
umpy.object_types.question_list_info), 376 QuestionPoller (class in pytan.pollers), 54	SavedQuestion (class in tani-
Question offer (class in pytant.policis), 34	umpy.object_types.saved_question), 377
R	SavedQuestionList (class in tani-
RECORD_ALL_REQUESTS (pytan.sessions.Session attribute), 36	umpy.object_types.saved_question_list), 377
remove_file_log() (in module pytan.binsupport), 79	seconds_from_now() (in module pytan.utils), 70
remove_logging_handler() (in module pytan.utils), 70	seen_eq_passed_loop() (pytan.pollers.ActionPoller
replace_invalid_unicode() (in module pytan.xml_clean),	method), 54
81	Select (class in taniumpy.object_types.select), 377
replace_restricted_unicode() (in module py-tan.xml_clean), 82	SelectList (class in taniumpy.object_types.select_list), 377
REQ_KWARGS (in module pytan.constants), 59	SELECTORS (in module pytan.constants), 59
REQUEST_BODY_BASE (pytan.sessions.Session at-	Sensor (class in taniumpy.object_types.sensor), 377
tribute), 37	SENSOR_TYPE_MAP (in module pytan.constants), 60
requests (module), 382	SensorList (class in taniumpy.object_types.sensor_list),
REQUESTS_SESSION (pytan.sessions.Session at-	377 SensorQuery (class in tani-
tribute), 36	SensorQuery (class in tani- umpy.object_types.sensor_query), 377
resolverlog (pytan.pollers.QuestionPoller attribute), 56	SensorQueryList (class in tani-
RESTRICTED_UNICODE_RAW_RE (in module py-tan.xml_clean), 81	umpy.object_types.sensor_query_list), 378
RESTRICTED_UNICODE_RE (in module py-	SensorSubcolumn (class in tani-
tan.xml_clean), 81	umpy.object_types.sensor_subcolumn), 378
result_info (pytan.pollers.QuestionPoller attribute), 56	SensorSubcolumnList (class in tani-
ResultInfo (class in taniumpy.object_types.result_info), 376	umpy.object_types.sensor_subcolumn_list), 378
ResultSet (class in taniumpy.object_types.result_set), 376	server_version (pytan.sessions.Session attribute), 52
Pow (class in tanium v object types row) 376	ServerParseError, 35

ServerSideExportError, 35	shrink_obj() (in module pytan.utils), 71
Session (class in pytan.sessions), 35	SOAP_CONNECT_TIMEOUT_SEC (py-
session_id (pytan.sessions.Session attribute), 52	tan.sessions.Session attribute), 37
set_all_loglevels() (in module pytan.utils), 70	SOAP_REQUEST_HEADERS (pytan.sessions.Session
set_complect_pct() (pytan.pollers.QuestionPoller	attribute), 37
method), 57	SOAP_RES (pytan.sessions.Session attribute), 37
set_log_levels() (in module pytan.utils), 70	SOAP_RESPONSE_TIMEOUT_SEC (py-
setup_ask_manual_argparser() (in module py-	tan.sessions.Session attribute), 37
tan.binsupport), 79 setup ask saved argparser() (in module py-	SoapError (class in taniumpy.object_types.soap_error), 378
setup_ask_saved_argparser() (in module py- tan.binsupport), 79	
setup_console_logging() (in module pytan.utils), 71	spew() (in module pytan.utils), 71 spew() (in module test_pytan_invalid_server_tests), 91
setup_consore_logging() (in module pytan.utris), /1 setup_create_group_argparser() (in module py-	spew() (in module test_pytan_invalid_server_tests), 91
tan.binsupport), 79	SplitStreamHandler (class in pytan.utils), 60
setup_create_json_object_argparser() (in module py-	SSE_CRASH_MAP (in module pytan.constants), 60
tan.binsupport), 79	SSE_FORMAT_MAP (in module pytan.constants), 60
setup_create_package_argparser() (in module py-	SSE_RESTRICT_MAP (in module pytan.constants), 60
tan.binsupport), 79	sse_status_has_completed_loop() (py-
setup_create_sensor_argparser() (in module py-	tan.pollers.SSEPoller method), 58
tan.binsupport), 79	SSEPoller (class in pytan.pollers), 57
setup_create_user_argparser() (in module py-	STATS_LOOP_ENABLED (pytan.sessions.Session at-
tan.binsupport), 79	tribute), 37
setup_create_whitelisted_url_argparser() (in module py-	STATS_LOOP_SLEEP_SEC (pytan.sessions.Session at-
tan.binsupport), 79	tribute), 37
setup_delete_object_argparser() (in module py-	STATS_LOOP_TARGETS (pytan.sessions.Session at-
tan.binsupport), 79	tribute), 37
setup_deploy_action_argparser() (in module py-	statslog (pytan.sessions.Session attribute), 52
tan.binsupport), 80	stop() (pytan.pollers.QuestionPoller method), 57
setup_get_object_argparser() (in module py-	stop_action() (pytan.handler.Handler method), 33
tan.binsupport), 80	STR_ATTRS (pytan.pollers.QuestionPoller attribute), 55
setup_get_results_argparser() (in module py-	STR_ATTRS (pytan.pollers.SSEPoller attribute), 57
tan.binsupport), 80	StringHintList (class in tani-
setup_logging() (pytan.pollers.QuestionPoller method), 57	umpy.object_types.string_hint_list), 378
setup_logging() (pytan.sessions.Session method), 52	SystemSetting (class in tani- umpy.object_types.system_setting), 378
setup_loggring() (pytan.sessions.session method), 32 setup_parent_parser() (in module pytan.binsupport), 80	SystemSettingList (class in tani-
setup_parser() (in module pytan.binsupport), 80	umpy.object_types.system_setting_list),
setup_print_sensors_argparser() (in module py-	378
tan.binsupport), 80	SystemStatusAggregate (class in tani-
setup_print_server_info_argparser() (in module py-	umpy.object_types.system_status_aggregate),
tan.binsupport), 80	378
setup_pytan_shell_argparser() (in module py-	SystemStatusList (class in tani-
tan.binsupport), 80	umpy.object_types.system_status_list), 378
setup_stop_action_argparser() (in module py-	_
tan.binsupport), 80	1
$setup_test() \ (test_pytan_valid_server_tests. ValidServerTest$	Staniumpy (module), 371
method), 87	taniumpy.object_types (module), 371
setup_tsat_argparser() (in module pytan.binsupport), 80	taniumpy.object_types.action (module), 371
$set Up Class() \ (test_pytan_invalid_server_tests. InvalidServe$	
class method), 91	taniumpy.object_types.action_list_info (module), 371
setUpClass() (test_pytan_unit.TestManualBuildObjectUtils	
class method), 85	taniumpy.object_types.action_stop_list (module), 371
setUpClass() (test_pytan_valid_server_tests.ValidServerTest class method), 87	Praniumpy.object_types.all_objects (module), 371
ciass incurva), or	taniumpy.object_types.archived_question (module), 371

taniumpy.object_types.archived_question_list (module),	taniumpy.object_types.plugin_list (module), 375
371	taniumpy.object_types.plugin_schedule (module), 375
taniumpy.object_types.audit_data (module), 371	taniumpy.object_types.plugin_schedule_list (module),
taniumpy.object_types.base (module), 371	376
taniumpy.object_types.cache_filter (module), 372	taniumpy.object_types.plugin_sql (module), 376
taniumpy.object_types.cache_filter_list (module), 372	taniumpy.object_types.plugin_sql_column (module), 376
taniumpy.object_types.cache_info (module), 373	taniumpy.object_types.plugin_sql_result (module), 376
taniumpy.object_types.client_count (module), 373	taniumpy.object_types.question (module), 376
taniumpy.object_types.client_status (module), 373	taniumpy.object_types.question_list (module), 376
taniumpy.object_types.column (module), 373	taniumpy.object_types.question_list_info (module), 376
taniumpy.object_types.column_set (module), 373	taniumpy.object_types.result_info (module), 376
taniumpy.object_types.computer_group (module), 373	taniumpy.object_types.result_set (module), 376
taniumpy.object_types.computer_group_list (module),	taniumpy.object_types.row (module), 376
373	taniumpy.object_types.saved_action (module), 377
taniumpy.object_types.computer_group_spec (module), 373	taniumpy.object_types.saved_action_approval (module), 377
taniumpy.object_types.computer_spec_list (module), 373	taniumpy.object_types.saved_action_list (module), 377
taniumpy.object_types.error_list (module), 373	taniumpy.object_types.saved_action_policy (module),
taniumpy.object_types.filter (module), 373	377
taniumpy.object_types.filter_list (module), 373	taniumpy.object_types.saved_action_row_id_list (mod-
taniumpy.object_types.group (module), 373	ule), 377
taniumpy.object_types.group_list (module), 373	taniumpy.object_types.saved_question (module), 377
taniumpy.object_types.metadata_item (module), 374	taniumpy.object_types.saved_question_list (module), 377
taniumpy.object_types.metadata_list (module), 374	taniumpy.object_types.select (module), 377
taniumpy.object_types.object_list (module), 374	taniumpy.object_types.select_list (module), 377
taniumpy.object_types.object_list_types (module), 374	taniumpy.object_types.sensor (module), 377
taniumpy.object_types.options (module), 374	taniumpy.object_types.sensor_list (module), 377
taniumpy.object_types.package_file (module), 374	taniumpy.object_types.sensor_query (module), 377
taniumpy.object_types.package_file_list (module), 374	taniumpy.object_types.sensor_query_list (module), 378
taniumpy.object_types.package_file_status (module), 374	taniumpy.object_types.sensor_subcolumn (module), 378
taniumpy.object_types.package_file_status_list (module),	taniumpy.object_types.sensor_subcolumn_list (module),
374	378
taniumpy.object_types.package_file_template (module),	taniumpy.object_types.sensor_types (module), 378
374	taniumpy.object_types.soap_error (module), 378
taniumpy.object_types.package_file_template_list (mod-	taniumpy.object_types.string_hint_list (module), 378
ule), 374	taniumpy.object_types.system_setting (module), 378
taniumpy.object_types.package_spec (module), 374	taniumpy.object_types.system_setting_list (module), 378
taniumpy.object_types.package_spec_list (module), 374	taniumpy.object_types.system_status_aggregate (mod-
taniumpy.object_types.parameter (module), 374	ule), 378
taniumpy.object_types.parameter_list (module), 375	taniumpy.object_types.system_status_list (module), 378
taniumpy.object_types.parse_job (module), 375	taniumpy.object_types.upload_file (module), 378
taniumpy.object_types.parse_job_list (module), 375	taniumpy.object_types.upload_file_list (module), 378
taniumpy.object_types.parse_result (module), 375	taniumpy.object_types.upload_file_status (module), 378
taniumpy.object_types.parse_result_group (module), 375	taniumpy.object_types.user (module), 378
taniumpy.object_types.parse_result_group_list (module),	taniumpy.object_types.user_list (module), 378
375	taniumpy.object_types.user_role (module), 379
taniumpy.object_types.parse_result_list (module), 375	taniumpy.object_types.user_role_list (module), 379
taniumpy.object_types.permission_list (module), 375	taniumpy.object_types.version_aggregate (module), 379
taniumpy.object_types.plugin (module), 375	taniumpy.object_types.version_aggregate_list (module),
taniumpy.object_types.plugin_argument (module), 375	379
taniumpy.object_types.plugin_argument_list (module),	taniumpy.object_types.white_listed_url (module), 379
375	taniumpy.object_types.white_listed_url_list (module),
taniumpy.object_types.plugin_command_list (module),	379
375	taniumpy.object_types.xml_error (module), 379

tearDownClass() (test_pytan_valid_server_tests.ValidServe class method), 87	rTests	(test_pytan_unit.TestDehumanizeExtractionUtils method), 83
test_app_port() (in module pytan.utils), 71	test extra	act_filter_valid()
test_bad_chars_basetype_control()	test_extr	(test_pytan_unit.TestDehumanizeExtractionUtils
(test_pytan_unit.TestDeserializeBadXML		method), 83
method), 84	test extra	act_filter_valid_all()
test_bad_chars_resultset_latin1()	test_extr	(test_pytan_unit.TestDehumanizeExtractionUtils
(test_pytan_unit.TestDeserializeBadXML		method), 83
method), 84	test extre	act_options_invalid_option()
test_bad_chars_resultset_surrogate()	test_extra	(test_pytan_unit.TestDehumanizeExtractionUtils
(test_pytan_unit.TestDeserializeBadXML		method), 83
method), 84	tost avtre	act_options_many()
· · · · · · · · · · · · · · · · · · ·		
test_build_group_obj() (test_pytan_unit.TestManualBuildObjectUtils (test_pytan_unit.TestDehumanizeExtractionUtils method), 85 method), 83		
test_build_manual_q() (test_pytan_unit.TestManualBuildO	byext <u>t</u> letatsa	
method), 85		(test_pytan_unit.TestDehumanizeExtractionUtils
test_build_selectlist_obj_invalid_filter()		method), 83
(test_pytan_unit.TestManualBuildObjectUtils	test_extra	act_options_missing_value_value_type()
method), 85		(test_pytan_unit.TestDehumanizeExtractionUtils
test_build_selectlist_obj_missing_value()		method), 83
(test_pytan_unit.TestManualBuildObjectUtils	test_extra	act_options_nooptions()
method), 85		(test_pytan_unit.TestDehumanizeExtractionUtils
test_build_selectlist_obj_noparamssensorobj_noparams()		method), 83
(test_pytan_unit.TestManualBuildObjectUtils	test_extra	act_options_single()
method), 85		(test_pytan_unit.TestDehumanizeExtractionUtils
test_build_selectlist_obj_noparamssensorobj_withparams()		method), 83
(test_pytan_unit.TestManualBuildObjectUtils	test_extra	act_params() (test_pytan_unit.TestDehumanizeExtractionUtils
method), 85		method), 83
test_build_selectlist_obj_withparamssensorobj_noparams()	test_extra	
(test_pytan_unit.TestManualBuildObjectUtils		(test_pytan_unit.TestDehumanizeExtractionUtils
method), 85	6	method), 83
test_build_selectlist_obj_withparamssensorobj_withparams	s(t)est_extra	
(test_pytan_unit.TestManualBuildObjectUtils		(test_pytan_unit.TestDehumanizeExtractionUtils
method), 85		method), 83
test_empty_args_dict() (test_pytan_unit.TestDehumanizeSe	ensor <u>e</u> misa	
method), 84	naami itila	(test_pytan_unit.TestDehumanizeExtractionUtils
test_empty_args_list() (test_pytan_unit.TestDehumanizeSet		
method), 84 test_empty_args_str() (test_pytan_unit.TestDehumanizeSer		act_selector() (test_pytan_unit.TestDehumanizeExtractionUtils
method), 84		act_selector_use_name_if_noselector()
test_empty_filterlist() (test_pytan_unit.TestDehumanizeQuo		
method), 83	estioni iic	method), 83
test_empty_filterstr() (test_pytan_unit.TestDehumanizeQue	ettaon Fridto	
method), 83	Surbi <u>ng</u> ut <u>e</u> i	method), 84
test_empty_obj() (test_pytan_unit.TestGenericUtils	test get	
method), 84	icsi_gci_	method), 84
test_empty_optionlist() (test_pytan_unit.TestDehumanizeQ	utesticm(f)n	
method), 83	are briegen p	method), 84
	ıtestioin(Va	tioh() (itsst_pytan_unit.TestManualPackageDefValidateUtils
method), 83	<u></u>	method), 85
test_extract_filter_invalid()	test inval	lid1() (test_pytan_unit.TestManualQuestionFilterDefValidateUtils
(test_pytan_unit.TestDehumanizeExtractionUtils		method), 86
method), 83	test inval	lid1() (test_pytan_unit.TestManualSensorDefValidateUtils
test_extract_filter_nofilter()		method), 87

method), 88

method), 88

 $(test_pytan_valid_server_tests.ValidServerTests$

test_invalid_deploy_action_7_invalid_deploy_action_missing_parameters(od), 83

```
test_invalid2() (test_pytan_unit.TestManualPackageDefValidateUtils (test_pytan_valid_server_tests.ValidServerTests
                   method), 85
                                                                                                                                       method), 88
test_invalid2() (test_pytan_unit.TestManualSensorDefValidatesUtihsvalid_export_basetype_1_invalid_export_basetype_csv_bad_explode
                   method), 87
                                                                                                                                       (test\_pytan\_valid\_server\_tests.ValidServerTests
test_invalid3() (test_pytan_unit.TestManualSensorDefValidateUtils
                                                                                                                                      method), 88
                   method), 87
                                                                                                                   test_invalid_export_basetype_2_invalid_export_basetype_csv_bad_sort_sul
test\_invalid4() \ (test\_pytan\_unit. TestManual Sensor Def Validate Utils
                                                                                                                                      (test pytan valid server tests. ValidServerTests
                   method), 87
                                                                                                                                       method), 88
test_invalid_connect_1_bad_username()
                                                                                                                   test_invalid_export_basetype_3_invalid_export_basetype_csv_bad_sort_type_sort_type_sort_basetype_sort_type_sort_basetype_sort_basetype_sort_type_sort_basetype_sort_basetype_sort_basetype_sort_type_sort_basetype_sort_basetype_sort_basetype_sort_basetype_sort_basetype_sort_basetype_sort_basetype_sort_basetype_sort_basetype_sort_basetype_sort_basetype_sort_basetype_sort_basetype_sort_basetype_sort_basetype_sort_basetype_sort_basetype_sort_basetype_sort_basetype_sort_basetype_sort_basetype_sort_basetype_sort_basetype_sort_basetype_sort_basetype_sort_basetype_sort_basetype_sort_basetype_sort_basetype_sort_basetype_sort_basetype_sort_basetype_sort_basetype_sort_basetype_sort_basetype_sort_basetype_sort_basetype_sort_basetype_sort_basetype_sort_basetype_sort_basetype_sort_basetype_sort_basetype_sort_basetype_sort_basetype_sort_basetype_sort_basetype_sort_basetype_sort_basetype_sort_basetype_sort_basetype_sort_basetype_sort_basetype_sort_basetype_sort_basetype_sort_basetype_sort_basetype_sort_basetype_sort_basetype_sort_basetype_sort_basetype_sort_basetype_sort_basetype_sort_basetype_sort_basetype_sort_basetype_sort_basetype_sort_basetype_sort_basetype_sort_basetype_sort_basetype_sort_basetype_sort_basetype_sort_basetype_sort_basetype_sort_basetype_sort_basetype_sort_basetype_sort_basetype_sort_basetype_sort_basetype_sort_basetype_sort_basetype_sort_basetype_sort_basetype_sort_basetype_sort_basetype_sort_basetype_sort_basetype_sort_basetype_sort_basetype_sort_basetype_sort_basetype_sort_basetype_sort_basetype_sort_basetype_sort_basetype_sort_basetype_sort_basetype_sort_basetype_sort_basetype_sort_basetype_sort_basetype_sort_basetype_sort_basetype_sort_basetype_sort_basetype_sort_basetype_sort_basetype_sort_basetype_sort_basetype_sort_basetype_sort_basetype_sort_basetype_sort_basetype_sort_basetype_sort_basetype_sort_basetype_sort_basetype_sort_basetype_sort_basetype_sort_basetype_sort_basetype_sort_basetype_sort_basetype_sort_basetype_sort_basetype_sort_basetype_sort_basetype_sort_basetype_sort_basetype_sort_basetype_s
                   (test_pytan_invalid_server_tests.InvalidServerTests
                                                                                                                                       (test\_pytan\_valid\_server\_tests. ValidServerTests
                   method), 91
                                                                                                                                       method), 88
test_invalid_connect_2_bad_host_and_non_ssl_port()
                                                                                                                   test_invalid_export_basetype_4_invalid_export_basetype_xml_bad_minima
                   (test_pytan_invalid_server_tests.InvalidServerTests
                                                                                                                                       (test_pytan_valid_server_tests.ValidServerTests
                   method), 91
                                                                                                                                       method), 88
test_invalid_connect_3_bad_password()
                                                                                                                   test_invalid_export_basetype_5_invalid_export_basetype_json_bad_include
                                                                                                                                       (test_pytan_valid_server_tests.ValidServerTests
                   (test_pytan_invalid_server_tests.InvalidServerTests
                   method), 91
                                                                                                                                       method), 88
test_invalid_connect_4_bad_host_and_bad_port()
                                                                                                                   test_invalid_export_basetype_6_invalid_export_basetype_json_bad_explod
                   (test_pytan_invalid_server_tests.InvalidServerTests
                                                                                                                                       (test_pytan_valid_server_tests.ValidServerTests
                   method), 91
                                                                                                                                       method), 88
test_invalid_create_object_1_invalid_create_sensor()
                                                                                                                   test_invalid_export_basetype_7_invalid_export_basetype_bad_format()
                   (test\_pytan\_valid\_server\_tests.ValidServerTests
                                                                                                                                       (test\_pytan\_valid\_server\_tests.ValidServerTests
                   method), 87
                                                                                                                                       method), 88
test_invalid_create_object_from_json_1_invalid_create_savectsaccentering invalid_export_resultset_1_invalid_export_resultset_csv_bad_sort_subset_1_invalid_export_resultset_csv_bad_sort_subset_1_invalid_export_resultset_csv_bad_sort_subset_1_invalid_export_resultset_1_invalid_export_resultset_1_invalid_export_resultset_1_invalid_export_resultset_1_invalid_export_resultset_1_invalid_export_resultset_1_invalid_export_resultset_1_invalid_export_resultset_1_invalid_export_resultset_1_invalid_export_resultset_1_invalid_export_resultset_1_invalid_export_resultset_1_invalid_export_resultset_1_invalid_export_resultset_1_invalid_export_resultset_1_invalid_export_resultset_1_invalid_export_resultset_1_invalid_export_resultset_1_invalid_export_resultset_1_invalid_export_resultset_1_invalid_export_resultset_1_invalid_export_resultset_1_invalid_export_resultset_1_invalid_export_resultset_1_invalid_export_resultset_1_invalid_export_resultset_1_invalid_export_resultset_1_invalid_export_resultset_1_invalid_export_resultset_1_invalid_export_resultset_1_invalid_export_resultset_1_invalid_export_resultset_1_invalid_export_resultset_1_invalid_export_resultset_1_invalid_export_resultset_1_invalid_export_resultset_1_invalid_export_resultset_1_invalid_export_resultset_1_invalid_export_resultset_1_invalid_export_resultset_1_invalid_export_resultset_1_invalid_export_resultset_1_invalid_export_resultset_1_invalid_export_resultset_1_invalid_export_resultset_1_invalid_export_resultset_1_invalid_export_resultset_1_invalid_export_resultset_1_invalid_export_resultset_1_invalid_export_resultset_1_invalid_export_resultset_1_invalid_export_resultset_1_invalid_export_resultset_1_invalid_export_resultset_1_invalid_export_resultset_1_invalid_export_resultset_1_invalid_export_resultset_1_invalid_export_resultset_1_invalid_export_resultset_1_invalid_export_resultset_1_invalid_export_resultset_1_invalid_export_resultset_1_invalid_export_resultset_1_invalid_export_resultset_1_invalid_export_resultset_1_invalid_export_resultset_1_invalid_export_resultse
                   (test_pytan_valid_server_tests.ValidServerTests
                                                                                                                                       (test\_pytan\_valid\_server\_tests.ValidServerTests
                   method), 87
                                                                                                                                       method), 88
test_invalid_create_object_from_json_2_invalid_create_cliettest_fromvaljstone@port_resultset_2_invalid_export_resultset_csv_bad_sort_typ
                   (test_pytan_valid_server_tests.ValidServerTests
                                                                                                                                       (test_pytan_valid_server_tests.ValidServerTests
                   method), 87
                                                                                                                                       method), 88
test_invalid_create_object_from_json_3_invalid_create_useresteinfvalid_jssp@rt_resultset_3_invalid_export_resultset_csv_bad_expand_
                   (test_pytan_valid_server_tests.ValidServerTests
                                                                                                                                       (test_pytan_valid_server_tests.ValidServerTests
                   method), 87
                                                                                                                                       method), 88
test_invalid_create_object_from_json_4_invalid_create_setting_fromligs@xport_resultset_4_invalid_export_resultset_csv_bad_sensors_
                   (test_pytan_valid_server_tests.ValidServerTests
                                                                                                                                       (test_pytan_valid_server_tests.ValidServerTests
                                                                                                                                       method), 88
                   method), 87
test_invalid_deploy_action_1_invalid_deploy_action_run_fabsat()invalid_export_resultset_5_invalid_export_resultset_bad_format()
                   (test_pytan_valid_server_tests.ValidServerTests
                                                                                                                                       (test pytan valid server tests. ValidServerTests
                                                                                                                                       method), 88
test_invalid_deploy_action_2_invalid_deploy_action_packagesthelp@lid_filter1() (test_pytan_unit.TestDehumanizeQuestionFilterUtils
                   (test\_pytan\_valid\_server\_tests.ValidServerTests
                                                                                                                                       method), 83
                   method), 87
                                                                                                                   test invalid filter2() (test pytan unit.TestDehumanizeQuestionFilterUtils
test_invalid_deploy_action_3_invalid_deploy_action_package()
                                                                                                                                       method), 83
                   (test_pytan_valid_server_tests.ValidServerTests test_invalid_filter3() (test_pytan_unit.TestDehumanizeQuestionFilterUtils
                   method), 87
                                                                                                                                       method), 83
test\_invalid\_deploy\_action\_4\_invalid\_deploy\_action\_option \\ \textbf{test\_invalid\_get\_object\_1\_invalid\_get\_action\_single\_by\_name()}
                   (test_pytan_valid_server_tests.ValidServerTests
                                                                                                                                       (test_pytan_valid_server_tests.ValidServerTests
                   method), 88
                                                                                                                                       method), 88
test_invalid_deploy_action_5_invalid_deploy_action_emptyteptackaadd_get_object_2_invalid_get_question_by_name()
                   (test\_pytan\_valid\_server\_tests. ValidServerTests
                                                                                                                                       (test_pytan_valid_server_tests.ValidServerTests
```

398 Index

method), 88

method), 83

test_invalid_option2() (test_pytan_unit.TestDehumanizeQuestionOptionUti

```
test_invalid_port()
                                                         (test_pytan_unit.TestGenericUtils
                                                                                                                                                                           (test_pytan_unit.TestGenericUtils
                                                                                                                                                                                                                                                                        method),
                        method), 85
                                                                                                                                                                           85
test invalid question 1 invalid ask manual question sensteesthed (d) taniumpy file invalid ison()
                        (test_pytan_valid_server_tests.ValidServerTests
                                                                                                                                                                           (test_pytan_unit.TestGenericUtils
                                                                                                                                                                                                                                                                        method),
                        method), 88
test invalid question 2 invalid ask manual question paratoexateurus blitfilter list() (test pytan unit. Test Dehumanize Question Filter Utils
                        (test pytan valid server tests. ValidServerTests
                                                                                                                                                                          method), 83
                        method), 88
                                                                                                                                                  test_multi_list_complex()
test_invalid_question_3_invalid_ask_manual_question_filter_help() (test_pytan_unit.TestDehumanizeSensorUtils
                        (test_pytan_valid_server_tests.ValidServerTests
                                                                                                                                                                          method), 84
                        method), 88
                                                                                                                                                  test_option_list_many() (test_pytan_unit.TestDehumanizeQuestionOptionU
test_invalid_question_4_invalid_ask_manual_question_option()
                                                                                                                                                                          method), 83
                        (test_pytan_valid_server_tests.ValidServerTests test_option_list_multi() (test_pytan_unit.TestDehumanizeQuestionOptionU
                                                                                                                                                                          method), 84
                        method), 88
test_invalid_question_5_invalid_ask_manual_question_senstors(t)_option_list_single() (test_pytan_unit.TestDehumanizeQuestionOptionU
                        (test_pytan_valid_server_tests.ValidServerTests
                                                                                                                                                                           method), 84
                        method), 88
                                                                                                                                                  test_option_str() (test_pytan_unit.TestDehumanizeQuestionOptionUtils
test_invalid_question_6_invalid_ask_manual_question_option_help() method), 84
                        (test_pytan_valid_server_tests.ValidServerTests test_parse_complex() (test_pytan_unit.TestManualSensorDefParseUtils
                        method), 88
                                                                                                                                                                          method), 86
test_invalid_question_7_invalid_ask_manual_question_paratoextrepatsoe_dinatn_Massh() (test_pytan_unit.TestManualSensorDefParseUtils
                        (test\_pytan\_valid\_server\_tests.ValidServerTests
                                                                                                                                                                          method), 86
                        method), 88
                                                                                                                                                  test_parse_dict_id() (test_pytan_unit.TestManualSensorDefParseUtils
test invalid question 8 invalid ask manual question filter()
                                                                                                                                                                          method), 86
                        (test_pytan_valid_server_tests.ValidServerTests test_parse_dict_name() (test_pytan_unit.TestManualSensorDefParseUtils
                        method), 88
                                                                                                                                                                          method), 86
test_is_dict() (test_pytan_unit.TestGenericUtils method),
                                                                                                                                                  test_parse_emptydict() (test_pytan_unit.TestManualQuestionFilterDefParse
                                                                                                                                                                          method), 86
test_is_list() (test_pytan_unit.TestGenericUtils method),
                                                                                                                                                  test_parse_emptydict() (test_pytan_unit.TestManualQuestionOptionDefPars
                        85
                                                                                                                                                                          method), 86
test_is_not_dict()
                                                          (test_pytan_unit.TestGenericUtils
                                                                                                                                                  test_parse_emptydict() (test_pytan_unit.TestManualSensorDefParseUtils
                        method), 85
                                                                                                                                                                          method), 86
                                                                                                                                                  test_parse_emptylist() (test_pytan_unit.TestManualQuestionFilterDefParse
test_is_not_list()
                                                         (test_pytan_unit.TestGenericUtils
                                                                                                                                                                          method), 86
                        method), 85
test_is_not_num()
                                                         (test_pytan_unit.TestGenericUtils
                                                                                                                                                  test_parse_emptylist() (test_pytan_unit.TestManualQuestionOptionDefPars
                        method), 85
                                                                                                                                                                          method), 86
test_is_not_str()
                                                         (test pytan unit.TestGenericUtils
                                                                                                                                                  test parse emptylist() (test pytan unit.TestManualSensorDefParseUtils
                        method), 85
                                                                                                                                                                          method), 86
test_is_num() (test_pytan_unit.TestGenericUtils method),
                                                                                                                                                  test\_parse\_emptystr() (test\_pytan\_unit.TestManualQuestionFilterDefParseUptractionFilterDefParseUptractionFilterDefParseUptractionFilterDefParseUptractionFilterDefParseUptractionFilterDefParseUptractionFilterDefParseUptractionFilterDefParseUptractionFilterDefParseUptractionFilterDefParseUptractionFilterDefParseUptractionFilterDefParseUptractionFilterDefParseUptractionFilterDefParseUptractionFilterDefParseUptractionFilterDefParseUptractionFilterDefParseUptractionFilterDefParseUptractionFilterDefParseUptractionFilterDefParseUptractionFilterDefParseUptractionFilterDefParseUptractionFilterDefParseUptractionFilterDefParseUptractionFilterDefParseUptractionFilterDefParseUptractionFilterDefParseUptractionFilterDefParseUptractionFilterDefParseUptractionFilterDefParseUptractionFilterDefParseUptractionFilterDefParseUptractionFilterDefParseUptractionFilterDefParseUptractionFilterDefParseUptractionFilterParseUptractionFilterParseUptractionFilterParseUptractionFilterParseUptractionFilterParseUptractionFilterParseUptractionFilterParseUptractionFilterParseUptractionFilterParseUptractionFilterParseUptractionFilterParseUptractionFilterParseUptractionFilterParseUptractionFilterParseUptractionFilterParseUptractionFilterParseUptractionFilterParseUptractionFilterParseUptractionFilterParseUptractionFilterParseUptractionFilterParseUptractionFilterParseUptractionFilterParseUptractionFilterParseUptractionFilterParseUptractionFilterParseUptractionFilterParseUptractionFilterParseUptractionFilterParseUptractionFilterParseUptractionFilterParseUptractionFilterParseUptractionFilterParseUptractionFilterParseUptractionFilterParseUptractionFilterParseUptractionFilterParseUptractionFilterParseUptractionFilterParseUptractionFilterParseUptractionFilterParseUptractionFilterParseUptractionFilterParseUptractionFilterParseUptractionFilterParseUptractionFilterParseUptractionFilterParseUptractionFilterParseUptractionFilterParseUptractionFilterParseUptractionFilterParseUptractionFilterParseUptractionFilterParseUptractionFilterParseUptractionFilterParse
                                                                                                                                                                          method), 86
test is str() (test pytan unit.TestGenericUtils method),
                                                                                                                                                  test parse emptystr() (test pytan unit.TestManualQuestionOptionDefParse
                                                                                                                                                                          method), 86
test_jsonify() (test_pytan_unit.TestGenericUtils method),
                                                                                                                                                  test parse emptystr() (test pytan unit.TestManualSensorDefParseUtils
                                                                                                                                                                          method), 87
test_load_param_file_invalid_file()
                                                                                                                                                  test\_parse\_list() \ (test\_pytan\_unit. TestManual Question Option Def Parse Utils \\
                        (test_pytan_unit.TestGenericUtils
                                                                                                                                                                          method), 86
                                                                                                                      method),
                                                                                                                                                  test_parse_multi_filter() (test_pytan_unit.TestManualQuestionFilterDefPars
test_load_param_file_invalid_json()
                                                                                                                                                                          method), 86
                        (test_pytan_unit.TestGenericUtils
                                                                                                                      method),
                                                                                                                                                  test\_parse\_noargs() \ (test\_pytan\_unit. TestManual Question Filter Def Parse Utille States and the partial of the partial of
                                                                                                                                                                          method), 86
test_load_param_file_valid()
                                                                                                                                                  test\_parse\_noargs() \ (test\_pytan\_unit. TestManual Question Option Def Parse Unit TestManual Question Option Opt
                        (test pytan unit.TestGenericUtils
                                                                                                                      method),
                                                                                                                                                                          method), 86
                                                                                                                                                  test_parse_noargs() (test_pytan_unit.TestManualSensorDefParseUtils
```

method), 87

test load taniumpy file invalid file()

```
test_parse_none() (test_pytan_unit.TestManualQuestionFilterDefPalrse_Utelate_object_2_create package()
              method), 86
                                                                                                       (test pytan valid server tests. ValidServerTests
test parse none() (test pytan unit.TestManualQuestionOptionDefParsaetfibed), 88
              method), 86
                                                                                        test_valid_create_object_3_create_group()
test_parse_none() (test_pytan_unit.TestManualSensorDefParseUtils (test_pytan_valid_server_tests.ValidServerTests
              method), 87
                                                                                                       method), 88
test parse options dict()
                                                                                        test valid create object 4 create whitelisted url()
              (test pytan unit.TestManualQuestionOptionDefParseUtils (test pytan valid server tests.ValidServerTests
              method), 86
                                                                                                       method), 88
test_parse_single_filter() (test_pytan_unit.TestManualQuestionFiltediDefFeatse_ldti]sct_from_json_1_create_package_from_json()
              method), 86
                                                                                                       (test_pytan_valid_server_tests.ValidServerTests
test_parse_str() (test_pytan_unit.TestManualQuestionFilterDefParseUtilethod), 88
              method), 86
                                                                                        test_valid_create_object_from_json_2_create_user_from_json()
test_parse_str() (test_pytan_unit.TestManualQuestionOptionDefParseVtti$ts_pytan_valid_server_tests.ValidServerTests
              method), 86
                                                                                                       method), 88
test_parse_str1() (test_pytan_unit.TestManualSensorDefPartelItilsalid_create_object_from_json_3_create_saved_question_from_json()
                                                                                                       (test_pytan_valid_server_tests.ValidServerTests
              method), 87
test pytan invalid server tests (module), 91
                                                                                                       method), 88
test pytan unit (module), 83
                                                                                        test_valid_create_object_from_json_4_create_action_from_json()
                                                                                                       (test pytan valid server tests. ValidServerTests
test pytan valid server tests (module), 87
test_single_filter_list() (test_pytan_unit.TestDehumanizeQuestionFilterrlettlod), 88
              method), 83
                                                                                        test_valid_create_object_from_json_5_create_sensor_from_json()
test_single_filter_str() (test_pytan_unit.TestDehumanizeQuestionFilter_tests_bytan_valid_server_tests.ValidServerTests
              method), 83
                                                                                                       method), 88
test_single_str() (test_pytan_unit.TestDehumanizeSensorUtillest_valid_create_object_from_json_6_create_question_from_json()
              method), 84
                                                                                                       (test pytan valid server tests. ValidServerTests
test_single_str_complex1()
                                                                                                       method), 88
              (test_pytan_unit.TestDehumanizeSensorUtils
                                                                                        test_valid_create_object_from_json_7_create_whitelisted_url_from_json()
                                                                                                       (test_pytan_valid_server_tests.ValidServerTests
              method), 84
test_single_str_complex2()
                                                                                                       method), 88
              (test_pytan_unit.TestDehumanizeSensorUtils
                                                                                        test_valid_create_object_from_json_8_create_group_from_json()
                                                                                                       (test_pytan_valid_server_tests.ValidServerTests
              method), 84
test_single_str_with_filter()
                                                                                                       method), 89
              (test\_pytan\_unit.TestDehumanizeSensorUtils
                                                                                        test_valid_deploy_action_1_deploy_action_simple_against_windows_comp
              method), 84
                                                                                                       (test pytan valid server tests. ValidServerTests
test\_valid1() \ (test\_pytan\_unit. TestManual Package Def Validate Utils
                                                                                                       method), 89
                                                                                        test valid deploy action 2 deploy action simple without results()
              method), 85
test\_valid1() \ (test\_pytan\_unit. TestManual Question Filter Def Validate U \ (test\_pytan\_valid\_server\_tests. Valid Server Tests) \ (test\_pytan\_unit. TestManual Question Filter Def Validate U \ (test\_pytan\_valid\_server\_tests. Valid Server Tests) \ (test\_pytan\_valid\_server\_tests) \ (test\_pytan\_valid\_server\_tes
                                                                                                       method), 89
              method), 86
test_valid1() (test_pytan_unit.TestManualSensorDefValidateExtilevalid_deploy_action_3_deploy_action_with_params_against_windows
                                                                                                       (test pytan valid server tests. ValidServerTests
              method), 87
                                                                                                       method), 89
test valid2() (test pytan unit.TestManualPackageDefValidateUtils
              method), 86
                                                                                        test valid deploy action 4 deploy action simple()
test_valid2() (test_pytan_unit.TestManualQuestionFilterDefValidateU(test_pytan_valid_server_tests.ValidServerTests
              method), 86
                                                                                                       method), 89
test_valid2() (test_pytan_unit.TestManualSensorDefValidateExtilsvalid_export_basetype_10_export_basetype_xml_default_options()
              method), 87
                                                                                                       (test_pytan_valid_server_tests.ValidServerTests
test_valid3() (test_pytan_unit.TestManualSensorDefValidateUtils
                                                                                                       method), 89
              method), 87
                                                                                        test_valid_export_basetype_11_export_basetype_csv_with_explode_true()
test_valid4() (test_pytan_unit.TestManualSensorDefValidateUtils
                                                                                                       (test_pytan_valid_server_tests.ValidServerTests
              method), 87
                                                                                                       method), 89
test_valid_create_object_1_create_user()
                                                                                        test_valid_export_basetype_12_export_basetype_json_explode_false()
              (test pytan valid server tests. ValidServerTests
                                                                                                       (test pytan valid server tests. ValidServerTests
              method), 88
                                                                                                       method), 89
```

- test_valid_export_basetype_13_export_basetype_json_type_teatse@lid_export_resultset_4_export_resultset_csv_expand_false() (test_pytan_valid_server_tests.ValidServerTests method), 89 (test_pytan_valid_server_tests.ValidServerTests method), 89
- test_valid_export_basetype_1_export_basetype_csv_with_start_lixat()d_export_resultset_6_export_resultset_csv_sort_true() (test_pytan_valid_server_tests.ValidServerTests method), 89 (test_pytan_valid_server_tests.ValidServerTests method), 89
- test_valid_export_basetype_2_export_basetype_csv_with_explode_lift_lsex(port_resultset_7_export_resultset_csv_sort_list() (test_pytan_valid_server_tests.ValidServerTests method), 89 (test_pytan_valid_server_tests.ValidServerTests method), 89
- test_valid_export_basetype_3_export_basetype_json_type_tment()valid_export_resultset_8_export_resultset_csv_sensor_false() (test_pytan_valid_server_tests.ValidServerTests method), 89 (test_pytan_valid_server_tests.ValidServerTests method), 89
- test_valid_export_basetype_4_export_basetype_xml_minimtelstfallabid_export_resultset_9_export_resultset_csv_expand_true() (test_pytan_valid_server_tests.ValidServerTests method), 89 (test_pytan_valid_server_tests.ValidServerTests method), 89
- test_valid_export_basetype_6_export_basetype_csv_with_start_evality_distt()pject_11_get_user_by_name() (test_pytan_valid_server_tests.ValidServerTests method), 89 (test_pytan_valid_server_tests.ValidServerTests method), 89
- test_valid_export_basetype_7_export_basetype_csv_defaulttesptivalid_get_object_12_get_all_userroless() (test_pytan_valid_server_tests.ValidServerTests method), 89 (test_pytan_valid_server_tests.ValidServerTests method), 89
- test_valid_export_basetype_8_export_basetype_json_explo**desttrwa**(i)d_get_object_13_get_all_questions() (test_pytan_valid_server_tests.ValidServerTests method), 89 (test_pytan_valid_server_tests.ValidServerTests method), 89
- test_valid_export_basetype_9_export_basetype_csv_with_s**terst_tmab**(d_get_object_14_get_sensor_by_id() (test_pytan_valid_server_tests.ValidServerTests method), 89 (test_pytan_valid_server_tests.ValidServerTests method), 90
- test_valid_export_resultset_11_export_resultset_csv_type_trust()valid_get_object_16_get_all_sensors() (test_pytan_valid_server_tests.ValidServerTests method), 89 (test_pytan_valid_server_tests.ValidServerTests method), 90
- test_valid_export_resultset_13_export_resultset_csv_sort_fatbset()valid_get_object_18_get_whitelisted_url_by_id() (test_pytan_valid_server_tests.ValidServerTests method), 89 (test_pytan_valid_server_tests.ValidServerTests method), 90
- test_valid_export_resultset_1_export_resultset_json() test_pytan_valid_server_tests. ValidServerTests method), 89 test_valid_get_object_19_get_group_by_name() (test_pytan_valid_server_tests. ValidServerTests method), 90
- test_valid_export_resultset_2_export_resultset_csv_sensor_teste()valid_get_object_1_get_all_users() (test_pytan_valid_server_tests.ValidServerTests method), 89 (test_pytan_valid_server_tests.ValidServerTests method), 90
- test_valid_export_resultset_3_export_resultset_csv_type_fal**ss()** valid_get_object_20_get_all_whitelisted_urls() (test_pytan_valid_server_tests.ValidServerTests method), 89 (test_pytan_valid_server_tests.ValidServerTests method), 90

- test_valid_get_object_21_get_sensor_by_hash() (test_pytan_valid_server_tests.ValidServerTests method), 90
- test_valid_get_object_22_get_package_by_name() (test_pytan_valid_server_tests.ValidServerTests method), 90
- test_valid_get_object_23_get_all_clients() $(test_pytan_valid_server_tests.ValidServerTests$ method), 90
- test_valid_get_object_24_get_sensor_by_names() (test_pytan_valid_server_tests.ValidServerTests method), 90
- test_valid_get_object_25_get_all_packages() (test_pytan_valid_server_tests.ValidServerTests method), 90
- (test_pytan_valid_server_tests.ValidServerTests method), 90
- test_valid_get_object_27_get_all_actions() (test pytan valid server tests. ValidServerTests method), 90
- test_valid_get_object_28_get_user_by_id() $(test_pytan_valid_server_tests.ValidServerTests$ method), 90
- test_valid_get_object_29_get_sensor_by_name() (test pytan valid server tests. ValidServerTests method), 90
- test_valid_get_object_2_get_action_by_id() (test_pytan_valid_server_tests.ValidServerTests method), 90
- test_valid_get_object_30_get_saved_action_by_name() (test_pytan_valid_server_tests.ValidServerTests method), 90
- test_valid_get_object_3_get_question_by_id() (test pytan valid server tests. ValidServerTests method), 90
- test_valid_get_object_4_get_saved_question_by_names() $(test_pytan_valid_server_tests.ValidServerTests$ method), 90
- test_valid_get_object_5_get_userrole_by_id() (test pytan valid server tests. ValidServerTests method), 90
- test_valid_get_object_6_get_all_saved_actions() $(test_pytan_valid_server_tests. ValidServerTests$ method), 90
- test_valid_get_object_7_get_leader_clients() (test_pytan_valid_server_tests.ValidServerTests method), 90
- test_valid_get_object_8_get_all_settings() (test_pytan_valid_server_tests.ValidServerTests method), 90
- test_valid_get_object_9_get_setting_by_name() $(test_pytan_valid_server_tests.ValidServerTests$ method), 90

- test_valid_question_10_ask_manual_question_sensor_with_filter() (test pytan valid server tests. ValidServerTests method), 90
- test_valid_question_11_ask_manual_question_multiple_sensors_identified_ (test_pytan_valid_server_tests.ValidServerTests method), 90
- test_valid_question_12_ask_manual_question_sensor_with_parameters_and (test_pytan_valid_server_tests.ValidServerTests method), 90
- test_valid_question_13_ask_manual_question_sensor_with_filter_and_3_o (test_pytan_valid_server_tests.ValidServerTests method), 90
- test_valid_question_14_ask_manual_question_complex_query2() (test_pytan_valid_server_tests.ValidServerTests method), 90
- $test_valid_get_object_26_get_saved_question_by_name() \\ test_valid_question_15_ask_manual_question_complex_query1() \\$ (test_pytan_valid_server_tests.ValidServerTests method), 90
 - test_valid_question_1_ask_manual_question_sensor_with_parameters_and (test pytan valid server tests. ValidServerTests method), 90
 - test_valid_question_2_ask_manual_question_multiple_sensors_with_paran $(test_pytan_valid_server_tests.ValidServerTests$ method), 90
 - test_valid_question_3_ask_manual_question_simple_multiple_sensors() (test pytan valid server tests. ValidServerTests method), 90
 - test_valid_question_4_ask_manual_question_sensor_without_parameters_a (test_pytan_valid_server_tests.ValidServerTests method), 90
 - test_valid_question_5_ask_manual_question_sensor_with_filter_and_2_op (test_pytan_valid_server_tests.ValidServerTests method), 91
 - test_valid_question_6_ask_manual_question_sensor_with_parameters_and (test pytan valid server tests. ValidServerTests method), 91
 - test_valid_question_7__ask_manual_question_sensor_complex() (test_pytan_valid_server_tests.ValidServerTests method), 91
 - test_valid_question_8_ask_manual_question_sensor_with_parameters_and (test pytan valid server tests. ValidServerTests method), 91
 - test_valid_question_9_ask_manual_question_simple_single_sensor() (test_pytan_valid_server_tests.ValidServerTests
 - test_valid_saved_question_1_ask_saved_question_refresh_data() (test_pytan_valid_server_tests.ValidServerTests method), 91
 - test_valid_saved_question_2_ask_saved_question_by_name() (test_pytan_valid_server_tests.ValidServerTests method), 91
 - test valid saved question 3 ask saved question by name in list() (test_pytan_valid_server_tests.ValidServerTests method), 91

test_valid_simple_list() (test_pytan_unit.TestDehumanizeS	
method), 84	to_jsonable() (taniumpy.object_types.base.BaseType
test_valid_simple_str_hash_selector()	method), 372
(test_pytan_unit.TestDehumanizeSensorUtils	to_jsonable() (taniumpy.object_types.result_set.ResultSet
method), 84	method), 376
test_valid_simple_str_id_selector()	toSOAPBody() (taniumpy.object_types.base.BaseType
(test_pytan_unit.TestDehumanizeSensorUtils	method), 372
method), 84	toSOAPElement() (tani-
test_valid_simple_str_name_selector()	umpy.object_types.base.BaseType method),
(test_pytan_unit.TestDehumanizeSensorUtils	372
method), 84	
test_version_higher() (test_pytan_unit.TestGenericUtils	U
method), 85	unpack() (in module ddt), 381
test_version_lower() (test_pytan_unit.TestGenericUtils	unparse() (in module xmltodict), 380
method), 85	UnsupportedVersionError, 35
TestDehumanizeExtractionUtils (class in	
· · · · · · · · · · · · · · · · · · ·	UploadFile (class in taniumpy.object_types.upload_file),
test_pytan_unit), 83	378
TestDehumanizeQuestionFilterUtils (class in	UploadFileList (class in tani-
test_pytan_unit), 83	umpy.object_types.upload_file_list), 378
TestDehumanizeQuestionOptionUtils (class in	UploadFileStatus (class in tani-
test_pytan_unit), 83	umpy.object_types.upload_file_status), 378
TestDehumanizeSensorUtils (class in test_pytan_unit), 84	User (class in taniumpy.object_types.user), 378
TestDeserializeBadXML (class in test_pytan_unit), 84	UserList (class in taniumpy.object_types.user_list), 378
TestGenericUtils (class in test_pytan_unit), 84	UserRole (class in taniumpy.object_types.user_role), 379
TestManualBuildObjectUtils (class in test_pytan_unit),	UserRoleList (class in tani-
85	umpy.object_types.user_role_list), 379
TestManualPackageDefValidateUtils (class in	M
test_pytan_unit), 85	V
$Test Manual Question Filter Def Parse Utils \qquad (class \qquad in$	val_package_def() (in module pytan.utils), 71
test_pytan_unit), 86	val_q_filter_defs() (in module pytan.utils), 72
TestManualQuestionFilterDefValidateUtils (class in	val_sensor_defs() (in module pytan.utils), 72
test_pytan_unit), 86	ValidServerTests (class in test_pytan_valid_server_tests),
TestManualQuestionOptionDefParseUtils (class in	87
test_pytan_unit), 86	version_check() (in module pytan.binsupport), 80
TestManualSensorDefParseUtils (class in	VersionAggregate (class in tani-
test_pytan_unit), 86	umpy.object_types.version_aggregate), 379
TestManualSensorDefValidateUtils (class in	VersionAggregateList (class in tani-
test_pytan_unit), 87	umpy.object_types.version_aggregate_list),
threaded_http (module), 381	379
threaded_http() (in module threaded_http), 382	VersionMismatchError, 35
ThreadedHTTPServer (class in threaded_http), 382	VersionParseError, 35
TIME_FORMAT (in module pytan.constants), 60	versioni discilitor, 33
TIMEOUT_SECS_DEFAULT (pytan.pollers.SSEPoller	W
attribute), 57	WhiteListedUrl (class in tani-
TimeoutException, 35	`
timestr_to_datetime() (in module pytan.utils), 71	umpy.object_types.white_listed_url), 379
to_flat_dict() (taniumpy.object_types.base.BaseType	WhiteListedUrlList (class in tani-
method), 372	umpy.object_types.white_listed_url_list),
to_flat_dict_explode_json() (tani-	379
umpy.object_types.base.BaseType method),	write_csv() (taniumpy.object_types.base.BaseType static
372	method), 372
to_json() (taniumpy.object_types.base.BaseType static	write_csv() (taniumpy.object_types.result_set.ResultSet
method), 372	static method), 376
to_json() (taniumpy.object_types.result_set.ResultSet	
to_json() (tallfallpy.coject_types.resuit_set.itesuitset	

Χ

```
XML_1_0_RESTRICTED_HEX (in module pytan.xml_clean), 81

XML_1_0_VALID_HEX (in module pytan.xml_clean), 81

xml_cleaner() (in module pytan.xml_clean), 82

xml_pretty() (in module pytan.utils), 72

xml_pretty_resultobj() (in module pytan.utils), 72

xml_pretty_resultxml() (in module pytan.utils), 72

xml_to_result_set_obj() (pytan.handler.Handler method), 34

XmlError (class in taniumpy.object_types.xml_error), 379

XMLNS (pytan.sessions.Session attribute), 37

xmltodict (module), 379
```