

---

# PyTan Documentation

*Release 2.1.0*

**Jim Olsen**

September 03, 2015



## CONTENTS

<b>1</b>	<b>Table of Contents</b>	<b>1</b>
1.1	PyTan Introduction . . . . .	1
1.2	pytan package . . . . .	3
1.3	taniumpy package . . . . .	579
1.4	xmldict module . . . . .	588
1.5	ddt module . . . . .	589
1.6	threaded_http module . . . . .	590
1.7	requests package . . . . .	590
<b>2</b>	<b>Indices and tables</b>	<b>593</b>
	<b>Python Module Index</b>	<b>595</b>
	<b>Index</b>	<b>597</b>



## TABLE OF CONTENTS

### 1.1 PyTan Introduction

#### 1.1.1 Description

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

- *Tanium Server SOAP API*: The SOAP server embedded into the Tanium server itself. For Tanium version 6.2: The SOAP servers listens on port 444 but is also available via port 443. For Tanium version 6.5: The SOAP servers listens on port 443, and is not available on port 444
- *TaniumPy Python Package*: ([taniumpy](#)) A python package comprised of a set of python objects automatically generated from the WSDL file that describes the Tanium SOAP API. These python objects handle the serialization and deserialization of XML to and from the Tanium Server SOAP API. Located in `lib/taniumpy`
- *PyTan Python Package*: ([pytan](#)) A python package that provides a set of methods to make interfacing with TaniumPy more human friendly. Located in `lib/pytan`
- *PyTan Command Line Scripts*: A set of command line scripts that utilize the PyTan Package ([pytan](#)) to make it easy for non-programmers to create/get/delete/ask/deploy objects via the Tanium Server SOAP API.

#### 1.1.2 Why it was created

This was created to solve for the following needs:

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

#### 1.1.3 Requirements

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

### 1.1.4 Tanium Versions Validated Against

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

- 6.2.314.3315
- 6.2.314.3321
- 6.5.314.4254
- 6.5.314.4268
- 6.5.314.4275

### 1.1.5 Installation

#### Windows Installation

- Download Python 2.7.9 from <https://www.python.org/downloads/windows/>
- Install Python 2.7.9 – if you accept the default paths it will install to C:\Python27
- Copy PyTan from github to your local machine somewhere
- If you did not accept the default install path for Python 2.7, edit `pytan\winbin\CONFIG.bat` to change the `PYTHON-` variable to point to the full path of `*python.exe`

#### OS X Installation

- OS X 10.8 and higher come with Python 2.7.6 out of the box
- Copy PyTan from github to your local machine somewhere

#### Linux Installation

- Ensure Python 2.7.9 is installed
- Ensure the first python binary in your path points to your Python 2.7 installation
- Copy PyTan from github to your local machine somewhere

### 1.1.6 Usage

- For command line usage, refer to Command Line Help Index
- For API Examples, refer to the [PyTan API Examples](#)
- For in depth API Documentation, refer to the [pytan package](#), especially the `pytan.handler` module

### 1.1.7 Directory Layout

- `EXAMPLES/` directory: contains a set of example python files that show how to use the various methods exposed by (`pytan`)
- `BUILD/` directory: contains the scripts that build the HTML and PDF documentation in `doc/`, generate the (`taniumpy`), generate the python examples in `EXAMPLES/`, generate some of the command line scripts in `bin/`, and generate all of the documentation for the command line scripts in `doc/_static/bin_doc`
- `bin/` directory: contains all of the command line scripts that utilize the (`pytan`)
- `doc/` directory: contains the HTML and PDF documentation

- *lib/ directory*: contains the python libraries (*pytan*) and (*taniumpy*), as well as other python libraries
- *test/ directory*: contains the unit and functional tests for (*pytan*)
- *winbin/ directory*: contains the Windows batch scripts which wrap around the python command line scripts in *bin/*
- *ZIP\_DIST/ directory*: contains standalone windows executables for certain tools, created by batch files in *BUILD/STATICWINBUILD/*

### 1.1.8 Other References

- [Tanium Platform Website](#)
- [Tanium Knowledge Base](#)
- [Tanium SOAP Knowledge Base Article](#)
- The console.wsdl used to build the taniumpy library for this version, also useful as a reference tool.

## 1.2 pytan package

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

```
pytan.__version__ = '2.1.0'
    Version of PyTan
```

```
pytan.__copyright__ = 'Copyright 2015 Tanium'
    Copyright for PyTan
```

```
pytan.__license__ = 'MIT'
    License for PyTan
```

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

### 1.2.1 pytan.handler module

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

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

Bases: *object*

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

**Parameters***username* : str

- default: None

- *username* to connect to *host* with

**password** : str

- default: None

- *password* to connect to *host* with

**host** : str

- default: None

- hostname or ip of Tanium SOAP Server

**port** : int, optional

- default: 443
- port of Tanium SOAP Server on *host*

**loglevel** : int, optional

- default: 0
- 0 do not print anything except warnings/errors
- 1 and higher will print more

**debugformat** : bool, optional

- default: False
- False: use one line logformat
- True: use two lines

**gmt\_log** : bool, optional

- default: True
- True: use GMT timezone for log output
- False: use local time for log output

**session\_id** : str, optional

- default: None
- session\_id to use while authenticating instead of username/password

**Other Parameters**  
**http\_debug** : bool, optional

- default: False
- False: do not print requests package debug
- True: do print requests package debug
- This is passed through to `pytan.sessions.Session`

**http\_auth\_retry**: bool, optional

- default: True
- True: retry HTTP GET/POST's
- False: do not retry HTTP GET/POST's
- This is passed through to `pytan.sessions.Session`

**http\_retry\_count**: int, optional

- default: 5
- number of times to retry HTTP GET/POST's if the connection times out/fails
- This is passed through to `pytan.sessions.Session`

**soap\_request\_headers** : dict, optional

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



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

**auth\_connect\_timeout\_sec** : int, optional

- default: 5
- number of seconds before timing out for a connection while authenticating
- This is passed through to `pytan.sessions.Session`

**auth\_response\_timeout\_sec** : int, optional

- default: 15
- number of seconds before timing out for a response while authenticating
- This is passed through to `pytan.sessions.Session`

**info\_connect\_timeout\_sec** : int, optional

- default: 5
- number of seconds before timing out for a connection while getting /info.json
- This is passed through to `pytan.sessions.Session`

**info\_response\_timeout\_sec** : int, optional

- default: 15
- number of seconds before timing out for a response while getting /info.json
- This is passed through to `pytan.sessions.Session`

**soap\_connect\_timeout\_sec** : int, optional

- default: 15
- number of seconds before timing out for a connection for a SOAP request
- This is passed through to `pytan.sessions.Session`

**soap\_response\_timeout\_sec** : int, optional

- default: 540
- number of seconds before timing out for a response for a SOAP request
- This is passed through to `pytan.sessions.Session`

**stats\_loop\_enabled** : bool, optional

- default: False
- False: do not enable the statistics loop thread
- True: enable the statistics loop thread
- This is passed through to `pytan.sessions.Session`

**stats\_loop\_sleep\_sec** : int, optional

- default: 5
- number of seconds to sleep in between printing the statistics when stats\_loop\_enabled is True
- This is passed through to `pytan.sessions.Session`

**record\_all\_requests**: bool, optional

- default: False
- False: do not add each requests response object to session.ALL\_REQUESTS\_RESPONSES
- True: add each requests response object to session.ALL\_REQUESTS\_RESPONSES
- This is passed through to `pytan.sessions.Session`

**stats\_loop\_targets** : list of dict, optional

- default: [{ 'Version': 'Settings/Version' }, { 'Active Questions': 'Active Question Cache/Active Question Estimate' }, { 'Clients': 'Active Question Cache/Active Client Estimate' }, { 'Strings': 'String Cache/Total String Count' }, { 'Handles': 'System Performance Info/HandleCount' }, { 'Processes': 'System Performance Info/ProcessCount' }, { 'Memory Available': 'percentage(System Performance Info/PhysicalAvailable,System Performance Info/PhysicalTotal)' }]
- list of dictionaries with the key being the section of info.json to print info from, and the value being the item with in that section to print the value
- This is passed through to `pytan.sessions.Session`

**persistent**: bool, optional

- default: False
- False: do not request a persistent session
- True: do request a persistent
- This is passed through to `pytan.sessions.Session.authenticate()`

See also:

`pytan.constants.LOG_LEVEL_MAPS` maps a given *loglevel* to respective logger names and their logger levels

`pytan.constants.INFO_FORMAT` debugformat=False

`pytan.constants.DEBUG_FORMAT` debugformat=True

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

## Notes

- for 6.2: port 444 is the default SOAP port, port 443 forwards /soap/ URLs to the SOAP port, Use port 444 if you have direct access to it. However, port 444 is the only port that exposes the /info page in 6.2
- for 6.5: port 443 is the default SOAP port, there is no port 444

## 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: False
- True: perform a server side export when getting result data
- False: perform a normal get result data (default for 6.2)
- Keeping False by default for now until the columnset's are properly identified in the server export

**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()`

**Returns**ret : dict, containing:

- question\_object** : `taniumpy.object_types.question.Question`, the actual question created and added by PyTan
- question\_results** : `taniumpy.object_types.result_set.ResultSet`, the Result Set for **question\_object** if `get_results == True`
- poller\_object** : `pytan.pollers.QuestionPoller`, poller object used to wait until all results are in before getting **question\_results**
- poller\_success** : None if `get_results == True`, otherwise True or False

See also:

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

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

### Examples

```
>>> # example of str for sensor_defs
>>> sensor_defs = 'Sensor1'
```

```
>>> # example of dict for sensor_defs
>>> sensor_defs = {
...     'name': 'Sensor1',
...     'filter': {
...         'operator': 'RegexMatch',
...         'not_flag': 0,
...         'value': '.*'
...     },
...     'params': {'key': 'value'},
```

```
...     'options': {'and_flag': 1}
... }
```

```
>>> # example of dict for question_filter_defs
>>> question_filter_defs = {
...     'operator': 'RegexMatch',
...     'not_flag': 0,
...     'value': '.*'
... }
```

**`_check_sse_crash_prevention`** (*obj*, *\*\*kwargs*)

Runs a number of methods used to prevent crashing the platform server when performing server side exports

**Parameters***obj* : *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

**Parameters***obj* : *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

**Parameters***sse\_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`

**Parameters***ok\_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.

**Parameters***package\_def* : dict

- definition that describes a package

**action\_filter\_defs** : str, dict, list of str or dict, optional

- default: []

- action filter definitions

**action\_option\_defs** : dict, list of dict, optional

- default: []

- action filter option definitions

**start\_seconds\_from\_now** : int, optional

- default: 0

- start action N seconds from now

**expire\_seconds** : int, optional

- default: package.expire\_seconds

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

**run** : bool, optional

- default: False

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

- True: actually deploy the action

**get\_results** : bool, optional

- default: True

- True: wait for result completion after deploying action

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

**action\_name** : str, optional

- default: prepend package name with “API Deploy “

- custom name for action

**action\_comment** : str, optional

- default:

- custom comment for action

**Returns****ret** : dict, containing:

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

- action\_object* : `taniumpy.object_types.action.Action`, the action object that *tanium* created for *saved\_action*

- package\_object* : `taniumpy.object_types.package_spec.PackageSpec`, the package object used in *saved\_action*

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

- poller\_object* : `pytan.pollers.ActionPoller`, poller object used to wait until all results are in before getting *action\_results*

- poller\_success* : None if *get\_results* == False, otherwise True or False

- action\_results* : None if *get\_results* == False, otherwise *taniumpy.object\_types.result\_set.ResultSet*, the results for *action\_object*
- action\_result\_map* : None if *get\_results* == False, otherwise progress map for *action\_object* in dictionary form

See also:

*pytan.constants.FILTER\_MAPS* valid filter dictionaries for filters

*pytan.constants.OPTION\_MAPS* valid option dictionaries for options

## Notes

### •For 6.2:

- We need to add an Action object
- The Action object should not be in an ActionList
- Action.start\_time must be specified, if it is not specified the action shows up as expired immediately. We default to 1 second from current time if start\_seconds\_from\_now is not passed in

### •For 6.5 / 6.6:

- We need to add a SavedAction object, the server creates the Action object for us
- To emulate what the console does, the SavedAction should be in a SavedActionList
- Action.start\_time does not need to be specified

## Examples

```
>>> # example of dict for `package_def`
>>> package_def = {'name': 'PackageName1', 'params':{'param1': 'value1'}}
```

```
>>> # example of str for `action_filter_defs`
>>> action_filter_defs = 'Sensor1'
```

```
>>> # example of dict for `action_filter_defs`
>>> action_filter_defs = {
...     'name': 'Sensor1',
...     'filter': {
...         'operator': 'RegexMatch',
...         'not_flag': 0,
...         'value': '.*'
...     },
...     'options': {'and_flag': 1}
... }
```

**`_export_class_BaseType`** (*obj*, *export\_format*, **`**kwargs`**)

Handles exporting *taniumpy.object\_types.base.BaseType*

**Parameters***obj* : *taniumpy.object\_types.base.BaseType*

- taniumpy* object to export

**export\_format** : str

- str of format to perform export in

**Returns**result : str

- results of exporting *obj* into format *export\_format*

**\_export\_class\_ResultSet** (*obj*, *export\_format*, *\*\*kwargs*)

Handles exporting `taniumpy.object_types.result_set.ResultSet`

**Parameters**obj : `taniumpy.object_types.result_set.ResultSet`

- taniumpy object to export

**export\_format** : str

- str of format to perform export in

**Returns**result : str

- results of exporting *obj* into format *export\_format*

**\_export\_format\_csv** (*obj*, *\*\*kwargs*)

Handles exporting format: CSV

**Parameters**obj : `taniumpy.object_types.result_set.ResultSet` or  
`taniumpy.object_types.base.BaseType`

- taniumpy object to export

**Returns**result : str

- results of exporting *obj* into csv format

**\_export\_format\_json** (*obj*, *\*\*kwargs*)

Handles exporting format: JSON

**Parameters**obj : `taniumpy.object_types.result_set.ResultSet` or  
`taniumpy.object_types.base.BaseType`

- taniumpy object to export

**Returns**result : str

- results of exporting *obj* into json format

**\_export\_format\_xml** (*obj*, *\*\*kwargs*)

Handles exporting format: XML

**Parameters**obj : `taniumpy.object_types.result_set.ResultSet` or  
`taniumpy.object_types.base.BaseType`

- taniumpy object to export

**Returns**result : str

- results of exporting *obj* into XML format

**\_find** (*obj*, *\*\*kwargs*)

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

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

- object to find

**Returns**found : `taniumpy.object_types.base.BaseType`

- full object that was found



**`_get_multi`** (*obj\_map*, *\*\*kwargs*)  
Find multiple item wrapper using `_find()`

**Parameters***obj\_map* : dict

- dict containing the map for a given object type

**Returns***found* : `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

**Parameters***d* : dict

- dict containing package definition

**Returns***d* : 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

**Parameters***defs* : list of dict

- list of dicts containing sensor definitions

**Returns***defs* : 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()`

**Parameters***obj\_map* : dict

- dict containing the map for a given object type

**Returns***found* : `taniumpy.object_types.base.BaseType`

- full object that was found

**`_resolve_sse_format`** (*sse\_format*, *\*\*kwargs*)  
Resolves the server side export format the user supplied to an integer for the API

**Parameters***sse\_format* : str or int

- user supplied export format

**Returns***sse\_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()`

**Parameters***obj\_map* : dict

- dict containing the map for a given object type

**k** : str

- attribute name to set to *v*

**v** : str

- attribute value to set on *k*

**Returns**`found` : `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

**Parameters**`v_maps` : list of dict

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

**Returns**`bool`

- True if all values in all `v_maps` are greater than or equal to all values in `self.session.server_version_dict`
- False otherwise

**`approve_saved_action`** (`id`, `**kwargs`)

Approve a saved action

**Parameters**`id` : int

- id of saved action to approve

**Returns**`saved_action_approve_obj` : `taniumpy.object_types.saved_action_approval.SavedActionApproval`

- The object containing the return from `SavedActionApproval`

**`ask`** (`**kwargs`)

Ask a type of question and get the results back

**Parameters**`qtype` : str, optional

- default: 'manual'
- type of question to ask: {'saved', 'manual', '\_manual'}

**Returns**`result` : dict, containing:

- `question_object` : one of the following depending on `qtype`:  
`taniumpy.object_types.question.Question` or  
`taniumpy.object_types.saved_question.SavedQuestion`
- `question_results` : `taniumpy.object_types.result_set.ResultSet`

**See also:**

`pytan.constants.Q_OBJ_MAP` maps `qtype` to a method in `Handler()`

`pytan.handler.Handler.ask_saved()` method used when `qtype == 'saved'`

`pytan.handler.Handler.ask_manual()` method used when `qtype == 'manual'`

`pytan.handler.Handler._ask_manual()` method used when `qtype == '_manual'`

**`ask_manual`** (`**kwargs`)

Ask a manual question using human strings and get the results back

This method takes a string or list of strings and parses them into their corresponding definitions needed by

`_ask_manual()`

**Parameters**`sensors` : str, list of str

- default: []

- sensors (columns) to include in question

**question\_filters** : str, list of str, optional

- default: []
- filters that apply to the whole question

**question\_options** : str, list of str, optional

- default: []
- options that apply to the whole question

**get\_results** : bool, optional

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

**sensors\_help** : bool, optional

- default: False
- False: do not print the help string for sensors
- True: print the help string for sensors and exit

**filters\_help** : bool, optional

- default: False
- False: do not print the help string for filters
- True: print the help string for filters and exit

**options\_help** : bool, optional

- default: False
- False: do not print the help string for options
- True: print the help string for options and exit

**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()`

**Returns**`result` : dict, containing:

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

**See also:**

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

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

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

## Notes

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

Get Computer Name and IP Route Details from all machines with Is Windows containing “True”

Asking the same question in PyTan has some similarities:

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

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

- sensors
- left hand side
- column selects

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

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

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

And in PyTan:

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

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*

**Parameters**  
**question\_text** : str

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

**picker** : int

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

**get\_results** : bool, optional

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

**sse** : bool, optional

- default: False
- True: perform a server side export when getting result data
- False: perform a normal get result data (default for 6.2)
- Keeping False by default for now until the columnset's are properly identified in the server export

**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()*

**Returns**ret : dict, containing:

- question\_object* : *taniumpy.object\_types.question.Question*, the actual question added by PyTan
- question\_results* : *taniumpy.object\_types.result\_set.ResultSet*, the Result Set for *question\_object* if *get\_results* == True

- *poller\_object* : `pytan.pollers.QuestionPoller`, poller object used to wait until all results are in before getting *question\_results*
- *poller\_success* : None if *get\_results* == True, otherwise 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 level)

```
>>> 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

**Parameters***sid* : 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: False
- True: perform a server side export when getting result data
- False: perform a normal get result data (default for 6.2)
- Keeping False by default for now until the columnset’s are properly identified in the server export

**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

**Returns***ret* : dict, containing

- question\_object*: *taniumpy.object\_types.saved\_question.SavedQuestion*, the saved question object
- question\_object*: *taniumpy.object\_types.question.Question*, the question asked by *saved\_question\_object*
- question\_results*: *taniumpy.object\_types.result\_set.ResultSet*, the results for *question\_object*
- poller\_object* : None if *refresh\_data* == False, otherwise *pytan.pollers.QuestionPoller*, poller object used to wait until all results are in before getting *question\_results*,
- poller\_success* : None if *refresh\_data* == False, otherwise True or False

## Notes

id or name must be supplied

**create\_dashboard** (*name*, *text*='', *group*='', *public\_flag*=True, *\*\*kwargs*)

Calls *pytan.handler.Handler.run\_plugin()* to run the CreateDashboard plugin and parse the response

**Parameters***name* : str

- name of dashboard to create

**text** : str, optional

- default: ''
- text for this dashboard

**group** : str, optional

- default: ''
- group name for this dashboard

**public\_flag** : bool, optional

- default: True
- True: make this dashboard public
- False: do not make this dashboard public

**Returns***plugin\_result*, *sql\_zipped* : tuple

- plugin\_result* will be the *taniumpy* object representation of the SOAP response from Tanium server
- sql\_zipped* will be a dict with the SQL results embedded in the SOAP response

**create\_from\_json** (*objtype*, *json\_file*, *\*\*kwargs*)

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

**Parameters***objtype* : str

- Type of object described in *json\_file*

**json\_file** : str

- path to JSON file that describes an API object

**Returns***ret* : *taniumpy.object\_types.base.BaseType*



- TaniumPy object added to Tanium SOAP Server

See also:

*pytan.constants.GET\_OBJ\_MAP* maps objtype to supported 'create\_json' types

**create\_group** (*groupname*, *filters*=[], *filter\_options*=[], **\*\*kwargs**)

Create a group object

**Parameters***groupname* : str

- name of group to create

**filters** : str or list of str, optional

- default: []
- each string must describe a filter

**filter\_options** : str or list of str, optional

- default: []
- each string must describe an option for *filters*

**filters\_help** : bool, optional

- default: False
- False: do not print the help string for filters
- True: print the help string for filters and exit

**options\_help** : bool, optional

- default: False
- False: do not print the help string for options
- True: print the help string for options and exit

**Returns***group\_obj* : *taniumpy.object\_types.group.Group*

- TaniumPy object added to Tanium SOAP Server

See also:

*pytan.constants.FILTER\_MAPS* valid filters for filters

*pytan.constants.OPTION\_MAPS* valid options for filter\_options

**create\_package** (*name*, *command*, *display\_name*='', *file\_urls*=[], *command\_timeout\_seconds*=600, *expire\_seconds*=600, *parameters\_json\_file*='', *verify\_filters*=[], *verify\_filter\_options*=[], *verify\_expire\_seconds*=600, **\*\*kwargs**)

Create a package object

**Parameters***name* : str

- name of package to create

**command** : str

- command to execute

**display\_name** : str, optional

- display name of package

**file\_urls** : list of strings, optional

- default: []
- URL of file to add to package
- can optionally define download\_seconds by using SECONDS::URL
- can optionally define file name by using FILENAME||URL
- can combine optionals by using SECONDS::FILENAME||URL
- FILENAME will be extracted from basename of URL if not provided

**command\_timeout\_seconds** : int, optional

- default: 600
- timeout for command execution in seconds

**parameters\_json\_file** : str, optional

- default: ''
- path to json file describing parameters for package

**expire\_seconds** : int, optional

- default: 600
- timeout for action expiry in seconds

**verify\_filters** : str or list of str, optional

- default: []
- each string must describe a filter to be used to verify the package

**verify\_filter\_options** : str or list of str, optional

- default: []
- each string must describe an option for *verify\_filters*

**verify\_expire\_seconds** : int, optional

- default: 600
- timeout for verify action expiry in seconds

**filters\_help** : bool, optional

- default: False
- False: do not print the help string for filters
- True: print the help string for filters and exit

**options\_help** : bool, optional

- default: False
- False: do not print the help string for options
- True: print the help string for options and exit

**metadata**: list of list of strs, optional

- default: []
- each list must be a 2 item list:

- list item 1 property name
- list item 2 property value

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

- TaniumPy object added to Tanium SOAP Server

See also:

`pytan.constants.FILTER_MAPS` valid filters for `verify_filters`

`pytan.constants.OPTION_MAPS` valid options for `verify_filter_options`

**create\_report\_file** (`contents`, `report_file=None`, `**kwargs`)

Exports a python API object to a file

**Parameters**`contents` : 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`

**Returns**`report_path` : str

- the full path to the file created with `contents`

**create\_sensor** (`**kwargs`)

Create a sensor object

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

**Warning:** Not currently supported, too complicated to add. Use `create_from_json()` instead for this object type!

**create\_user** (`name`, `rolename=[]`, `roleid=[]`, `properties=[]`, `**kwargs`)

Create a user object

**Parameters**`name` : 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 str, optional

- default: []
- each list must be a 2 item list:
- list item 1 property name
- list item 2 property value

**Returns****user\_obj** : *taniumpy.object\_types.user.User*

- TaniumPy object added to Tanium SOAP Server

**create\_whitelisted\_url** (*url*, *regex=False*, *download\_seconds=86400*, *properties=[]*,  
*\*\*kwargs*)

Create a whitelisted url object

**Parameters****url** : str

- text of new url

**regex** : bool, optional

- default: False
- False: *url* is not a regex pattern
- True: *url* is a regex pattern

**download\_seconds** : int, optional

- default: 86400
- how often to re-download *url*

**properties**: list of list of str, optional

- default: []
- each list must be a 2 item list:
- list item 1 property name
- list item 2 property value

**Returns****url\_obj** : *taniumpy.object\_types.white\_listed\_url.WhiteListedUrl*

- TaniumPy object added to Tanium SOAP Server

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

Delete an object type

**Parameters****objtype** : string

- type of object to delete

**id/name/hash** : int or string, list of int or string

- search attributes of object to delete, must supply at least one valid search attr

**Returns****ret** : dict

- dict containing deploy action object and results from deploy action

See also:

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

**delete\_dashboard** (*name*, *\*\*kwargs*)

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

**Parameters***name* : str

- name of dashboard to delete

**Returns***plugin\_result, sql\_zipped* : tuple

- plugin\_result* will be the taniumpy object representation of the SOAP response from Tanium server
- sql\_zipped* will be a dict with the SQL results embedded in the SOAP response

**deploy\_action** (*\*\*kwargs*)

Deploy an action and get the results back

This method takes a string or list of strings and parses them into their corresponding definitions needed by `_deploy_action()`

**Parameters***package* : str

- package to deploy with this action

**action\_filters** : str, list of str, optional

- default: []
- each string must describe a sensor and a filter which limits which computers the action will deploy *package* to

**action\_options** : str, list of str, optional

- default: []
- options to apply to *action\_filters*

**start\_seconds\_from\_now** : int, optional

- default: 0
- start action N seconds from now

**expire\_seconds** : int, optional

- default: package.expire\_seconds
- expire action N seconds from now, will be derived from package if not supplied

**run** : bool, optional

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

**get\_results** : bool, optional

- default: True
- True: wait for result completion after deploying action

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

**package\_help** : bool, optional

- default: False
- False: do not print the help string for package
- True: print the help string for package and exit

**filters\_help** : bool, optional

- default: False
- False: do not print the help string for filters
- True: print the help string for filters and exit

**options\_help** : bool, optional

- default: False
- False: do not print the help string for options
- True: print the help string for options and exit

**Returns****ret** : dict, containing:

- saved\_action\_object* : `taniumpy.object_types.saved_action.SavedAction`, the *saved\_action* added for this action (None if 6.2)
- action\_object* : `taniumpy.object_types.action.Action`, the action object that *tanium* created for *saved\_action*
- package\_object* : `taniumpy.object_types.package_spec.PackageSpec`, the package object used in *saved\_action*
- action\_info* : `taniumpy.object_types.result_info.ResultInfo`, the initial `GetResultInfo` call done before getting results
- poller\_object* : `pytan.pollers.ActionPoller`, poller object used to wait until all results are in before getting *action\_results*
- poller\_success* : None if *get\_results* == False, otherwise True or False
- action\_results* : None if *get\_results* == False, otherwise `taniumpy.object_types.result_set.ResultSet`, the results for *action\_object*
- action\_result\_map* : None if *get\_results* == False, otherwise progress map for *action\_object* in dictionary form

**See also:**

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

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

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

### Examples

```
>>> # example of str for `package`
>>> package = 'Package1'
```

```
>>> # example of str for `package` with params
>>> package = 'Package1{key:value}'
```

```
>>> # example of str for `action_filters` with params and filter for sensors
>>> action_filters = 'Sensor1{key:value}, that contains:example text'
```

```
>>> # example of list of str for `action_options`
>>> action_options = ['max_data_age:3600', 'and']
```

**export\_obj** (*obj*, *export\_format*='csv', *\*\*kwargs*)

Exports a python API object to a given export format

**Parameters***obj* : *taniumpy.object\_types.base.BaseType* or *taniumpy.object\_types.result\_set.ResultSet*

- TaniumPy object to export

**export\_format** : str, optional

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

**header\_sort** : list of str, bool, optional

- default: True
- for *export\_format* csv and *obj* types *taniumpy.object\_types.base.BaseType* or *taniumpy.object\_types.result\_set.ResultSet*
- True: sort the headers automatically
- False: do not sort the headers at all
- list of str: sort the headers returned by priority based on provided list

**header\_add\_sensor** : bool, optional

- default: False
- for *export\_format* csv and *obj* type *taniumpy.object\_types.result\_set.ResultSet*
- False: do not prefix the headers with the associated sensor name for each column
- True: prefix the headers with the associated sensor name for each column

**header\_add\_type** : bool, optional

- default: False
- for *export\_format* csv and *obj* type *taniumpy.object\_types.result\_set.ResultSet*
- False: do not postfix the headers with the result type for each column
- True: postfix the headers with the result type for each column

**expand\_grouped\_columns** : bool, optional

- default: False
- for *export\_format* csv and *obj* type *taniumpy.object\_types.result\_set.ResultSet*
- False: do not expand multiline row entries into their own rows
- True: expand multiline row entries into their own rows

**explode\_json\_string\_values** : bool, optional

- default: False
- for *export\_format* json or csv and *obj* type `taniumpy.object_types.base.BaseType`
- False: do not explode JSON strings in object attributes into their own object attributes
- True: explode JSON strings in object attributes into their own object attributes

**minimal** : bool, optional

- default: False
- for *export\_format* xml and *obj* type `taniumpy.object_types.base.BaseType`
- False: include empty attributes in XML output
- True: do not include empty attributes in XML output

**Returns**result : str

- the contents of exporting *export\_format*

**See also:**

`pytan.constants.EXPORT_MAPS` maps the type *obj* to *export\_format* and the optional args supported for each

## Notes

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

**export\_to\_report\_file** (*obj*, *export\_format*='csv', **\*\*kwargs**)  
Exports a python API object to a file

**Parameters***obj* : `taniumpy.object_types.base.BaseType` or  
`taniumpy.object_types.result_set.ResultSet`

- TaniumPy object to export

**export\_format** : str, optional

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

**header\_sort** : list of str, bool, optional

- default: True
- for *export\_format* csv and *obj* types `taniumpy.object_types.base.BaseType` or `taniumpy.object_types.result_set.ResultSet`
- True: sort the headers automatically
- False: do not sort the headers at all
- list of str: sort the headers returned by priority based on provided list

**header\_add\_sensor** : bool, optional

- default: False
- for *export\_format* csv and *obj* type `taniumpy.object_types.result_set.ResultSet`



- False: do not prefix the headers with the associated sensor name for each column
- True: prefix the headers with the associated sensor name for each column

**header\_add\_type** : bool, optional

- default: False
- for *export\_format* csv and *obj* type `taniumpy.object_types.result_set.ResultSet`
- False: do not postfix the headers with the result type for each column
- True: postfix the headers with the result type for each column

**expand\_grouped\_columns** : bool, optional

- default: False
- for *export\_format* csv and *obj* type `taniumpy.object_types.result_set.ResultSet`
- False: do not expand multiline row entries into their own rows
- True: expand multiline row entries into their own rows

**explode\_json\_string\_values** : bool, optional

- default: False
- for *export\_format* json or csv and *obj* type `taniumpy.object_types.base.BaseType`
- False: do not explode JSON strings in object attributes into their own object attributes
- True: explode JSON strings in object attributes into their own object attributes

**minimal** : bool, optional

- default: False
- for *export\_format* xml and *obj* type `taniumpy.object_types.base.BaseType`
- False: include empty attributes in XML output
- True: do not include empty attributes in XML output

**report\_file**: str, optional

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

**report\_dir**: str, optional

- default: None
- directory to save report in, will use current working directory if not supplied

**prefix**: str, optional

- default: ''
- prefix to add to *report\_file*

**postfix**: str, optional

- default: ''
- postfix to add to *report\_file*

**Returns**`report_path, result` : tuple

- report\_path* : str, the full path to the file created with contents of *result*

- result : str, the contents written to report\_path

See also:

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

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

## Notes

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

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

**Parameters***objtype* : string

- type of object to get
- id/name/hash** : int or string, list of int or string
- search attributes of object to get, must supply at least one valid search attr

**Returns***obj\_list* : `taniumpy.object_types.base.BaseType`

- The object list of items found for *objtype*

See also:

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

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

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

**get\_all** (*objtype*, *\*\*kwargs*)  
Get all objects of a type

**Parameters***objtype* : string

- type of object to get

**Returns***obj\_list* : `taniumpy.object_types.base.BaseType`

- The object list of items found for *objtype*

See also:

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

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

**get\_dashboards** (*name*=' ', *\*\*kwargs*)

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

**Parameters***name* : str, optional

- default: ' '

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

**Returns** `plugin_result, sql_zipped` : tuple

- `plugin_result` will be the `taniumpy` object representation of the SOAP response from Tanium server
- `sql_zipped` will be a dict with the SQL results embedded in the SOAP response

**get\_result\_data** (*obj*, *aggregate=False*, *shrink=True*, *\*\*kwargs*)

Get the result data for a python API object

This method issues a `GetResultData` command to the SOAP api for *obj*. `GetResultData` returns the columns and rows that are currently available for *obj*.

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

- object to get result data for

**aggregate** : bool, optional

- default: False
- False: get all the data
- True: get just the aggregate data (row counts of matches)

**shrink** : bool, optional

- default: True
- True: Shrink the object down to just id/name/hash attributes (for smaller request)
- False: Use the full object as is

**Returns** *rd* : `taniumpy.object_types.result_set.ResultSet`

The return of `GetResultData` for *obj*

**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

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

- object to get result data for

**sse\_format** : str, optional

- default: 'csv'

- format to have server create report in, one of: {'csv', 'xml', 'xml\_obj', 'cef', 0, 1, 2}

**leading** : str, optional

- default: ''

- used for sse\_format 'cef' only, the string to prepend to each row

**trailing** : str, optional

- default: ''

- used for sse\_format 'cef' only, the string to append to each row

**Returnsexport\_data** : either *str* or *taniumpy.object\_types.result\_set.ResultSet*

- If sse\_format is one of csv, xml, or cef, export\_data will be a *str* containing the contents of the ResultSet in said format

- If sse\_format is xml\_obj, export\_data will be a *taniumpy.object\_types.result\_set.ResultSet*

**See also:**

*pytan.constants.SSE\_FORMAT\_MAP* maps sse\_format to an integer for use by the SOAP API

*pytan.constants.SSE\_RESTRICT\_MAP* maps sse\_format integers to supported platform versions

*pytan.constants.SSE\_CRASH\_MAP* maps platform versions that can cause issues in various scenarios

**get\_result\_info** (*obj*, *shrink=True*, *\*\*kwargs*)

Get the result info for a python API object

This method issues a GetResultInfo command to the SOAP api for *obj*. GetResultInfo returns information about how many servers have passed the *obj*, total number of servers, and so on.

**Parametersobj** : *taniumpy.object\_types.base.BaseType*

- object to get result data for

**shrink** : bool, optional

- default: True

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

- False: Use the full object as is

**Returnsri** : *taniumpy.object\_types.result\_info.ResultInfo*

- The return of GetResultData for *obj*

**get\_server\_version** (*\*\*kwargs*)

Uses *taniumpy.session.Session.get\_server\_version()* to get the version of the Tanium Server

**Returnsserver\_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

**Returns**`parse_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

**Parameters**`obj`: `taniumpy.object_types.plugin.Plugin`

- Plugin object to run

**Returns**`plugin_result, sql_zipped`: tuple

- `plugin_result` will be the `taniumpy` object representation of the SOAP response from Tanium server
- `sql_zipped` will be a dict with the SQL results embedded in the SOAP response

**stop\_action** (*id*, *\*\*kwargs*)  
 Stop an action

**Parameters**`id`: int

- id of action to stop

**Returns**`action_stop_obj`: `taniumpy.object_types.action_stop.ActionStop`

The object containing the ID of the action stop job

**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

**Parameters**`xs`: str

- str of XML to convert to a `ResultSet` object

**Returns**`rs`: `taniumpy.object_types.result_set.ResultSet`

- `x` converted into a `ResultSet` object

## 1.2.2 pytan.exceptions module

Provides exceptions for the `pytan` module.

**exception** `pytan.exceptions.AuthorizationError`

Bases: `exceptions.Exception`

Exception thrown for authorization errors in `pytan.sessions`

**exception** `pytan.exceptions.BadResponseError`

Bases: `exceptions.Exception`

Exception thrown for BadResponse messages from Tanium in `pytan.sessions`

**exception** `pytan.exceptions.DefinitionParserError`

Bases: `exceptions.Exception`

Exception thrown for errors while parsing definitions from `pytan.handler`

**exception** `pytan.exceptions.HandlerError`

Bases: `exceptions.Exception`

Exception thrown for errors in `pytan.handler`

**exception** `pytan.exceptions.HttpError`

Bases: `exceptions.Exception`

Exception thrown for HTTP errors in `pytan.sessions`

**exception** `pytan.exceptions.HumanParserError`

Bases: `exceptions.Exception`

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

**exception** `pytan.exceptions.NotFoundError`

Bases: `exceptions.Exception`

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

**exception** `pytan.exceptions.PickerError`

Bases: `exceptions.Exception`

Exception thrown for picker errors in `pytan.handler`

**exception** `pytan.exceptions.PollingError`

Bases: `exceptions.Exception`

Exception thrown for errors in `pytan.polling`

**exception** `pytan.exceptions.PytanHelp`

Bases: `exceptions.Exception`

Exception thrown when printing out help

**exception** `pytan.exceptions.RunFalse`

Bases: `exceptions.Exception`

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

**exception** `pytan.exceptions.ServerParseError`

Bases: `exceptions.Exception`

Exception thrown for server parsing errors in `pytan.handler`

**exception** `pytan.exceptions.ServerSideExportError`

Bases: `exceptions.Exception`

Exception thrown for server side export errors in `pytan.handler`

**exception** `pytan.exceptions.TimeoutException`

Bases: `exceptions.Exception`

Exception thrown for timeout errors in `pytan.polling`

**exception** `pytan.exceptions.UnsupportedVersionError`

Bases: `exceptions.Exception`

Exception thrown for version checks in `pytan.handler`

**exception** `pytan.exceptions.VersionMismatchError`

Bases: `exceptions.Exception`

Exception thrown for `version_check` in `pytan.utils`

**exception** `pytan.exceptions.VersionParseError`

Bases: `exceptions.Exception`

Exception thrown for server version parsing errors in `pytan.handler`

### 1.2.3 pytan.sessions module

Session classes for the *pytan* module.

**class** `pytan.sessions.Session` (*host*, *port*=443, *\*\*kwargs*)

Bases: `object`

This session object uses the *requests* package instead of the built in `httplib` library.

This provides support for keep alive, gzip, cookies, forwarding, and a host of other features automatically.

#### Examples

Setup a `Session()` object:

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

Authenticate with the `Session()` object:

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

**ALL\_REQUESTS\_RESPONSES = []**

This list will be updated with each requests response object that was received

**AUTH\_CONNECT\_TIMEOUT\_SEC = 5**

number of seconds before timing out for a connection while authenticating

**AUTH\_FAIL\_CODES = [401, 403]**

List of HTTP response codes that equate to authorization failures

**AUTH\_RES = 'auth'**

The URL to use for authentication requests

**AUTH\_RESPONSE\_TIMEOUT\_SEC = 15**

number of seconds before timing out for a response while authenticating

**BAD\_RESPONSE\_CMD\_PRUNES = ['\n', 'XML Parse Error: ', 'SOAPProcessing Exception: class ', 'ERROR: 400 Bad Re**

List of strings to remove from commands in responses that do not match the response in the request

**BAD\_SERVER\_VERSIONS = [None, '', 'Unable to determine', 'Not yet determined']**

List of server versions that are not valid

**ELEMENT\_RE\_TXT = '<{0}>(.\*?)</{0}>'**

regex string to search for an element in XML bodies

**HTTP\_AUTH\_RETRY = True**

retry HTTP GET/POST's with username/password if session\_id fails or not

**HTTP\_DEBUG = False**

print requests package debug or not

**HTTP\_RETRY\_COUNT = 5**

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

**INFO\_CONNECT\_TIMEOUT\_SEC = 5**

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

**INFO\_RES = 'info.json'**

The URL to use for server info requests

**INFO\_RESPONSE\_TIMEOUT\_SEC = 15**

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

**LAST\_REQUESTS\_RESPONSE = None**

This variable will be updated with the last requests response object that was received

**LAST\_RESPONSE\_INFO = {}**

This variable will be updated with the information from the most recent call to `_get_response()`

**RECORD\_ALL\_REQUESTS = False**

Controls whether each requests response object is appended to the `self.ALL_REQUESTS_RESPONSES` list

**REQUESTS\_SESSION = None**

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/" xmlns:'**

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

**Parameters****command** : str

- text to use in command node when building template

**object\_list** : str

- XML string to use in object list node when building template

**kwargs** : dict, optional

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

**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

**Returns**body : 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.

**Parameters**headers : dict

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

**Returns**headers : dict

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

**\_create\_add\_object\_body** (obj, \*\*kwargs)

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

**Parameters**obj : *taniumpy.object\_types.base.BaseType*

- object to convert into XML

**kwargs** : dict, optional

- any number of attributes that can be set via *taniumpy.object\_types.options.Options* that control the servers response.

**Returns**obj\_body : str

- The XML request body created from *pytan.sessions.Session.\_build\_body()*

**\_create\_delete\_object\_body** (obj, \*\*kwargs)

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

**Parameters**obj : *taniumpy.object\_types.base.BaseType*

- object to convert into XML

**kwargs** : dict, optional

- any number of attributes that can be set via *taniumpy.object\_types.options.Options* that control the servers response.

**Returns**obj\_body : str

- The XML request body created from *pytan.sessions.Session.\_build\_body()*

**\_create\_get\_object\_body** (obj, \*\*kwargs)

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

**Parameters**obj : *taniumpy.object\_types.base.BaseType*

- object to convert into XML

**kwargs** : dict, optional

- any number of attributes that can be set via *taniumpy.object\_types.options.Options* that control the servers response.

**Returns**`obj_body` : str

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

**`_create_get_result_data_body`** (*obj*, *\*\*kwargs*)

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

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

- object to convert into XML

**kwargs** : dict, optional

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

**Returns**`obj_body` : str

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

**`_create_get_result_info_body`** (*obj*, *\*\*kwargs*)

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

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

- object to convert into XML

**kwargs** : dict, optional

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

**Returns**`obj_body` : str

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

**`_create_run_plugin_object_body`** (*obj*, *\*\*kwargs*)

Utility method for building an XML Request Body to run a plugin

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

- object to convert into XML

**kwargs** : dict, optional

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

**Returns**`obj_body` : str

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

**`_create_update_object_body`** (*obj*, *\*\*kwargs*)

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

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

- object to convert into XML

**kwargs** : dict, optional

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

**Returns**`obj_body` : str

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

**`_extract_resultxml`** (*response\_body*)

Utility method to get the 'ResultXML' element from an XML body

**Parameters***response\_body* : str

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

**Returns***ret* : str 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(

**Parameter***target* : list

- index0 : label : human friendly name to refer to search\_path
- index1 : search\_path : / seperated search path to find a given value from info.json

**diags** : dict

- flattened dictionary of info.json diagnostics

**Returns**dict

- label : same as provided in *target* index0 (label)
- result : value resolved from `pytan.sessions.Session._resolve_stat_target()` for *target* index1 (search\_path)

**`_flatten_server_info`** (*structure*)

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

**Parameters***structure*

- dict/tuple/list to flatten

**Returns**flattened

- the dict/tuple/list flattened out

**`_full_url`** (*url, \*\*kwargs*)

Utility method for constructing a full url

**Parameters***url* : 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

**Returns***full\_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

**Parameters****part**: int, float

**whole**: int, float

**Returns****str** : 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.

**Parameters****request\_body** : str

- the XML request body to send to the server

**connect\_timeout**: int, optional

- default: `self.SOAP_CONNECT_TIMEOUT_SEC`
- timeout in seconds for connection to host

**response\_timeout**: int, optional

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

**retry\_auth**: bool, optional

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

**retry\_count**: int, optional

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

**pytan\_help** : str, optional

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

**Returns****body** : str

- str containing body of response from server

**See also:**

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

**\_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.

**Parameters****host** : str

- host to connect to

**port** : int

- port to connect to

**url** : str

- url to fetch on the server

**headers** : dict, optional

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

**connect\_timeout** : int, optional

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

**response\_timeout** : int, optional

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

**debug** : bool, optional

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

**pytan\_help** : str, optional

- default: ''
- help string to add to self.LAST\_REQUESTS\_RESPONSE.pytan\_help

**perform\_xml\_clean** : bool, optional

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

**clean\_restricted** : bool, optional

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

**log\_clean\_messages** : bool, optional

- default: True
- True: When XML cleaning the response\_body, enable logging messages about invalid/restricted matches
- False: When XML cleaning the response\_body, disable logging messages about invalid/restricted matches

**log\_bad\_characters** : bool, optional

- default: False
- False: When XML cleaning the response\_body, disable logging messages about the actual characters that were invalid/restricted
- True: When XML cleaning the response\_body, enable logging messages about the actual characters that were invalid/restricted

**Returns**`body` : 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.

**Parameters**`host` : str

- host to connect to

**`port`** : int

- port to connect to

**`url`** : str

- url to fetch on the server

**`body`** : str, optional

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

**`headers`** : dict, optional

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

**`connect_timeout`** : int, optional

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

**`response_timeout`** : int, optional

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

**`debug`** : bool, optional

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

**`pytan_help`** : str, optional

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

**`perform_xml_clean`** : bool, optional

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

**`clean_restricted`** : bool, optional

- default: True

- True: When XML cleaning the response\_body, remove restricted characters as well as invalid characters
- False: When XML cleaning the response\_body, remove only invalid characters

**log\_clean\_messages** : bool, optional

- default: True
- True: When XML cleaning the response\_body, enable logging messages about invalid/restricted matches
- False: When XML cleaning the response\_body, disable logging messages about invalid/restricted matches

**log\_bad\_characters** : bool, optional

- default: False
- False: When XML cleaning the response\_body, disable logging messages about the actual characters that were invalid/restricted
- True: When XML cleaning the response\_body, enable logging messages about the actual characters that were invalid/restricted

**Returns**body : str

- str containing body of response from server

**See also:**

[`pytan.xml\_clean.xml\_cleaner\(\)`](#) function to remove invalid/bad characters from XML responses

**`_invalid_server_version()`**

Utility method to find out if self.server\_version is valid or not

**`_parse_versioning(**kwargs)`**

Parses self.server\_version into a dictionary

**Returns**dict

- dict of parsed tanium server version containing keys: major, minor, revision, and build

## Notes

- If Session is unable to fetch info.json properly for some reason, then self.server\_version will be “Unable to determine”
- If Session has not yet fetched info.json, then self.server\_version will be “Not yet determined”

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

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

**Parameters**body : 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*

**Returns**ret : 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

**Parameters**headers : dict

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

**Returns**headers : dict

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

**\_resolve\_stat\_target** (*search\_path*, *diags*)

Utility method for resolving the value of search\_path in info.json and returning the value

**Parameters**search\_path : str

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

**diags** : dict

- flattened dictionary of info.json diagnostics

**Returns**str

- value resolved from *diags* for *search\_path*

**\_start\_stats\_thread** (*\*\*kwargs*)

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

**\_stats\_loop** (*\*\*kwargs*)

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

**Parameters**obj : *taniumpy.object\_types.base.BaseType*

- object to add

**Returns**obj : *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



**Parameters****username** : str, optional

- default: None
- username to authenticate as

**password** : str, optional

- default: None
- password for *username*

**session\_id** : str, optional

- default: None
- session\_id to authenticate with, this will be used in favor of username/password if all 3 are supplied.

**persistent**: bool, optional

- default: False
- False: do not request a persistent session (returns a session\_id that expires 5 minutes after last use)
- True: do request a persistent (returns a session\_id that expires 1 week after last use)

**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.

**delete** (*obj*, *\*\*kwargs*)

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

**Parametersobj** : *taniumpy.object\_types.base.BaseType*

- object to delete

**Returnsobj** : *taniumpy.object\_types.base.BaseType*

- deleted object

**disable\_stats\_loop** (*sleep=None*)

Disables the stats loop thread, which will print out the results of *pytan.sessions.Session.get\_server\_stats()* every *pytan.sessions.Session.STATS\_LOOP\_SLEEP\_SEC*

**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 stats loop thread, which will print out the results 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 a thread which checks *self.STATS\_LOOP\_ENABLED* before running *pytan.sessions.Session.get\_server\_stats()*

**find**(*obj*, **\*\*kwargs**)

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

**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 appropriate *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 received from info.json into a python friendly format

#### Notes

- 6.2 /info.json is only available on soap port (default port: 444)
- 6.5 /info.json is only available on server port (default port: 443)

**get\_server\_stats** (*\*\*kwargs*)

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

**Returns**str

- str containing stats from /info.json

**See also:**

`pytan.sessions.Session.STATS_LOOP_TARGETS` list of dict containing stat keys to pull from /info.json

**get\_server\_version** (*\*\*kwargs*)

Tries to parse the server version from /info.json

**Returns**str

- 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.

**Parameters**url : str

- url to fetch on the server

**host** : str, optional

- default: self.host
- host to connect to

**port** : int, optional

- default: self.port
- port to connect to

**headers** : dict, optional

- default: { }
- headers to supply as part of GET request

**connect\_timeout** : int, optional

- default: self.SOAP\_CONNECT\_TIMEOUT\_SEC

- timeout in seconds for connection to host

**response\_timeout** : int, optional

- default: self.SOAP\_RESPONSE\_TIMEOUT\_SEC

- timeout in seconds for response from host

**debug** : bool, optional

- default: self.HTTP\_DEBUG

- False: do not print requests debug messages

- True: print requests debug messages

**auth\_retry** : bool, optional

- default: self.HTTP\_AUTH\_RETRY

- True: retry authentication with username/password if session\_id fails

- False: throw exception if session\_id fails

**retry\_count** : int, optional

- default: self.HTTP\_RETRY\_COUNT

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

**pytan\_help** : str, optional

- default: ''

- help string to add to self.LAST\_REQUESTS\_RESPONSE.pytan\_help

**Returns**body : str

- str containing body of response from server

**See also:**

*[pytan.sessions.Session.\\_http\\_get\(\)](#)* private method used to perform the actual HTTP GET

**http\_post** (\*\*kwargs)

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

**Parameters**url : str, optional

- default: self.SOAP\_RES

- url to fetch on the server

**host** : str, optional

- default: self.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

**Returns**body : str

- str containing body of response from server

**See also:**

[\*pytan.sessions.Session.\\_http\\_post\(\)\*](#) private method used to perform the actual HTTP POST

**is\_auth**

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

**Returns**bool

- True: if self.\_session\_id or self.\_username and \_self.password are set

- False: if not

**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.

**Parameters***all\_session\_ids* : bool, optional

- default: False
- False: only log out the current session id for the current user
- True: log out ALL session id's associated for the current user

**pytan\_help** : str, optional

- default: ''
- help string to add to self.LAST\_REQUESTS\_RESPONSE.pytan\_help

**platform\_is\_6\_2** (*\*\*kwargs*)

Check to see if self.server\_version\_dict matches 6.2.xxx.xxx

**Returns**bool

- True if self.server\_version\_dict major == 6 and minor == 2
- False otherwise

**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

**Parameters***obj* : *taniumpy.object\_types.base.BaseType*

- object to run

**Returns***obj* : *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

**Parameters***obj* : *taniumpy.object\_types.base.BaseType*

- object to save

**Returns***obj* : *taniumpy.object\_types.base.BaseType*

- saved object

**server\_version = 'Not yet determined'**

version string of server, will be updated when get\_server\_version() is called

**server\_version\_dict = {}**

dictionary of self.server\_version parsed into major, minor, build, revision – will be updated when get\_server\_version() is called

**session\_id**

Property to fetch the session\_id for this object

**Return**self.\_session\_id : str

**setup\_logging** ()

## 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’

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

- PyTan handler to use for GetResultInfo calls

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

- object to poll for progress

**polling\_secs** : int, optional

- default: 5
- Number of seconds to wait in between GetResultInfo loops

**complete\_pct** : int/float, optional

- default: 100
- Percentage of mr\_tested out of estimated\_total to consider the question “done”

**override\_timeout\_secs** : int, optional

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

**ACTION\_DONE\_KEY** = ‘success’

key in action\_result\_map that maps to an action being done

**COMPLETE\_PCT\_DEFAULT** = 100

default value for self.complete\_pct

**EXPIRATION\_ATTR** = ‘expiration\_time’

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

**OBJECT\_TYPE**

valid type of object that can be passed in as obj to \_\_init\_\_

alias of `Action`

**RUNNING\_STATUSES** = [‘active’, ‘open’]

values for status attribute of action object that mean the action is running

**\_derive\_object\_info** (*\*\*kwargs*)

Derive self.object\_info from self.obj

**\_derive\_package\_spec** (*\*\*kwargs*)

Get the package\_spec attribute for self.obj, then fetch the full package\_spec object



**`_derive_result_map (**kwargs)`**

Determine what self.result\_map should contain for the various statuses an action can have

A package object has to have a verify\_group defined on it in order for deploy action verification to trigger. That can be only done at package creation/update

If verify\_enable is True, then the various result states for an action change

**`_derive_status (**kwargs)`**

Get the status attribute for self.obj

**`_derive_stopped_flag (**kwargs)`**

Get the stopped\_flag attribute for self.obj

**`_derive_target_group (**kwargs)`**

Get the target\_group attribute for self.obj, then fetch the full group object

**`_derive_verify_enabled (**kwargs)`**

Determine if this action has verification enabled

**`_fix_group (g, **kwargs)`**

Sets ID to null on a group object and all of it's sub\_groups, needed for 6.5

**`_post_init (**kwargs)`**

Post init class setup

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

Method to poll Result Info for self.obj until the percentage of 'finished\_count' out of 'self.passed\_count' is greater than or equal to self.complete\_pct

- finished\_count is calculated from a full GetResultData call that is parsed into self.action\_result\_map
- self.passed\_count is calculated by the question asked before this method is called. that question has no selects, but has a group that is the same group as the action for this object

**`run (callbacks={}, **kwargs)`**

Poll for action data and issue callbacks.

**Parameters**callbacks : dict

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

- 'SeenProgressChanged'
- 'SeenAnswersComplete'
- 'FinishedProgressChanged'
- 'FinishedAnswersComplete'

•Each callback should be a function that accepts:

- 'poller': a poller instance
- 'pct': a percent complete
- 'kwargs': a dict of other args

## Notes

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

- Any callback can choose to stop the poller by calling `pytan.poller.QuestionPoller.stop()`
- Polling will be stopped only when one of the callbacks calls the `pytan.poller.QuestionPoller.stop()` method or the answers are complete.
- Any callbacks can call `pytan.poller.QuestionPoller.setPercentCompleteThreshold()` to change what “done” means on the fly

**seen\_eq\_passed\_loop** (*callbacks={}, \*\*kwargs*)

Method to poll Result Info for self.obj until the percentage of ‘seen\_count’ out of ‘self.passed\_count’ is greater than or equal to self.complete\_pct

- seen\_count is calculated from an aggregate GetResultData
- self.passed\_count is calculated by the question asked before this method is called. that question has no selects, but has a group that is the same group as the action for this object

**class** `pytan.pollers.QuestionPoller` (*handler, obj, \*\*kwargs*)

Bases: `object`

A class to poll the progress of a Question.

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

- ProgressChanged
- AnswersChanged
- AnswersComplete

**Parametershandler** : `pytan.handler.Handler`

- PyTan handler to use for GetResultInfo calls

**obj** : `taniumpy.object_types.question.Question`

- object to poll for progress

**polling\_secs** : int, optional

- default: 5
- Number of seconds to wait in between GetResultInfo loops

**complete\_pct** : int/float, optional

- default: 99
- Percentage of mr\_tested out of estimated\_total to consider the question “done”

**override\_timeout\_secs** : int, optional

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

**COMPLETE\_PCT\_DEFAULT = 99**

default value for self.complete\_pct

**EXPIRATION\_ATTR = ‘expiration’**

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

**EXPIRY\_FALLBACK\_SECS = 600**

If the EXPIRATION\_ATTR of *obj* can’t be automatically determined, then this is used as a fallback for timeout - polling will failed after this many seconds if completion not reached

**OBJECT\_TYPE**

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

alias of `Question`

**OVERRIDE\_TIMEOUT\_SECS\_DEFAULT = 0**

default value for `self.override_timeout_secs`

**POLLING\_SECS\_DEFAULT = 5**

default value for `self.polling_secs`

**STR\_ATTRS = ['object\_info', 'polling\_secs', 'override\_timeout\_secs', 'complete\_pct', 'expiration']**

Class attributes to include in `__str__` output

**\_\_derive\_attribute** (*attr, fallback=''*, *\*\*kwargs*)

Derive an attributes value from `self.obj`

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

**Parameters***attr* : string

string of attribute name to fetch from `self.obj`

**fallback** : string

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

**Returns***val* : perspective

The value of the attr from `self.obj`

**\_\_derive\_expiration** (*\*\*kwargs*)

Derive the expiration datetime string from a object

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

**\_\_derive\_object\_info** (*\*\*kwargs*)

Derive `self.object_info` from `self.obj`

**\_\_post\_init** (*\*\*kwargs*)

Post init class setup

**\_\_refetch\_obj** (*\*\*kwargs*)

Utility method to re-fetch a object

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

**\_\_stop = False**

Controls whether a `run()` loop should stop or not

**get\_result\_data** (*\*\*kwargs*)

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

**get\_result\_info** (*\*\*kwargs*)

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

**handler = None**

The Handler object for this poller

**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

**result\_info** = None

This will be updated with the ResultInfo object during run() calls

**run** (*callbacks*={}, *\*\*kwargs*)

Poll for question data and issue callbacks.

**Parameters***callbacks* : dict

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

–'ProgressChanged'

–'AnswersChanged'

–'AnswersComplete'

•**Each callback should be a function that accepts:**

–'poller': a poller instance

–'pct': a percent complete

–'kwargs': a dict of other args

## Notes

- Any callback can choose to get data from the session by calling poller.get\_result\_data() or new info by calling poller.get\_result\_info()
- Any callback can choose to stop the poller by calling poller.stop()
- Polling will be stopped only when one of the callbacks calls the stop() method or the answers are complete.
- Any callback can call setPercentCompleteThreshold to change what "done" means on the fly

**run\_callback** (*callbacks*, *callback*, *pct*, *\*\*kwargs*)

Utility method to find a callback in callbacks dict and run it

**set\_complect\_pct** (*val*)

Set the complete\_pct to a new value

**Parameters***val* : 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.

**Parameters***handler* : [pytan.handler.Handler](#)

PyTan handler to use for GetResultInfo calls

**export\_id** : str

- ID of server side export

**polling\_secs** : int, optional

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

**timeout\_secs** : int, optional

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

**POLLING\_SECS\_DEFAULT = 2**  
default value for self.polling\_secs

**STR\_ATTRS = ['export\_id', 'polling\_secs', 'timeout\_secs', 'sse\_status']**  
Class attributes to include in \_\_str\_\_ output

**TIMEOUT\_SECS\_DEFAULT = 600**  
default value for self.timeout\_secs

**\_post\_init** (\*\*kwargs)  
Post init class setup

**export\_id = None**  
The export\_id for this poller

**get\_sse\_data** (\*\*kwargs)  
Function to get the data of a server side export  
Constructs a URL via: export/\${export\_id}.gz and performs an authenticated HTTP get

**get\_sse\_status** (\*\*kwargs)  
Function to get the status of a server side export  
Constructs a URL via: export/\${export\_id}.status and performs an authenticated HTTP get

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

**sse\_status\_has\_completed\_loop** (\*\*kwargs)  
Method to poll the status file for a server side export until it contains 'Completed'

## 1.2.5 pytan.constants module

### PyTan Constants

This contains a number of constants that drive PyTan.

`pytan.constants.DEBUG_FORMAT = "[% (lineno)-5d - %(filename)20s: %(funcName)s()] %(asctime)s\n%(levelname)-8s %"`  
Logging format for debugformat=True

`pytan.constants.EXPORT_MAPS = {'ResultSet': {'xml': [], 'json': [], 'csv': [{'valid_list_types': ['str', 'unicode'], 'key': 'h`

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

- 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': ['<', 'less than']}]
```

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

- `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': 'UserList', 'delete': True, 'create_json': True}}
```

**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 except for the specified loglevel'), (1, {'stats': 'WARN'}, 'Sets all loggers to only output at INFO or above except for the specified loglevel'), (2, {'stats': 'INFO'}, 'Sets all loggers to only output at DEBUG or above except for the specified loglevel'), (3, {'stats': 'DEBUG'}, 'Sets all loggers to output at all levels except for the specified loglevel')]
```

**Map for loglevel(int) -> logger -> logger level(logging.INFO|WARN|DEBUG|...). Higher loglevels will include all levels up to and including the specified loglevel.**

- `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': {'ignore': True, 'manual': True, 'multi': True, 'single': True, 'delete': True, 'create_json': True}}]
```

**Maps a given human string into the various options for filters used by the SOAP API. Also used to verify that a manually entered filter is valid.**

- `human`: the human string that can be used after `'opt: '`. Ex: `'opt:value_type:value'`
- `destination`: the type of object this option can be applied to (filter or group)
- `attrs`: the attributes and their values used by the SOAP API when building a filter with an option that matches *human*

- attr**: the attribute used by the SOAP API when building a filter with an option that matches *human*. value is pulled from after a **:** when only **attr** exists for an option map, and not **attrs**.
- valid\_values**: if supplied, the list of valid values for this option
- valid\_type**: performs type checking on the value supplied to verify it is correct
- human\_type**: the human string for the value type if the option requires a value

```
pytan.constants.OPTION_RE = '\\s*opt:'
```

The regex that is used to find options in a string. Ex: *Sensor1, that contains blah, opt:ignore\_case, opt:max\_data\_age:3600*

```
pytan.constants.PARAM_DELIM = '||'
```

The string to surround a parameter with when passing parameters to the SOAP API for a sensor in a question.  
Ex: `||parameter_key||`

```
pytan.constants.PARAM_KEY_SPLIT = '='
```

The string that is used to split parameter key from parameter value. Ex: *key1=value1*

```
pytan.constants.PARAM_RE = '(?<!(\\\\\\\\)\\\\{(.*)}(?<!(\\\\\\\\)\\\\{'
```

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

```
pytan.constants.PARAM_SPLIT_RE = '(?<!(\\\\\\\\),'
```

The regex that is used to split multiple parameters. Ex: *key1=value1, key2=value2*

```
pytan.constants.Q_OBJ_MAP = {'manual': {'handler': 'ask_manual'}, 'saved': {'handler': 'ask_saved'}, 'parsed': {'hand
```

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: 'IPAC
```

Maps a Result type from the Tanium SOAP API from an int to a string

```
pytan.constants.SSE_CRASH_MAP = [{'major': 6, 'build': 4300, 'minor': 5, 'revision': 314}]
```

Mapping of versions to watch out for crashes/handle bugs for server side export

```
pytan.constants.SSE_FORMAT_MAP = [('csv', '0', 0), ('xml', '1', 1), ('xml_obj', '1', 1), ('cef', '2', 2)]
```

Mapping of human friendly strings to API integers for server side export

```
pytan.constants.SSE_RESTRICT_MAP = {1: [{'major': 6, 'build': 4300, 'minor': 5, 'revision': 314}], 2: [{'major': 6, 'bui
```

Mapping of API integers for server side export format to version support

```
pytan.constants.TIME_FORMAT = '%Y-%m-%dT%H:%M:%S'
```

Tanium's format for date time strings

## 1.2.6 pytan.utils module

Collection of classes and methods used throughout *pytan*

```
class pytan.utils.SplitStreamHandler
```

Bases: `logging.Handler`

Custom `logging.Handler` class that sends all messages that are `logging.INFO` and below to `STDOUT`, and all messages that are `logging.WARNING` and above to `STDERR`

**emit** (*record*)

`pytan.utils.apply_options_obj(options, obj, dest)`

Updates an object with options

**Parameters**`options` : dict

- dict containing options definition

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

- TaniumPy object to apply *options* to

`dest` : list of str

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

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

- TaniumPy object updated with attributes from *options*

`pytan.utils.build_group_obj(q_filter_defs, q_option_defs)`

Creates a Group object from *q\_filter\_defs* and *q\_option\_defs*

**Parameters**`q_filter_defs` : list of dict

- List of dict that are question filter definitions

`q_option_defs` : dict

- dict of question filter options

**Returns**`group_obj` : `taniumpy.object_types.group.Group`

- Group object with list of `taniumpy.object_types.filter.Filter` built from *q\_filter\_defs* and *q\_option\_defs*

`pytan.utils.build_manual_q(selectlist_obj, group_obj)`

Creates a Question object from *selectlist\_obj* and *group\_obj*

**Parameters**`selectlist_obj` : `taniumpy.object_types.select_list.SelectList`

- SelectList object to add to Question object

`group_obj` : `taniumpy.object_types.group.Group`

- Group object to add to Question object

**Returns**`add_q_obj` : `taniumpy.object_types.question.Question`

- Question object built from *selectlist\_obj* and *group\_obj*

`pytan.utils.build_metadatalist_obj(properties, nameprefix='')`

Creates a MetadataList object from *properties*

**Parameters**`properties` : list of list of str

- list of lists, each list having two str - str 1: property key, str2: property value

`nameprefix` : str

- prefix to insert in front of property key when creating MetadataItem

**Returns**`metadatalist_obj` : `taniumpy.object_types.metadata_list.MetadataList`

- MetadataList object with list of `taniumpy.object_types.metadata_item.MetadataItem` built from *properties*

`pytan.utils.build_param_obj(key, val, delim='')`

Creates a Parameter object from key and value, surrounding key with *delim*



**Parameters**`key` : str

- key to use for parameter

**value** : str

- value to use for parameter

**delim** : str

- str to surround key with when adding to parameter object

**Returns**`param_obj` : *taniumpy.object\_types.parameter.Parameter*

- Parameter object built from key and val

`pytan.utils.build_param_objlist(obj, user_params, delim='', derive_def=False, empty_ok=False)`

Creates a ParameterList object from user\_params

**Parameters**`obj` : *taniumpy.object\_types.base.BaseType*

- TaniumPy object to verify parameters against

**user\_params** : dict

- dict describing key and value of user supplied params

**delim** : str

- str to surround key with when adding to parameter object

**derive\_def** : bool, optional

- False: Do not derive default values, and throw a *pytan.exceptions.HandlerError* if user did not supply a value for a given parameter
- True: Try to derive a default value for each parameter if user did not supply one

**empty\_ok** : bool, optional

- False: If user did not supply a value for a given parameter, throw a *pytan.exceptions.HandlerError*
- True: If user did not supply a value for a given parameter, do not add the parameter to the ParameterList object

**Returns**`param_objlist` : *taniumpy.object\_types.parameter\_list.ParameterList*

ParameterList object with list of *taniumpy.object\_types.parameter.Parameter* built from user\_params

`pytan.utils.build_selectlist_obj(sensor_defs)`

Creates a SelectList object from sensor\_defs

**Parameters**`sensor_defs` : list of dict

- List of dict that are sensor definitions

**Returns**`select_objlist` : *taniumpy.object\_types.select\_list.SelectList*

- SelectList object with list of *taniumpy.object\_types.select.Select* built from *sensor\_defs*

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

Utility method for getting percentage of whole

**Parameters****percent**: int, float

**whole**: int, float

**Returns****int** : 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`

**Parameters****debug** : bool, optional

- False : set logging format for console handler to `pytan.constants.INFO_FORMAT`

- True : set logging format for console handler to `pytan.constants.DEBUG_FORMAT`

`pytan.utils.check_dictkey(d, key, valid_types, valid_list_types)`

Yet another method to check a dictionary for a key

**Parameters****d** : dict

- dictionary to check for key

**key** : str

- key to check for in d

**valid\_types** : list of str

- list of str of valid types for key

**valid\_list\_types** : list of str

- if key is a list, validate that all values of list are in valid\_list\_types

`pytan.utils.check_for_help(kwargs)`

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

**Parameters****kwargs** : dict

- dict of keyword args

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

Checks that def\_dict has key

**Parameters****def\_dict** : dict

- Definition dictionary

**key** : str

- key to check for in def\_dict

**keytypes** : list of str

- list of str of valid types for key

**keysubtypes** : list of str

- if key is a dict or list, validate that all values of dict or list are in keysubtypes

**req** : bool

- False: key does not have to be in def\_dict

- True: key must be in def\_dict, throw `pytan.exceptions.DefinitionParserError` if not

`pytan.utils.clean_kwargs` (*kwargs*, *keys=None*)

Removes each key from kwargs dict if found

**Parameters***kwargs* : dict

- dict of keyword args

**keys** : list of str, optional

- default: ['obj', 'pytan\_help', 'objtype']
- list of str's of keys to remove from kwargs

**Returns***clean\_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

**Parameters***obj* : *taniumpy.object\_types.base.BaseType*

- Object to copy

**skip\_attrs** : list of str

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

**Returns***new\_obj* : *taniumpy.object\_types.base.BaseType*

- Copied object with attributes in skip\_attrs skipped

`pytan.utils.copy_package_obj_for_action` (*obj*, *skip\_attrs=None*)

Returns a new class of package obj with with out any attributes in skip\_attrs specified

**Parameters***obj* : *taniumpy.object\_types.base.BaseType*

- Object to copy

**skip\_attrs** : list of str

- default: None
- list of attribute str's to skip copying over to new object, default if None: ['id', 'deleted\_flag', 'available\_time', 'creation\_time', 'modification\_time', 'source\_id']

**Returns***new\_obj* : *taniumpy.object\_types.base.BaseType*

- Copied object with attributes in skip\_attrs skipped

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

Get a timestr for *dt*

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

- datetime object

**Return***timestr*: str

- the timestr for *dt* in taniums format

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

Turns a package str into a package definition

**Parameters***package* : str

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

**Returns**`package_def` : dict

- dict parsed from *sensors*

`pytan.utils.dehumanize_question_filters` (*question\_filters*)

Turns a `question_filters` str or list of str into a question filter definition

**Parameters**`question_filters` : str, list of str

- A str or list of str that describes a sensor for a question filter(s) and optionally a selector and/or filter

**Returns**`question_filter_defs` : list of dict

- list of dict parsed from *question\_filters*

`pytan.utils.dehumanize_question_options` (*question\_options*)

Turns a `question_options` str or list of str into a question option definition

**Parameters**`question_options` : str, list of str

- A str or list of str that describes question options

**Returns**`question_option_defs` : list of dict

- list of dict parsed from *question\_options*

`pytan.utils.dehumanize_sensors` (*sensors*, *key*='sensors', *empty\_ok*=True)

Turns a `sensors` str or list of str into a sensor definition

**Parameters**`sensors` : str, list of str

- A str or list of str that describes a sensor(s) and optionally a selector, parameters, filter, and/or options

**key** : str, optional

- Name of key that user should have provided *sensors* as

**empty\_ok** : bool, optional

- False: *sensors* is not allowed to be empty, throw `pytan.exceptions.HumanParserError` if it is empty
- True: *sensors* is allowed to be empty

**Return**`sensor_defs` : list of dict

- list of dict parsed from *sensors*

`pytan.utils.derive_param_default` (*obj\_param*)

Derive a parameter default

**Parameters**`obj_param` : dict

- parameter dict from TaniumPy object

**Returns**`def_val` : str

- default value derived from *obj\_param*

`pytan.utils.empty_obj` (*taniumpy\_object*)

Validate that a given TaniumPy object is not empty

**Parameters**`taniumpy_object` : `taniumpy.object_types.base.BaseType`

- object to check if empty

**Returns**bool

- True if *taniumpy\_object* is considered empty, False otherwise

`pytan.utils.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*

**Parameters** *ss* : str

- A str that may or may not have a filter identified by ‘, that HUMAN VALUE’

**Return** *ss* : str

- str *s* without the `parsed_filter` included

**parsed\_filter** : dict

- filter attributes mapped from filter from *s* if any found

`pytan.utils.extract_options(s)`

Extracts options from str *s*

**Parameters** *ss* : str

- A str that may or may not have options identified by ‘, opt:name[:value]’

**Return** *ss* : str

- str *s* without the `parsed_options` included

**parsed\_options** : list

- options extracted from *s* if any found

`pytan.utils.extract_params(s)`

Extracts parameters from str *s*

**Parameters** *ss* : str

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

**Return** *ss* : str

- str *s* without the `parsed_params` included

**parsed\_params** : list

- parameters extracted from *s* if any found

`pytan.utils.extract_selector(s)`

Extracts a selector from str *s*

**Parameters** *ss* : 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

**Return** *ss* : str

- str *s* without the `parsed_selector` included

**parsed\_selector** : str

- selector extracted from *s*, or ‘name’ if none found

`pytan.utils.func_timing(f)`

Decorator to add timing information around a function

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

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

**Parameters***d* : dict of str

- dict to sums of

**keys** : list of str

- list of keys to get sums of, if empty gets a sum of all keys

**negate** : bool

- only used if keys supplied
- False : get the sums of *d* that do match keys
- True : get the sums of *d* that do not match keys

**Returns***list\_len* : int

- sum of lists in *d* that match keys

`pytan.utils.get_filter_obj(sensor_def)`  
Creates a Filter object from *sensor\_def*

**Parameters***sensor\_def* : dict

- dict containing sensor definition

**Returns***filter\_obj* : `taniumpy.object_types.filter.Filter`

- Filter object created from *sensor\_def*

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

**Parameters***key* : str

- key to get from kwargs

**default** : int, optional

- default value to use if key not found in kwargs

**kwargs** : dict

- kwargs to get key from

**Returns***val* : int

value from key, or default if supplied

`pytan.utils.get_now()`  
Get current time in human friendly format

**Returns***str* :

str of current time return from `human_time()`

`pytan.utils.get_obj_map(objtype)`  
Gets an object map for *objtype*

**Parameters***objtype* : str

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

**Returns**`obj_map` : dict

- matching object map for `objtype` from `pytan.constants.GET_OBJ_MAP`

`pytan.utils.get_obj_params(obj)`

Get the parameters from a TaniumPy object and JSON load them

**obj**[`taniumpy.object_types.base.BaseType`]

- TaniumPy object to get parameters from

**Returns**`params` : dict

- JSON loaded dict of parameters from `obj`

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

Utility method for getting percentage of part out of whole

**Parameters**`part`: int, float

**whole**: int, float

**Returns**`int` : the percentage of part out of whole

`pytan.utils.get_q_obj_map(qtype)`

Gets an object map for `qtype`

**Parameters**`qtype` : str

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

**Returns**`obj_map` : dict

- matching object map for `qtype` from `pytan.constants.Q_OBJ_MAP`

`pytan.utils.get_taniumpy_obj(obj_map)`

Gets a taniumpy object from `obj_map`

**Parameters**`obj_map` : str

- str of taniumpy object to fetch

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

- matching taniumpy object for `obj_map`

`pytan.utils.human_time(t, tformat='%Y_%m_%d-%H_%M_%S-%Z')`

Get time in human friendly format

**Parameters**`t` : int, float, time

- either a unix epoch or struct\_time object to convert to string

**tformat** : str, optional

- format of string to convert time to

**Returns**`str` :

- `t` converted to str

`pytan.utils.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

**Parameters***v* : object

- python object to convert to JSON

**indent** : int, 2

- number of spaces to indent JSON string when pretty printing

**sort\_keys** : bool, True

- sort keys of JSON string when pretty printing

**Returns***str* :

- JSON pretty printed string

`pytan.utils.load_param_json_file(parameters_json_file)`

Opens a json file and sanity checks it for use as a parameters element for a taniumpy object

**Parameters***parameters\_json\_file* : str

- path to JSON file that describes an API object

**Returns***obj*

- contents of *parameters\_json\_file* de-serialized

`pytan.utils.load_taniumpy_from_json(json_file)`

Opens a json file and parses it into an taniumpy object

**Parameters***json\_file* : str

- path to JSON file that describes an API object

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

- TaniumPy object converted from json file

`pytan.utils.log_session_communication(h)`

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

**Parameters***h* : Handler object

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

`pytan.utils.map_filter(filter_str)`

Maps a filter *str* against `constants.FILTER_MAPS`

**Parameters***filter\_str* : str

- filter\_str* str that should be validated

**Returns***filter\_attrs* : dict

- dict containing mapped filter attributes for SOAP API

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

Maps an *opt* str against `constants.OPTION_MAPS`



**Parameters**`opt` : str

- option str that should be validated

**dest** : list of str

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

**Returns**`opt_attrs` : dict

- dict containing mapped option attributes for SOAP API

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

Maps a list of options using `map_option()`

**Parameters**`options` : list of str

- list of str that should be validated

**dest** : list of str

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

**Returns**`mapped_options` : dict

- dict of all mapped\_options

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

Parses and validates defs into new\_defs

**Parameters**`defname` : str

- Name of definition

**deftypes** : list of str

- list of valid types that defs can be

**strconv** : str

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

**empty\_ok** : bool

- True: defs is allowed to be empty
- False: defs is not allowed to be empty

**Returns**`new_defs` : list of dict

- parsed and validated defs

`pytan.utils.plugin_zip(p)`

Maps columns to values for each row in a plugins sql\_response and returns a list of dicts

**Parameters**`p` : `taniumpy.object_types.plugin.Plugin`

- plugin object

**Returns**`dict`

- the columns and result\_rows of the sql\_response in Plugin object zipped up into a dictionary

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

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

**Parameters**`address` : str

- hostname/ip address to check `port` on

**port** : int

- port to check on *address*

**timeout** : int, optional

- timeout after N seconds of not being able to connect

**Returns** `socket` or `False` :

- if connection succeeds, the socket object is returned, else `False` is returned

`pytan.utils.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

**Parameters** `name` : 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

**Parameters** `secs` : int

- seconds from now to get time str

**tz** : str, optional

- time zone to return string in, default is 'utc' - supplying anything else will supply local time

**Returns** str :

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

`pytan.utils.set_all_loglevels(level='DEBUG')`

Sets all loggers that the logging system knows about to a given logger level

`pytan.utils.set_log_levels(loglevel=0)`

Enables loggers based on loglevel and `pytan.constants.LOG_LEVEL_MAPS`

**Parameters** `loglevel` : int, optional

- loglevel to match against each item in `pytan.constants.LOG_LEVEL_MAPS` - each item that is greater than or equal to loglevel will have the according loggers set to their respective levels identified there-in.

`pytan.utils.setup_console_logging(gmt_tz=True)`

Creates a console logging handler using `SplitStreamHandler`

`pytan.utils.shrink_obj(obj, attrs=None)`

Returns a new class of obj with only id/name/hash defined

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

- Object to shrink

**attrs** : list of str

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

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

- Shrunk 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*

**Raises**`pytan.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

**Returns**`datetime.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

**Returns**str :

- The pretty printed string of *x*

```
pytan.utils.xml_pretty_resultobj(x)
```

Uses *xmltodict* to pretty print an the result-object element in XML str *x*

**Parameters**x : str

- XML string to pretty print

**Returns**str :

- The pretty printed string of result-object in *x*

```
pytan.utils.xml_pretty_resultxml(x)
```

Uses *xmltodict* to pretty print an the ResultXML element in XML str *x*

**Parameters**x : str

- XML string to pretty print

**Returns**str :

- The pretty printed string of ResultXML in *x*

## 1.2.7 pytan.binsupport module

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

```
class pytan.binsupport.CustomArgFormat(prog, indent_increment=2, max_help_position=24,
                                         width=None)
```

Bases: `argparse.ArgumentParser`, `argparse.RawDescriptionHelpFormatter`

Multiple inheritance Formatter class for `argparse.ArgumentParser`.

If a `argparse.ArgumentParser` class uses this as it's Formatter class, it will show the defaults for each argument in the *help* output

```
class pytan.binsupport.CustomArgParse(*args, **kwargs)
```

Bases: `argparse.ArgumentParser`

Custom `argparse.ArgumentParser` class which does a number of things:

- Uses `pytan.utils.CustomArgFormat` as it's Formatter class, if none was passed in
- Prints help if there is an error
- Prints the help for any subparsers that exist

```
error(message)
```

```
print_help(**kwargs)
```

```
class pytan.binsupport.HistoryConsole(locals=None, filename='<console>',
                                       histfile='/Users/jolsen/.console-history')
```

Bases: `code.InteractiveConsole`

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

### Examples

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

**init\_history** (*histfile*)

**static save\_history** (*histfile*)

`pytan.binsupport.add_ask_report_argparser` (*parser*)

Method to extend a `pytan.utils.CustomArgParse` class for command line scripts with arguments for scripts that need to supply export format subparsers for asking questions.

`pytan.binsupport.add_file_log` (*logfile*, *debug=False*)

Utility to add a log file from python's logging module

`pytan.binsupport.add_get_object_report_argparser` (*parser*)

Method to extend a `pytan.utils.CustomArgParse` class for command line scripts with arguments for scripts that need to supply export format subparsers for getting objects.

`pytan.binsupport.add_report_file_options` (*parser*)

Method to extend a `pytan.utils.CustomArgParse` class for command line scripts with arguments for scripts that need to supply export file and directory options.

`pytan.binsupport.csvdictwriter` (*rows\_list*, *\*\*kwargs*)

returns the *rows\_list* (list of dicts) as a CSV string

`pytan.binsupport.debug_list` (*debuglist*)

Utility function to print the variables for a list of objects

`pytan.binsupport.debug_obj` (*debugobj*)

Utility function to print the variables for an object

`pytan.binsupport.filter_filename` (*filename*)

Utility to filter a string into a valid filename

`pytan.binsupport.filter_sensors` (*sensors*, *filter\_platforms=[]*, *filter\_categories=[]*)

Utility to filter a list of sensors for specific platforms and/or categories

`pytan.binsupport.filter_sourced_sensors` (*sensors*)

Utility to filter out all sensors that have a *source\_id* specified (i.e. they are temp sensors created by the API)

`pytan.binsupport.get_all_headers` (*rows\_list*)

Utility to get all the keys for a list of dicts

`pytan.binsupport.get_grp_opts` (*parser*, *grp\_names*)

Used to get arguments in *parser* that match argument group names in *grp\_names*

**Parametersparser** : `argparse.ArgumentParser`

- `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

**Parameters**`parser` : `argparse.ArgumentParser`

- `ArgumentParser` object used to parse *all\_args*

**handler** : `pytan.handler.Handler`

- Instance of `Handler` created from command line args

**args** : args object

- args parsed from *parser*

**Returns**response

- response from `pytan.handler.Handler.ask_manual()`

`pytan.binsupport.process_ask_parsed_args(parser, handler, args)`

Process command line args supplied by user for ask parsed

**Parameters**`parser` : `argparse.ArgumentParser`

- `ArgumentParser` object used to parse *all\_args*

**handler** : `pytan.handler.Handler`

- Instance of `Handler` created from command line args

**args** : args object

- args parsed from *parser*

**Returns**response

- response from `pytan.handler.Handler.ask_parsed()`

`pytan.binsupport.process_ask_saved_args(parser, handler, args)`

Process command line args supplied by user for ask saved

**Parameters**`parser` : `argparse.ArgumentParser`

- `ArgumentParser` object used to parse *all\_args*

**handler** : `pytan.handler.Handler`

- Instance of `Handler` created from command line args

**args** : args object

- args parsed from *parser*

**Returns**response

- response from `pytan.handler.Handler.ask_saved()`

`pytan.binsupport.process_create_group_args(parser, handler, args)`

Process command line args supplied by user for create group object

**Parameters**`parser` : `argparse.ArgumentParser`

- `ArgumentParser` 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 json object

**Parametersparser** : `argparse.ArgumentParser`

- ArgumentParser 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.ArgumentParser`

- ArgumentParser 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.ArgumentParser`

- ArgumentParser 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

**Parameters**`parser` : `argparse.ArgumentParser`

- `ArgumentParser` object used to parse *all\_args*

**handler** : `pytan.handler.Handler`

- Instance of `Handler` created from command line args

**args** : args object

- args parsed from *parser*

**Returns**`response` : `taniumpy.object_types.base.BaseType`

- response from `pytan.handler.Handler.create_user()`

`pytan.binsupport.process_create_whitelisted_url_args (parser, handler, args)`

Process command line args supplied by user for create group object

**Parameters**`parser` : `argparse.ArgumentParser`

- `ArgumentParser` object used to parse *all\_args*

**handler** : `pytan.handler.Handler`

- Instance of `Handler` created from command line args

**args** : args object

- args parsed from *parser*

**Returns**`response` : `taniumpy.object_types.base.BaseType`

- response from `pytan.handler.Handler.create_group()`

`pytan.binsupport.process_delete_object_args (parser, handler, obj, args)`

Process command line args supplied by user for delete object

**Parameters**`parser` : `argparse.ArgumentParser`

- `ArgumentParser` 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*

**Returns**`response` : `taniumpy.object_types.base.BaseType`

- response from `pytan.handler.Handler.delete()`

`pytan.binsupport.process_deploy_action_args (parser, handler, args)`

Process command line args supplied by user for deploy action

**Parameters**`parser` : `argparse.ArgumentParser`

- `ArgumentParser` object used to parse *all\_args*

**handler** : `pytan.handler.Handler`



- Instance of Handler created from command line args

**args** : args object

- args parsed from *parser*

**Returns**response

- response from `pytan.handler.Handler.deploy_action()`

`pytan.binsupport.process_get_object_args(parser, handler, obj, args, report=True)`

Process command line args supplied by user for get object

**Parameters****parser** : `argparse.ArgumentParser`

- ArgParse object used to parse *all\_args*

**handler** : `pytan.handler.Handler`

- Instance of Handler created from command line args

**obj** : str

- Object type for get object

**args** : args object

- args parsed from *parser*

**Returns****response** : `taniumpy.object_types.base.BaseType`

- response from `pytan.handler.Handler.get()`

`pytan.binsupport.process_get_results_args(parser, handler, args)`

Process command line args supplied by user for getting results

**Parameters****parser** : `argparse.ArgumentParser`

- 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*

**Returns****report\_path, report\_contents** : tuple

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

`pytan.binsupport.process_handler_args(parser, args)`

Process command line args supplied by user for handler

**Parameters****parser** : `argparse.ArgumentParser`

- ArgParse object used to parse *all\_args*

**args** : args

- args parsed from *parser*

**Return****h** : `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*

**Returns**`report_path, report_contents` : tuple

- results from *pytan.handler.Handler.export\_to\_report\_file()* on the return of *pytan.handler.Handler.get\_result\_data()*

`pytan.binsupport.process_tsat_args(parser, handler, args)`

Process command line args supplied by user for tsat

**Parametersparser** : argparse.ArgParse

- ArgParse object used to parse *all\_args*

**handler** : *pytan.handler.Handler*

- Instance of Handler created from command line args

**args** : args object

- args parsed from *parser*

`pytan.binsupport.remove_file_log(logfile)`

Utility to remove a log file from python's logging module

`pytan.binsupport.setup_ask_manual_argparser(doc)`

Method to setup the base `pytan.utils.CustomArgParse` class for command line scripts using `pytan.utils.setup_parser()`, then add specific arguments for scripts that use *pytan* to ask manual questions.

`pytan.binsupport.setup_ask_parsed_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 parsed questions.

`pytan.binsupport.setup_ask_saved_argparser(doc)`

Method to setup the base `pytan.utils.CustomArgParse` class for command line scripts using `pytan.utils.setup_parser()`, then add specific arguments for scripts that use *pytan* to ask saved questions.

`pytan.binsupport.setup_create_group_argparser(doc)`

Method to setup the base `pytan.utils.CustomArgParse` class for command line scripts using `pytan.utils.setup_parser()`, then add specific arguments for scripts that use *pytan* to create a group.

`pytan.binsupport.setup_create_json_object_argparser(obj, doc)`

Method to setup the base `pytan.utils.CustomArgParse` class for command line scripts using `pytan.utils.setup_parser()`, then add specific arguments for scripts that use *pytan* to create objects from json files.

`pytan.binsupport.setup_create_package_argparser(doc)`

Method to setup the base `pytan.utils.CustomArgParse` class for command line scripts using `pytan.utils.setup_parser()`, then add specific arguments for scripts that use *pytan* to create a package.

`pytan.binsupport.setup_create_sensor_argparser(doc)`

Method to setup the base `pytan.utils.CustomArgParse` class for command line scripts using `pytan.utils.setup_parser()`, then add specific arguments for scripts that use *pytan* to create a sensor.

`pytan.binsupport.setup_create_user_argparser(doc)`

Method to setup the base `pytan.utils.CustomArgParse` class for command line scripts using `pytan.utils.setup_parser()`, then add specific arguments for scripts that use *pytan* to create a user.

`pytan.binsupport.setup_create_whitelisted_url_argparser(doc)`

Method to setup the base `pytan.utils.CustomArgParse` class for command line scripts using `pytan.utils.setup_parser()`, then add specific arguments for scripts that use *pytan* to create a whitelisted\_url.

`pytan.binsupport.setup_delete_object_argparser(obj, doc)`

Method to setup the base `pytan.utils.CustomArgParse` class for command line scripts using `pytan.utils.setup_parser()`, then add specific arguments for scripts that use *pytan* to delete objects.

`pytan.binsupport.setup_deploy_action_argparser(doc)`

Method to setup the base `pytan.utils.CustomArgParse` class for command line scripts using

`pytan.utils.setup_parser()`, then add specific arguments for scripts that use *pytan* to deploy actions.

`pytan.binsupport.setup_get_object_argparser(obj, doc)`

Method to setup the base `pytan.utils.CustomArgParse` class for command line scripts using `pytan.utils.setup_parser()`, then add specific arguments for scripts that use *pytan* to get objects.

`pytan.binsupport.setup_get_results_argparser(doc)`

Method to setup the base `pytan.utils.CustomArgParse` class for command line scripts using `pytan.utils.setup_parser()`, then add specific arguments for scripts that use *pytan* to get results for questions or actions.

`pytan.binsupport.setup_parent_parser(doc)`

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

`pytan.binsupport.setup_parser(desc, help=False)`

Method to setup the base `pytan.utils.CustomArgParse` class for command line scripts that use *pytan*. This establishes the basic arguments that are needed by all such scripts, such as:

- `--help`
- `--username`
- `--password`
- `--host`
- `--port`
- `--loglevel`
- `--debugformat`

`pytan.binsupport.setup_print_sensors_argparser(doc)`

Method to setup the base `pytan.utils.CustomArgParse` class for command line scripts using `pytan.utils.setup_parser()`, then add specific arguments for scripts that use *pytan* to print server info.

`pytan.binsupport.setup_print_server_info_argparser(doc)`

Method to setup the base `pytan.utils.CustomArgParse` class for command line scripts using `pytan.utils.setup_parser()`, then add specific arguments for scripts that use *pytan* to print sensor info.

`pytan.binsupport.setup_pytan_shell_argparser(doc)`

Method to setup the base `pytan.utils.CustomArgParse` class for command line scripts using `pytan.utils.setup_parser()`, then add specific arguments for scripts that use *pytan* to create a python shell.

`pytan.binsupport.setup_stop_action_argparser(doc)`

Method to setup the base `pytan.utils.CustomArgParse` class for command line scripts using `pytan.utils.setup_parser()`, then add specific arguments for scripts that use *pytan* to stop actions.

`pytan.binsupport.setup_tsat_argparser(doc)`

Method to setup the base `pytan.utils.CustomArgParse` class for command line scripts using `pytan.utils.setup_parser()`, then add specific arguments for scripts that use *pytan* to get objects.

`pytan.binsupport.version_check(reqver)`

Allows scripts using *pytan* to validate the version of the script against the version of *pytan*

**Parameters**`reqver` : str

- string containing version number to check against Exception

### 1.2.8 pytan.xml\_clean module

```
pytan.xml_clean.DEFAULT_REPLACEMENT = u'\ufffd'
```

```
pytan.xml_clean.INVALID_UNICODE_RAW_RE = u'[^t\n\r -\ud7ff\u0000-\ufffd]'
```

```
pytan.xml_clean.INVALID_UNICODE_RE = <_sre.SRE_Pattern object>
```

```
pytan.xml_clean.RESTRICTED_UNICODE_RAW_RE = u'[\x7f-\x84\x86-\x9f\u00dd-\ufdef]'
```

```
pytan.xml_clean.RESTRICTED_UNICODE_RE = <_sre.SRE_Pattern object>
```

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

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

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

**Parameterstext** : str**replacement** : str, optional

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

- if not supplied, will default to DEFAULT\_REPLACEMENT

**Returnsstr, cnt, RE** : tuple

- 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)
```

**Parameter**`text` : str

- str to clean

**replacement** : str, optional

- default: None
- if restricted characters found, they will be replaced with this
- if not supplied, will default to `DEFAULT_REPLACEMENT`

**Returns**`str, cnt, RE` : tuple

- `str` : the cleaned version of `text`
- `cnt` : the number of replacements that took place
- `RE` : the regex object that was used to do the replacements

```
pytan.xml_clean.xml_cleaner(s, encoding='utf-8', clean_restricted=True,  
                             log_clean_messages=True, log_bad_characters=False, replace-  
                             ment=None, **kwargs)
```

Removes invalid /restricted characters per XML 1.0 spec

**Parameters**`s` : str

- str to clean

**encoding** : str, optional

- default: 'utf-8'
- encoding of `s`

**clean\_restricted** : bool, optional

- default: True
- remove restricted characters from `s` or not

**log\_clean\_messages** : bool, optional

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

**log\_bad\_characters** : bool, optional

- default: False
- log bad character matches or not

**Returns**`str`

- the cleaned version of `s`

## 1.2.9 pytan Unit Tests

This contains unit tests for pytan.

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

```
class test_pytan_unit.TestDehumanizeExtractionUtils(methodName='runTest')  
    Bases: unittest.case.TestCase  
    test_extract_filter_invalid()
```

```

test_extract_filter_nofilter()
test_extract_filter_valid()
test_extract_filter_valid_all()
test_extract_options_invalid_option()
test_extract_options_many()
test_extract_options_missing_value_max_data_age()
test_extract_options_missing_value_value_type()
test_extract_options_nooptions()
test_extract_options_single()
test_extract_params()
test_extract_params_missing_seperator()
test_extract_params_multiparams()
test_extract_params_noparams()
test_extract_selector()
test_extract_selector_use_name_if_noselector()

class test_pytan_unit.TestDehumanizeQuestionFilterUtils (methodName='runTest')
    Bases: unittest.case.TestCase
        test_empty_filterlist()
        test_empty_filterstr()
        test_invalid_filter1()
        test_invalid_filter2()
        test_invalid_filter3()
        test_multi_filter_list()
        test_single_filter_list()
        test_single_filter_str()

class test_pytan_unit.TestDehumanizeQuestionOptionUtils (methodName='runTest')
    Bases: unittest.case.TestCase
        test_empty_optionlist()
        test_empty_optionstr()
        test_invalid_option1()
        test_invalid_option2()
        test_option_list_many()
        test_option_list_multi()
        test_option_list_single()
        test_option_str()

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

```

```
test_empty_args_dict()
test_empty_args_list()
test_empty_args_str()
test_multi_list_complex()
test_single_str()
test_single_str_complex1()
test_single_str_complex2()
test_single_str_with_filter()
test_valid_simple_list()
test_valid_simple_str_hash_selector()
test_valid_simple_str_id_selector()
test_valid_simple_str_name_selector()

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

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

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

    test_bad_chars_resultset_latin1()
        This XML file has some characters that are actually encoded as latin1 (as well as some restricted characters).

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

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

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

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

    test_empty_obj()
    test_get_now()
    test_get_obj_map()
    test_get_q_obj_map()
    test_invalid_port()
    test_is_dict()
    test_is_list()
    test_is_not_dict()
    test_is_not_list()
```



```

    test_is_not_num()
    test_is_not_str()
    test_is_num()
    test_is_str()
    test_jsonify()
    test_load_param_file_invalid_file()
    test_load_param_file_invalid_json()
    test_load_param_file_valid()
    test_load_taniumpy_file_invalid_file()
    test_load_taniumpy_file_invalid_json()
    test_version_higher()
    test_version_lower()

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

    classmethod setUpClass()
    test_build_group_obj()
    test_build_manual_q()
    test_build_selectlist_obj_invalid_filter()
    test_build_selectlist_obj_missing_value()
    test_build_selectlist_obj_noparamssensorobj_noparams()
        builds a selectlist object using a sensor obj with no params
    test_build_selectlist_obj_noparamssensorobj_withparams()
        builds a selectlist object using a sensor obj with no params, but passing in params (which should be added
        as of 1.0.4)
    test_build_selectlist_obj_withparamssensorobj_noparams()
        builds a selectlist object using a sensor obj with 4 params but not supplying any values for any of the
        params
    test_build_selectlist_obj_withparamssensorobj_withparams()
        builds a selectlist object using a sensor obj with 4 params but supplying a value for only one param

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

    test_invalid1()
    test_invalid2()
    test_valid1()
    test_valid2()

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

    test_parse_emptydict()
    test_parse_emptylist()

```

```
test_parse_emptystr()
test_parse_multi_filter()
test_parse_noargs()
test_parse_none()
test_parse_single_filter()
test_parse_str()

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

    test_invalid1()
    test_valid1()
    test_valid2()

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

    test_parse_emptydict()
    test_parse_emptylist()
    test_parse_emptystr()
    test_parse_list()
    test_parse_noargs()
    test_parse_none()
    test_parse_options_dict()
    test_parse_str()

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

    test_parse_complex()
        list with many items is parsed into same list
    test_parse_dict_hash()
        dict with hash is parsed into list of same dict
    test_parse_dict_id()
        dict with id is parsed into list of same dict
    test_parse_dict_name()
        dict with name is parsed into list of same dict
    test_parse_emptydict()
        args=={} throws exception
    test_parse_emptylist()
        args==[] throws exception
    test_parse_emptystr()
        args=="" throws exception
    test_parse_noargs()
        no args throws exception
```

```

    test_parse_none ()
        args==None throws exception

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

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

    test_invalid1 ()
    test_invalid2 ()
    test_invalid3 ()
    test_invalid4 ()
    test_valid1 ()
    test_valid2 ()
    test_valid3 ()
    test_valid4 ()

```

## 1.2.10 pytan Functional Tests

This contains valid functional tests for pytan.

These functional tests require a connection to a Tanium server in order to run. The connection info is pulled from the SERVER\_INFO dictionary in test/API\_INFO.py.

These tests all use *ddt*, a package that provides for data driven tests via JSON files.

```

class test_pytan_valid_server_tests.ValidServerTests (methodName='runTest')
    Bases: unittest.case.TestCase

    classmethod setUpClass ()

    setup_test ()

    classmethod tearDownClass ()

    test_invalid_create_object_1_invalid_create_sensor ()
    test_invalid_create_object_from_json_1_invalid_create_saved_action_from_json ()
    test_invalid_create_object_from_json_2_invalid_create_client_from_json ()
    test_invalid_create_object_from_json_3_invalid_create_userrole_from_json ()
    test_invalid_create_object_from_json_4_invalid_create_setting_from_json ()
    test_invalid_deploy_action_1_invalid_deploy_action_run_false ()
    test_invalid_deploy_action_2_invalid_deploy_action_package_help ()
    test_invalid_deploy_action_3_invalid_deploy_action_package ()
    test_invalid_deploy_action_4_invalid_deploy_action_options_help ()
    test_invalid_deploy_action_5_invalid_deploy_action_empty_package ()
    test_invalid_deploy_action_6_invalid_deploy_action_filters_help ()
    test_invalid_deploy_action_7_invalid_deploy_action_missing_parameters ()
    test_invalid_export_basetype_1_invalid_export_basetype_csv_bad_explode_type ()

```

```
test_invalid_export_basetype_2_invalid_export_basetype_csv_bad_sort_sub_type()
test_invalid_export_basetype_3_invalid_export_basetype_csv_bad_sort_type()
test_invalid_export_basetype_4_invalid_export_basetype_xml_bad_minimal_type()
test_invalid_export_basetype_5_invalid_export_basetype_json_bad_include_type()
test_invalid_export_basetype_6_invalid_export_basetype_json_bad_explode_type()
test_invalid_export_basetype_7_invalid_export_basetype_bad_format()
test_invalid_export_resultset_1_invalid_export_resultset_csv_bad_sort_sub_type()
test_invalid_export_resultset_2_invalid_export_resultset_csv_bad_sort_type()
test_invalid_export_resultset_3_invalid_export_resultset_csv_bad_expand_type()
test_invalid_export_resultset_4_invalid_export_resultset_csv_bad_sensors_sub_type()
test_invalid_export_resultset_5_invalid_export_resultset_bad_format()
test_invalid_get_object_1_invalid_get_action_single_by_name()
test_invalid_get_object_2_invalid_get_question_by_name()
test_invalid_question_1_invalid_ask_manual_question_sensor_help()
test_invalid_question_2_invalid_ask_manual_question_bad_filter()
test_invalid_question_3_invalid_ask_manual_question_filter_help()
test_invalid_question_4_invalid_ask_manual_question_bad_option()
test_invalid_question_5_invalid_ask_manual_question_missing_parameter_split()
test_invalid_question_6_invalid_ask_manual_question_option_help()
test_invalid_question_7_invalid_ask_manual_question_too_many_parameter_blocks()
test_invalid_question_8_invalid_ask_manual_question_bad_sensorname()
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_options()
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_parameters()
test_valid_question_2_ask_manual_question_multiple_sensors_with_parameters_and_some_supplied_parameters()
test_valid_question_3_ask_manual_question_simple_multiple_sensors()
test_valid_question_4_ask_manual_question_sensor_without_parameters_and_supplied_parameters()
test_valid_question_5_ask_manual_question_sensor_with_filter_and_2_options()
test_valid_question_6_ask_manual_question_sensor_with_parameters_and_filter()
test_valid_question_7_ask_manual_question_sensor_complex()
test_valid_question_8_ask_manual_question_sensor_with_parameters_and_no_supplied_parameters()
```

```

    test_valid_question_9_ask_manual_question_simple_single_sensor()
    test_valid_saved_question_1_ask_saved_question_refresh_data()
    test_valid_saved_question_2_ask_saved_question_by_name()
    test_valid_saved_question_3_ask_saved_question_by_name_in_list()
test_pytan_valid_server_tests.chew_csv(c)
test_pytan_valid_server_tests.spew(m, l=3)

```

This contains invalid functional tests for pytan.

These functional tests require a connection to a Tanium server in order to run. The connection info is pulled from the SERVER\_INFO dictionary in test/API\_INFO.py.

These tests all use *ddt*, a package that provides for data driven tests via JSON files.

```

class test_pytan_invalid_server_tests.InvalidServerTests (methodName='runTest')
    Bases: unittest.case.TestCase

    classmethod setUpClass()

    test_invalid_connect_1_bad_username()

    test_invalid_connect_2_bad_host_and_non_ssl_port()

    test_invalid_connect_3_bad_password()

    test_invalid_connect_4_bad_host_and_bad_port()

test_pytan_invalid_server_tests.spew(m, l=3)

```

## 1.2.11 PyTan API Examples

Each of these sections contains examples that show Example Python code for using a PyTan method, along with the standard output and standard error from running each example

### PyTan API Basic Handler Example

This 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.

- STDOUT from Example Python Code
- STDERR from Example Python Code
- Example Python Code

```

1  # import the basic python packages we need
2  import os
3  import sys
4  import tempfile
5  import pprint
6  import traceback
7
8  # disable python from generating a .pyc file
9  sys.dont_write_bytecode = True
10
11 # change me to the path of pytan if this script is not running from EXAMPLES/PYTAN_API
12 pytan_loc = "~/gh/pytan"

```

```
13 pytan_static_path = os.path.join(os.path.expanduser(pytan_loc), 'lib')
14
15 # Determine our script name, script dir
16 my_file = os.path.abspath(sys.argv[0])
17 my_dir = os.path.dirname(my_file)
18
19 # try to automatically determine the pytan lib directory by assuming it is in '../..lib/'
20 parent_dir = os.path.dirname(my_dir)
21 pytan_root_dir = os.path.dirname(parent_dir)
22 lib_dir = os.path.join(pytan_root_dir, 'lib')
23
24 # add pytan_loc and lib_dir to the PYTHONPATH variable
25 path_adds = [lib_dir, pytan_static_path]
26 [sys.path.append(aa) for aa in path_adds if aa not in sys.path]
27
28 # import pytan
29 import pytan
30
31 # create a dictionary of arguments for the pytan handler
32 handler_args = {}
33
34 # establish our connection info for the Tanium Server
35 handler_args['username'] = "Administrator"
36 handler_args['password'] = "Tanium2015!"
37 handler_args['host'] = "10.0.1.240"
38 handler_args['port'] = "443" # optional
39
40 # optional, level 0 is no output except warnings/errors
41 # level 1 through 12 are more and more verbose
42 handler_args['loglevel'] = 1
43
44 # optional, use a debug format for the logging output (uses two lines per log entry)
45 handler_args['debugformat'] = False
46
47 # optional, this saves all response objects to handler.session.ALL_REQUESTS_RESPONSES
48 # very useful for capturing the full exchange of XML requests and responses
49 handler_args['record_all_requests'] = True
50
51 # instantiate a handler using all of the arguments in the handler_args dictionary
52 print "...CALLING: pytan.handler() with args: {}".format(handler_args)
53 handler = pytan.Handler(**handler_args)
54
55 # print out the handler string
56 print "...OUTPUT: handler string: {}".format(handler)
```

## PyTan API Valid Create Object Examples

All of the PyTan API examples for Valid Create Object

### Create User

Create a user called API Test User

- STDOUT from Example Python Code
- STDERR from Example Python Code



- Example Python Code

```

1  # import the basic python packages we need
2  import os
3  import sys
4  import tempfile
5  import pprint
6  import traceback
7
8  # disable python from generating a .pyc file
9  sys.dont_write_bytecode = True
10
11 # change me to the path of pytan if this script is not running from EXAMPLES/PYTAN_API
12 pytan_loc = "~/gh/pytan"
13 pytan_static_path = os.path.join(os.path.expanduser(pytan_loc), 'lib')
14
15 # Determine our script name, script dir
16 my_file = os.path.abspath(sys.argv[0])
17 my_dir = os.path.dirname(my_file)
18
19 # try to automatically determine the pytan lib directory by assuming it is in '../..lib/'
20 parent_dir = os.path.dirname(my_dir)
21 pytan_root_dir = os.path.dirname(parent_dir)
22 lib_dir = os.path.join(pytan_root_dir, 'lib')
23
24 # add pytan_loc and lib_dir to the PYTHONPATH variable
25 path_adds = [lib_dir, pytan_static_path]
26 [sys.path.append(aa) for aa in path_adds if aa not in sys.path]
27
28 # import pytan
29 import pytan
30
31 # create a dictionary of arguments for the pytan handler
32 handler_args = {}
33
34 # establish our connection info for the Tanium Server
35 handler_args['username'] = "Administrator"
36 handler_args['password'] = "Tanium2015!"
37 handler_args['host'] = "10.0.1.240"
38 handler_args['port'] = "443" # optional
39
40 # optional, level 0 is no output except warnings/errors
41 # level 1 through 12 are more and more verbose
42 handler_args['loglevel'] = 1
43
44 # optional, use a debug format for the logging output (uses two lines per log entry)
45 handler_args['debugformat'] = False
46
47 # optional, this saves all response objects to handler.session.ALL_REQUESTS_RESPONSES
48 # very useful for capturing the full exchange of XML requests and responses
49 handler_args['record_all_requests'] = True
50
51 # instantiate a handler using all of the arguments in the handler_args dictionary
52 print "...CALLING: pytan.handler() with args: {}".format(handler_args)
53 handler = pytan.Handler(**handler_args)
54
55 # print out the handler string
56 print "...OUTPUT: handler string: {}".format(handler)
57

```

```
58 # setup the arguments for the handler.delete() method
59 delete_kwargs = {}
60 delete_kwargs["objtype"] = u'user'
61 delete_kwargs["name"] = u'API Test User'
62
63 # setup the arguments for the handler() class
64 kwargs = {}
65 kwargs["rolename"] = u'Administrator'
66 kwargs["name"] = u'API Test User'
67 kwargs["properties"] = [[u'property1', u'value1']]
68
69 # delete the object in case it already exists
70 # catch and print the exception error if it does not exist
71 print "...CALLING: handler.delete() with args: {}".format(delete_kwargs)
72 try:
73     handler.delete(**delete_kwargs)
74 except Exception as e:
75     print "...EXCEPTION: {}".format(e)
76
77 print "...CALLING: handler.create_user() with args: {}".format(kwargs)
78 response = handler.create_user(**kwargs)
79
80 print "...OUTPUT: Type of response: ", type(response)
81 print "...OUTPUT: print of response:"
82 print response
83
84 # call the export_obj() method to convert response to JSON and store it in out
85 export_kwargs = {}
86 export_kwargs['obj'] = response
87 export_kwargs['export_format'] = 'json'
88 print "...CALLING: handler.export_obj() with args {}".format(export_kwargs)
89 out = handler.export_obj(**export_kwargs)
90
91 # trim the output if it is more than 15 lines long
92 if len(out.splitlines()) > 15:
93     out = out.splitlines()[0:15]
94     out.append('..trimmed for brevity..')
95     out = '\n'.join(out)
96
97 print "...OUTPUT: print the objects returned in JSON format:"
98 print out
99
100 # delete the object, we are done with it now
101 print "...CALLING: handler.delete() with args: {}".format(delete_kwargs)
102 delete_response = handler.delete(**delete_kwargs)
103
104 print "...OUTPUT: print the delete response"
105 print delete_response
```

## Create Package

Create a package called package49

- STDOUT from Example Python Code
- STDERR from Example Python Code
- Example Python Code

```

1  # import the basic python packages we need
2  import os
3  import sys
4  import tempfile
5  import pprint
6  import traceback
7
8  # disable python from generating a .pyc file
9  sys.dont_write_bytecode = True
10
11  # change me to the path of pytan if this script is not running from EXAMPLES/PYTAN_API
12  pytan_loc = "~/gh/pytan"
13  pytan_static_path = os.path.join(os.path.expanduser(pytan_loc), 'lib')
14
15  # Determine our script name, script dir
16  my_file = os.path.abspath(sys.argv[0])
17  my_dir = os.path.dirname(my_file)
18
19  # try to automatically determine the pytan lib directory by assuming it is in '../..lib/'
20  parent_dir = os.path.dirname(my_dir)
21  pytan_root_dir = os.path.dirname(parent_dir)
22  lib_dir = os.path.join(pytan_root_dir, 'lib')
23
24  # add pytan_loc and lib_dir to the PYTHONPATH variable
25  path_adds = [lib_dir, pytan_static_path]
26  [sys.path.append(aa) for aa in path_adds if aa not in sys.path]
27
28  # import pytan
29  import pytan
30
31  # create a dictionary of arguments for the pytan handler
32  handler_args = {}
33
34  # establish our connection info for the Tanium Server
35  handler_args['username'] = "Administrator"
36  handler_args['password'] = "Tanium2015!"
37  handler_args['host'] = "10.0.1.240"
38  handler_args['port'] = "443" # optional
39
40  # optional, level 0 is no output except warnings/errors
41  # level 1 through 12 are more and more verbose
42  handler_args['loglevel'] = 1
43
44  # optional, use a debug format for the logging output (uses two lines per log entry)
45  handler_args['debugformat'] = False
46
47  # optional, this saves all response objects to handler.session.ALL_REQUESTS_RESPONSES
48  # very useful for capturing the full exchange of XML requests and responses
49  handler_args['record_all_requests'] = True
50
51  # instantiate a handler using all of the arguments in the handler_args dictionary
52  print "...CALLING: pytan.handler() with args: {}".format(handler_args)
53  handler = pytan.Handler(**handler_args)
54
55  # print out the handler string
56  print "...OUTPUT: handler string: {}".format(handler)
57
58  # setup the arguments for the handler.delete() method

```

```
59 delete_kwargs = {}
60 delete_kwargs["objtype"] = u'package'
61 delete_kwargs["name"] = u'package49'
62
63 # setup the arguments for the handler() class
64 kwargs = {}
65 kwargs["expire_seconds"] = 1500
66 kwargs["display_name"] = u'package49 API test'
67 kwargs["name"] = u'package49'
68 kwargs["parameters_json_file"] = u'../doc/example_of_all_package_parameters.json'
69 kwargs["verify_expire_seconds"] = 3600
70 kwargs["command"] = u'package49 $1 $2 $3 $4 $5 $6 $7 $8'
71 kwargs["file_urls"] = [u'3600::testing.vbs|https://content.tanium.com/files/initialcontent/bundles/2
72 kwargs["verify_filter_options"] = [u'and']
73 kwargs["verify_filters"] = [u'Custom Tags, that contains:tag']
74 kwargs["command_timeout_seconds"] = 9999
75
76 # delete the object in case it already exists
77 # catch and print the exception error if it does not exist
78 print "...CALLING: handler.delete() with args: {}".format(delete_kwargs)
79 try:
80     handler.delete(**delete_kwargs)
81 except Exception as e:
82     print "...EXCEPTION: {}".format(e)
83
84 print "...CALLING: handler.create_package() with args: {}".format(kwargs)
85 response = handler.create_package(**kwargs)
86
87 print "...OUTPUT: Type of response: ", type(response)
88 print "...OUTPUT: print of response:"
89 print response
90
91 # call the export_obj() method to convert response to JSON and store it in out
92 export_kwargs = {}
93 export_kwargs['obj'] = response
94 export_kwargs['export_format'] = 'json'
95 print "...CALLING: handler.export_obj() with args {}".format(export_kwargs)
96 out = handler.export_obj(**export_kwargs)
97
98 # trim the output if it is more than 15 lines long
99 if len(out.splitlines()) > 15:
100     out = out.splitlines()[0:15]
101     out.append('..trimmed for brevity..')
102     out = '\n'.join(out)
103
104 print "...OUTPUT: print the objects returned in JSON format:"
105 print out
106
107 # delete the object, we are done with it now
108 print "...CALLING: handler.delete() with args: {}".format(delete_kwargs)
109 delete_response = handler.delete(**delete_kwargs)
110
111 print "...OUTPUT: print the delete response"
112 print delete_response
```

## Create Group

Create a group called All Windows Computers API Test

- STDOUT from Example Python Code
- STDERR from Example Python Code
- Example Python Code

```

1  # import the basic python packages we need
2  import os
3  import sys
4  import tempfile
5  import pprint
6  import traceback
7
8  # disable python from generating a .pyc file
9  sys.dont_write_bytecode = True
10
11 # change me to the path of pytan if this script is not running from EXAMPLES/PYTAN_API
12 pytan_loc = "~/gh/pytan"
13 pytan_static_path = os.path.join(os.path.expanduser(pytan_loc), 'lib')
14
15 # Determine our script name, script dir
16 my_file = os.path.abspath(sys.argv[0])
17 my_dir = os.path.dirname(my_file)
18
19 # try to automatically determine the pytan lib directory by assuming it is in '../..lib/'
20 parent_dir = os.path.dirname(my_dir)
21 pytan_root_dir = os.path.dirname(parent_dir)
22 lib_dir = os.path.join(pytan_root_dir, 'lib')
23
24 # add pytan_loc and lib_dir to the PYTHONPATH variable
25 path_adds = [lib_dir, pytan_static_path]
26 [sys.path.append(aa) for aa in path_adds if aa not in sys.path]
27
28 # import pytan
29 import pytan
30
31 # create a dictionary of arguments for the pytan handler
32 handler_args = {}
33
34 # establish our connection info for the Tanium Server
35 handler_args['username'] = "Administrator"
36 handler_args['password'] = "Tanium2015!"
37 handler_args['host'] = "10.0.1.240"
38 handler_args['port'] = "443" # optional
39
40 # optional, level 0 is no output except warnings/errors
41 # level 1 through 12 are more and more verbose
42 handler_args['loglevel'] = 1
43
44 # optional, use a debug format for the logging output (uses two lines per log entry)
45 handler_args['debugformat'] = False
46
47 # optional, this saves all response objects to handler.session.ALL_REQUESTS_RESPONSES
48 # very useful for capturing the full exchange of XML requests and responses
49 handler_args['record_all_requests'] = True

```

```
50
51 # instantiate a handler using all of the arguments in the handler_args dictionary
52 print "...CALLING: pytan.handler() with args: {}".format(handler_args)
53 handler = pytan.Handler(**handler_args)
54
55 # print out the handler string
56 print "...OUTPUT: handler string: {}".format(handler)
57
58 # setup the arguments for the handler.delete() method
59 delete_kwargs = {}
60 delete_kwargs["objtype"] = u'group'
61 delete_kwargs["name"] = u'All Windows Computers API Test'
62
63 # setup the arguments for the handler() class
64 kwargs = {}
65 kwargs["groupname"] = u'All Windows Computers API Test'
66 kwargs["filters"] = [u'Operating System, that contains:Windows']
67 kwargs["filter_options"] = [u'and']
68
69 # delete the object in case it already exists
70 # catch and print the exception error if it does not exist
71 print "...CALLING: handler.delete() with args: {}".format(delete_kwargs)
72 try:
73     handler.delete(**delete_kwargs)
74 except Exception as e:
75     print "...EXCEPTION: {}".format(e)
76
77 print "...CALLING: handler.create_group() with args: {}".format(kwargs)
78 response = handler.create_group(**kwargs)
79
80 print "...OUTPUT: Type of response: ", type(response)
81 print "...OUTPUT: print of response:"
82 print response
83
84 # call the export_obj() method to convert response to JSON and store it in out
85 export_kwargs = {}
86 export_kwargs['obj'] = response
87 export_kwargs['export_format'] = 'json'
88 print "...CALLING: handler.export_obj() with args {}".format(export_kwargs)
89 out = handler.export_obj(**export_kwargs)
90
91 # trim the output if it is more than 15 lines long
92 if len(out.splitlines()) > 15:
93     out = out.splitlines()[0:15]
94     out.append('..trimmed for brevity..')
95     out = '\n'.join(out)
96
97 print "...OUTPUT: print the objects returned in JSON format:"
98 print out
99
100 # delete the object, we are done with it now
101 print "...CALLING: handler.delete() with args: {}".format(delete_kwargs)
102 delete_response = handler.delete(**delete_kwargs)
103
104 print "...OUTPUT: print the delete response"
105 print delete_response
```

## Create Whitelisted Url

Create a whitelisted url

- STDOUT from Example Python Code
- STDERR from Example Python Code
- Example Python Code

```

1  # import the basic python packages we need
2  import os
3  import sys
4  import tempfile
5  import pprint
6  import traceback
7
8  # disable python from generating a .pyc file
9  sys.dont_write_bytecode = True
10
11 # change me to the path of pytan if this script is not running from EXAMPLES/PYTAN_API
12 pytan_loc = "~/gh/pytan"
13 pytan_static_path = os.path.join(os.path.expanduser(pytan_loc), 'lib')
14
15 # Determine our script name, script dir
16 my_file = os.path.abspath(sys.argv[0])
17 my_dir = os.path.dirname(my_file)
18
19 # try to automatically determine the pytan lib directory by assuming it is in '../..lib/'
20 parent_dir = os.path.dirname(my_dir)
21 pytan_root_dir = os.path.dirname(parent_dir)
22 lib_dir = os.path.join(pytan_root_dir, 'lib')
23
24 # add pytan_loc and lib_dir to the PYTHONPATH variable
25 path_adds = [lib_dir, pytan_static_path]
26 [sys.path.append(aa) for aa in path_adds if aa not in sys.path]
27
28 # import pytan
29 import pytan
30
31 # create a dictionary of arguments for the pytan handler
32 handler_args = {}
33
34 # establish our connection info for the Tanium Server
35 handler_args['username'] = "Administrator"
36 handler_args['password'] = "Tanium2015!"
37 handler_args['host'] = "10.0.1.240"
38 handler_args['port'] = "443" # optional
39
40 # optional, level 0 is no output except warnings/errors
41 # level 1 through 12 are more and more verbose
42 handler_args['loglevel'] = 1
43
44 # optional, use a debug format for the logging output (uses two lines per log entry)
45 handler_args['debugformat'] = False
46
47 # optional, this saves all response objects to handler.session.ALL_REQUESTS_RESPONSES
48 # very useful for capturing the full exchange of XML requests and responses
49 handler_args['record_all_requests'] = True

```

```
50
51 # instantiate a handler using all of the arguments in the handler_args dictionary
52 print "...CALLING: pytan.handler() with args: {}".format(handler_args)
53 handler = pytan.Handler(**handler_args)
54
55 # print out the handler string
56 print "...OUTPUT: handler string: {}".format(handler)
57
58 # setup the arguments for the handler.delete() method
59 delete_kwargs = {}
60 delete_kwargs["objtype"] = u'whitelisted_url'
61 delete_kwargs["url_regex"] = u'regex:http://test.com/*.API_Test.*URL'
62
63 # setup the arguments for the handler() class
64 kwargs = {}
65 kwargs["url"] = u'http://test.com/*.API_Test.*URL'
66 kwargs["regex"] = True
67 kwargs["properties"] = [[u'property1', u'value1']]
68 kwargs["download_seconds"] = 3600
69
70 # delete the object in case it already exists
71 # catch and print the exception error if it does not exist
72 print "...CALLING: handler.delete() with args: {}".format(delete_kwargs)
73 try:
74     handler.delete(**delete_kwargs)
75 except Exception as e:
76     print "...EXCEPTION: {}".format(e)
77
78 print "...CALLING: handler.create_whitelisted_url() with args: {}".format(kwargs)
79 response = handler.create_whitelisted_url(**kwargs)
80
81 print "...OUTPUT: Type of response: ", type(response)
82 print "...OUTPUT: print of response:"
83 print response
84
85 # call the export_obj() method to convert response to JSON and store it in out
86 export_kwargs = {}
87 export_kwargs['obj'] = response
88 export_kwargs['export_format'] = 'json'
89 print "...CALLING: handler.export_obj() with args {}".format(export_kwargs)
90 out = handler.export_obj(**export_kwargs)
91
92 # trim the output if it is more than 15 lines long
93 if len(out.splitlines()) > 15:
94     out = out.splitlines()[0:15]
95     out.append('...trimmed for brevity..')
96     out = '\n'.join(out)
97
98 print "...OUTPUT: print the objects returned in JSON format:"
99 print out
100
101 # delete the object, we are done with it now
102 print "...CALLING: handler.delete() with args: {}".format(delete_kwargs)
103 delete_response = handler.delete(**delete_kwargs)
104
105 print "...OUTPUT: print the delete response"
106 print delete_response
```



## PyTan API Valid Create Object From JSON Examples

All of the PyTan API examples for Valid Create Object From JSON

### 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

- STDOUT from Example Python Code
- STDERR from Example Python Code
- Example Python Code

```

1  # import the basic python packages we need
2  import os
3  import sys
4  import tempfile
5  import pprint
6  import traceback
7
8  # disable python from generating a .pyc file
9  sys.dont_write_bytecode = True
10
11 # change me to the path of pytan if this script is not running from EXAMPLES/PYTAN_API
12 pytan_loc = "~/gh/pytan"
13 pytan_static_path = os.path.join(os.path.expanduser(pytan_loc), 'lib')
14
15 # Determine our script name, script dir
16 my_file = os.path.abspath(sys.argv[0])
17 my_dir = os.path.dirname(my_file)
18
19 # try to automatically determine the pytan lib directory by assuming it is in '../..lib/'
20 parent_dir = os.path.dirname(my_dir)
21 pytan_root_dir = os.path.dirname(parent_dir)
22 lib_dir = os.path.join(pytan_root_dir, 'lib')
23
24 # add pytan_loc and lib_dir to the PYTHONPATH variable
25 path_adds = [lib_dir, pytan_static_path]
26 [sys.path.append(aa) for aa in path_adds if aa not in sys.path]
27
28 # import pytan
29 import pytan
30
31 # create a dictionary of arguments for the pytan handler
32 handler_args = {}
33
34 # establish our connection info for the Tanium Server
35 handler_args['username'] = "Administrator"
36 handler_args['password'] = "Tanium2015!"
37 handler_args['host'] = "10.0.1.240"
38 handler_args['port'] = "443" # optional
39
40 # optional, level 0 is no output except warnings/errors
41 # level 1 through 12 are more and more verbose
42 handler_args['loglevel'] = 1

```

```
43 # optional, use a debug format for the logging output (uses two lines per log entry)
44 handler_args['debugformat'] = False
45
46 # optional, this saves all response objects to handler.session.ALL_REQUESTS_RESPONSES
47 # very useful for capturing the full exchange of XML requests and responses
48 handler_args['record_all_requests'] = True
49
50 # instantiate a handler using all of the arguments in the handler_args dictionary
51 print "...CALLING: pytan.handler() with args: {}".format(handler_args)
52 handler = pytan.Handler(**handler_args)
53
54 # print out the handler string
55 print "...OUTPUT: handler string: {}".format(handler)
56
57 # setup the arguments for the handler.get() method
58 get_kwargs = {}
59 get_kwargs["objtype"] = u'package'
60 get_kwargs["id"] = 31
61
62 # get objects to use as an export to JSON file
63 print "...CALLING: handler.get() with args: {}".format(get_kwargs)
64 orig_objs = handler.get(**get_kwargs)
65
66 # set the attribute name and value we want to add to the original objects
67 # this is necessary to avoid name conflicts when adding the new object
68 attr_name = u'name'
69 attr_value = u' API TEST'
70
71 # modify the orig_objs to add attr_value to attr_name
72 for x in orig_objs:
73     new_attr = getattr(x, attr_name)
74     new_attr += attr_value
75     setattr(x, attr_name, new_attr)
76
77 # delete the object in case it already exists
78 del_kwargs = {}
79 del_kwargs[attr_name] = new_attr
80 del_kwargs['objtype'] = u'package'
81
82 print "...CALLING: handler.delete() with args: {}".format(del_kwargs)
83 try:
84     handler.delete(**del_kwargs)
85 except Exception as e:
86     print "...EXCEPTION: {}".format(e)
87
88 # export orig_objs to a json file
89 export_kwargs = {}
90 export_kwargs['obj'] = orig_objs
91 export_kwargs['export_format'] = 'json'
92 export_kwargs['report_dir'] = tempfile.gettempdir()
93
94 print "...CALLING: handler.export_to_report_file() with args: {}".format(export_kwargs)
95 json_file, results = handler.export_to_report_file(**export_kwargs)
96
97 # create the object from the exported JSON file
98 create_kwargs = {}
99 create_kwargs['objtype'] = u'package'
```

```

101 create_kwargs['json_file'] = json_file
102
103 print "...CALLING: handler.create_from_json() with args {}".format(create_kwargs)
104 response = handler.create_from_json(**create_kwargs)
105
106 print "...OUTPUT: Type of response: ", type(response)
107
108 print "...OUTPUT: print of response:"
109 print response
110
111 # call the export_obj() method to convert response to JSON and store it in out
112 export_kwargs = {}
113 export_kwargs['obj'] = response
114 export_kwargs['export_format'] = 'json'
115
116 print "...CALLING: handler.export_obj() with args {}".format(export_kwargs)
117 out = handler.export_obj(**export_kwargs)
118
119 # trim the output if it is more than 15 lines long
120 if len(out.splitlines()) > 15:
121     out = out.splitlines()[0:15]
122     out.append('..trimmed for brevity..')
123     out = '\n'.join(out)
124
125 print "...OUTPUT: print the objects returned in JSON format:"
126 print out

```

## 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

- STDOUT from Example Python Code
- STDERR from Example Python Code
- Example Python Code

```

1  # import the basic python packages we need
2  import os
3  import sys
4  import tempfile
5  import pprint
6  import traceback
7
8  # disable python from generating a .pyc file
9  sys.dont_write_bytecode = True
10
11 # change me to the path of pytan if this script is not running from EXAMPLES/PYTAN_API
12 pytan_loc = "~/gh/pytan"
13 pytan_static_path = os.path.join(os.path.expanduser(pytan_loc), 'lib')
14
15 # Determine our script name, script dir
16 my_file = os.path.abspath(sys.argv[0])
17 my_dir = os.path.dirname(my_file)
18
19 # try to automatically determine the pytan lib directory by assuming it is in '../..lib/'
20 parent_dir = os.path.dirname(my_dir)

```

```
21 pytan_root_dir = os.path.dirname(parent_dir)
22 lib_dir = os.path.join(pytan_root_dir, 'lib')
23
24 # add pytan_loc and lib_dir to the PYTHONPATH variable
25 path_adds = [lib_dir, pytan_static_path]
26 [sys.path.append(aa) for aa in path_adds if aa not in sys.path]
27
28 # import pytan
29 import pytan
30
31 # create a dictionary of arguments for the pytan handler
32 handler_args = {}
33
34 # establish our connection info for the Tanium Server
35 handler_args['username'] = "Administrator"
36 handler_args['password'] = "Tanium2015!"
37 handler_args['host'] = "10.0.1.240"
38 handler_args['port'] = "443" # optional
39
40 # optional, level 0 is no output except warnings/errors
41 # level 1 through 12 are more and more verbose
42 handler_args['loglevel'] = 1
43
44 # optional, use a debug format for the logging output (uses two lines per log entry)
45 handler_args['debugformat'] = False
46
47 # optional, this saves all response objects to handler.session.ALL_REQUESTS_RESPONSES
48 # very useful for capturing the full exchange of XML requests and responses
49 handler_args['record_all_requests'] = True
50
51 # instantiate a handler using all of the arguments in the handler_args dictionary
52 print "...CALLING: pytan.handler() with args: {}".format(handler_args)
53 handler = pytan.Handler(**handler_args)
54
55 # print out the handler string
56 print "...OUTPUT: handler string: {}".format(handler)
57
58 # setup the arguments for the handler.get() method
59 get_kwargs = {}
60 get_kwargs["objtype"] = u'user'
61 get_kwargs["id"] = 1
62
63 # get objects to use as an export to JSON file
64 print "...CALLING: handler.get() with args: {}".format(get_kwargs)
65 orig_objs = handler.get(**get_kwargs)
66
67 # set the attribute name and value we want to add to the original objects
68 # this is necessary to avoid name conflicts when adding the new object
69 attr_name = u'name'
70 attr_value = u' API TEST'
71
72 # modify the orig_objs to add attr_value to attr_name
73 for x in orig_objs:
74     new_attr = getattr(x, attr_name)
75     new_attr += attr_value
76     setattr(x, attr_name, new_attr)
77
78     # delete the object in case it already exists
```

```

79     del_kwargs = {}
80     del_kwargs[attr_name] = new_attr
81     del_kwargs['objtype'] = u'user'
82
83     print "...CALLING: handler.delete() with args: {}".format(del_kwargs)
84     try:
85         handler.delete(**del_kwargs)
86     except Exception as e:
87         print "...EXCEPTION: {}".format(e)
88
89     # export orig_objs to a json file
90     export_kwargs = {}
91     export_kwargs['obj'] = orig_objs
92     export_kwargs['export_format'] = 'json'
93     export_kwargs['report_dir'] = tempfile.gettempdir()
94
95     print "...CALLING: handler.export_to_report_file() with args: {}".format(export_kwargs)
96     json_file, results = handler.export_to_report_file(**export_kwargs)
97
98     # create the object from the exported JSON file
99     create_kwargs = {}
100    create_kwargs['objtype'] = u'user'
101    create_kwargs['json_file'] = json_file
102
103    print "...CALLING: handler.create_from_json() with args {}".format(create_kwargs)
104    response = handler.create_from_json(**create_kwargs)
105
106    print "...OUTPUT: Type of response: ", type(response)
107
108    print "...OUTPUT: print of response:"
109    print response
110
111    # call the export_obj() method to convert response to JSON and store it in out
112    export_kwargs = {}
113    export_kwargs['obj'] = response
114    export_kwargs['export_format'] = 'json'
115
116    print "...CALLING: handler.export_obj() with args {}".format(export_kwargs)
117    out = handler.export_obj(**export_kwargs)
118
119    # trim the output if it is more than 15 lines long
120    if len(out.splitlines()) > 15:
121        out = out.splitlines()[0:15]
122        out.append('..trimmed for brevity..')
123        out = '\n'.join(out)
124
125    print "...OUTPUT: print the objects returned in JSON format:"
126    print out

```

### 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

- STDOUT from Example Python Code

- STDERR from Example Python Code
- Example Python Code

```
1  # import the basic python packages we need
2  import os
3  import sys
4  import tempfile
5  import pprint
6  import traceback
7
8  # disable python from generating a .pyc file
9  sys.dont_write_bytecode = True
10
11  # change me to the path of pytan if this script is not running from EXAMPLES/PYTAN_API
12  pytan_loc = "~/gh/pytan"
13  pytan_static_path = os.path.join(os.path.expanduser(pytan_loc), 'lib')
14
15  # Determine our script name, script dir
16  my_file = os.path.abspath(sys.argv[0])
17  my_dir = os.path.dirname(my_file)
18
19  # try to automatically determine the pytan lib directory by assuming it is in '../..lib/'
20  parent_dir = os.path.dirname(my_dir)
21  pytan_root_dir = os.path.dirname(parent_dir)
22  lib_dir = os.path.join(pytan_root_dir, 'lib')
23
24  # add pytan_loc and lib_dir to the PYTHONPATH variable
25  path_adds = [lib_dir, pytan_static_path]
26  [sys.path.append(aa) for aa in path_adds if aa not in sys.path]
27
28  # import pytan
29  import pytan
30
31  # create a dictionary of arguments for the pytan handler
32  handler_args = {}
33
34  # establish our connection info for the Tanium Server
35  handler_args['username'] = "Administrator"
36  handler_args['password'] = "Tanium2015!"
37  handler_args['host'] = "10.0.1.240"
38  handler_args['port'] = "443" # optional
39
40  # optional, level 0 is no output except warnings/errors
41  # level 1 through 12 are more and more verbose
42  handler_args['loglevel'] = 1
43
44  # optional, use a debug format for the logging output (uses two lines per log entry)
45  handler_args['debugformat'] = False
46
47  # optional, this saves all response objects to handler.session.ALL_REQUESTS_RESPONSES
48  # very useful for capturing the full exchange of XML requests and responses
49  handler_args['record_all_requests'] = True
50
51  # instantiate a handler using all of the arguments in the handler_args dictionary
52  print "...CALLING: pytan.handler() with args: {}".format(handler_args)
53  handler = pytan.Handler(**handler_args)
54
55  # print out the handler string
```

```

56 print "...OUTPUT: handler string: {}".format(handler)
57
58 # setup the arguments for the handler.get() method
59 get_kwargs = {}
60 get_kwargs["objtype"] = u'saved_question'
61 get_kwargs["id"] = 1
62
63 # get objects to use as an export to JSON file
64 print "...CALLING: handler.get() with args: {}".format(get_kwargs)
65 orig_objs = handler.get(**get_kwargs)
66
67 # set the attribute name and value we want to add to the original objects
68 # this is necessary to avoid name conflicts when adding the new object
69 attr_name = u'name'
70 attr_value = u' API TEST'
71
72 # modify the orig_objs to add attr_value to attr_name
73 for x in orig_objs:
74     new_attr = getattr(x, attr_name)
75     new_attr += attr_value
76     setattr(x, attr_name, new_attr)
77
78     # delete the object in case it already exists
79     del_kwargs = {}
80     del_kwargs[attr_name] = new_attr
81     del_kwargs['objtype'] = u'saved_question'
82
83     print "...CALLING: handler.delete() with args: {}".format(del_kwargs)
84     try:
85         handler.delete(**del_kwargs)
86     except Exception as e:
87         print "...EXCEPTION: {}".format(e)
88
89 # export orig_objs to a json file
90 export_kwargs = {}
91 export_kwargs['obj'] = orig_objs
92 export_kwargs['export_format'] = 'json'
93 export_kwargs['report_dir'] = tempfile.gettempdir()
94
95 print "...CALLING: handler.export_to_report_file() with args: {}".format(export_kwargs)
96 json_file, results = handler.export_to_report_file(**export_kwargs)
97
98 # create the object from the exported JSON file
99 create_kwargs = {}
100 create_kwargs['objtype'] = u'saved_question'
101 create_kwargs['json_file'] = json_file
102
103 print "...CALLING: handler.create_from_json() with args {}".format(create_kwargs)
104 response = handler.create_from_json(**create_kwargs)
105
106 print "...OUTPUT: Type of response: ", type(response)
107
108 print "...OUTPUT: print of response:"
109 print response
110
111 # call the export_obj() method to convert response to JSON and store it in out
112 export_kwargs = {}
113 export_kwargs['obj'] = response

```

```
114 export_kwargs['export_format'] = 'json'
115
116 print "...CALLING: handler.export_obj() with args {}".format(export_kwargs)
117 out = handler.export_obj(**export_kwargs)
118
119 # trim the output if it is more than 15 lines long
120 if len(out.splitlines()) > 15:
121     out = out.splitlines()[0:15]
122     out.append('..trimmed for brevity..')
123     out = '\n'.join(out)
124
125 print "...OUTPUT: print the objects returned in JSON format:"
126 print out
```

### 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.

- STDOUT from Example Python Code
- STDERR from Example Python Code
- Example Python Code

```
1  # import the basic python packages we need
2  import os
3  import sys
4  import tempfile
5  import pprint
6  import traceback
7
8  # disable python from generating a .pyc file
9  sys.dont_write_bytecode = True
10
11 # change me to the path of pytan if this script is not running from EXAMPLES/PYTAN_API
12 pytan_loc = "~/gh/pytan"
13 pytan_static_path = os.path.join(os.path.expanduser(pytan_loc), 'lib')
14
15 # Determine our script name, script dir
16 my_file = os.path.abspath(sys.argv[0])
17 my_dir = os.path.dirname(my_file)
18
19 # try to automatically determine the pytan lib directory by assuming it is in '../..lib/'
20 parent_dir = os.path.dirname(my_dir)
21 pytan_root_dir = os.path.dirname(parent_dir)
22 lib_dir = os.path.join(pytan_root_dir, 'lib')
23
24 # add pytan_loc and lib_dir to the PYTHONPATH variable
25 path_adds = [lib_dir, pytan_static_path]
26 [sys.path.append(aa) for aa in path_adds if aa not in sys.path]
27
28 # import pytan
29 import pytan
30
31 # create a dictionary of arguments for the pytan handler
32 handler_args = {}
33
```



```

34 # establish our connection info for the Tanium Server
35 handler_args['username'] = "Administrator"
36 handler_args['password'] = "Tanium2015!"
37 handler_args['host'] = "10.0.1.240"
38 handler_args['port'] = "443" # optional
39
40 # optional, level 0 is no output except warnings/errors
41 # level 1 through 12 are more and more verbose
42 handler_args['loglevel'] = 1
43
44 # optional, use a debug format for the logging output (uses two lines per log entry)
45 handler_args['debugformat'] = False
46
47 # optional, this saves all response objects to handler.session.ALL_REQUESTS_RESPONSES
48 # very useful for capturing the full exchange of XML requests and responses
49 handler_args['record_all_requests'] = True
50
51 # instantiate a handler using all of the arguments in the handler_args dictionary
52 print "...CALLING: pytan.handler() with args: {}".format(handler_args)
53 handler = pytan.Handler(**handler_args)
54
55 # print out the handler string
56 print "...OUTPUT: handler string: {}".format(handler)
57
58 # setup the arguments for the handler.get() method
59 get_kwargs = {}
60 get_kwargs["objtype"] = u'action'
61 get_kwargs["id"] = 1
62
63 # get objects to use as an export to JSON file
64 print "...CALLING: handler.get() with args: {}".format(get_kwargs)
65 orig_objs = handler.get(**get_kwargs)
66
67 # export orig_objs to a json file
68 export_kwargs = {}
69 export_kwargs['obj'] = orig_objs
70 export_kwargs['export_format'] = 'json'
71 export_kwargs['report_dir'] = tempfile.gettempdir()
72
73 print "...CALLING: handler.export_to_report_file() with args: {}".format(export_kwargs)
74 json_file, results = handler.export_to_report_file(**export_kwargs)
75
76 # create the object from the exported JSON file
77 create_kwargs = {}
78 create_kwargs['objtype'] = u'action'
79 create_kwargs['json_file'] = json_file
80
81 print "...CALLING: handler.create_from_json() with args {}".format(create_kwargs)
82 response = handler.create_from_json(**create_kwargs)
83
84 print "...OUTPUT: Type of response: ", type(response)
85
86 print "...OUTPUT: print of response:"
87 print response
88
89 # call the export_obj() method to convert response to JSON and store it in out
90 export_kwargs = {}
91 export_kwargs['obj'] = response

```

```
92 export_kwargs['export_format'] = 'json'
93
94 print "...CALLING: handler.export_obj() with args {}".format(export_kwargs)
95 out = handler.export_obj(**export_kwargs)
96
97 # trim the output if it is more than 15 lines long
98 if len(out.splitlines()) > 15:
99     out = out.splitlines()[0:15]
100     out.append('..trimmed for brevity..')
101     out = '\n'.join(out)
102
103 print "...OUTPUT: print the objects returned in JSON format:"
104 print out
```

### 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

- STDOUT from Example Python Code
- STDERR from Example Python Code
- Example Python Code

```
1  # import the basic python packages we need
2  import os
3  import sys
4  import tempfile
5  import pprint
6  import traceback
7
8  # disable python from generating a .pyc file
9  sys.dont_write_bytecode = True
10
11 # change me to the path of pytan if this script is not running from EXAMPLES/PYTAN_API
12 pytan_loc = "~/gh/pytan"
13 pytan_static_path = os.path.join(os.path.expanduser(pytan_loc), 'lib')
14
15 # Determine our script name, script dir
16 my_file = os.path.abspath(sys.argv[0])
17 my_dir = os.path.dirname(my_file)
18
19 # try to automatically determine the pytan lib directory by assuming it is in '../..lib/'
20 parent_dir = os.path.dirname(my_dir)
21 pytan_root_dir = os.path.dirname(parent_dir)
22 lib_dir = os.path.join(pytan_root_dir, 'lib')
23
24 # add pytan_loc and lib_dir to the PYTHONPATH variable
25 path_adds = [lib_dir, pytan_static_path]
26 [sys.path.append(aa) for aa in path_adds if aa not in sys.path]
27
28 # import pytan
29 import pytan
30
31 # create a dictionary of arguments for the pytan handler
32 handler_args = {}
```

```

33
34 # establish our connection info for the Tanium Server
35 handler_args['username'] = "Administrator"
36 handler_args['password'] = "Tanium2015!"
37 handler_args['host'] = "10.0.1.240"
38 handler_args['port'] = "443" # optional
39
40 # optional, level 0 is no output except warnings/errors
41 # level 1 through 12 are more and more verbose
42 handler_args['loglevel'] = 1
43
44 # optional, use a debug format for the logging output (uses two lines per log entry)
45 handler_args['debugformat'] = False
46
47 # optional, this saves all response objects to handler.session.ALL_REQUESTS_RESPONSES
48 # very useful for capturing the full exchange of XML requests and responses
49 handler_args['record_all_requests'] = True
50
51 # instantiate a handler using all of the arguments in the handler_args dictionary
52 print "...CALLING: pytan.handler() with args: {}".format(handler_args)
53 handler = pytan.Handler(**handler_args)
54
55 # print out the handler string
56 print "...OUTPUT: handler string: {}".format(handler)
57
58 # setup the arguments for the handler.get() method
59 get_kwargs = {}
60 get_kwargs["objtype"] = u'sensor'
61 get_kwargs["id"] = 381
62
63 # get objects to use as an export to JSON file
64 print "...CALLING: handler.get() with args: {}".format(get_kwargs)
65 orig_objs = handler.get(**get_kwargs)
66
67 # set the attribute name and value we want to add to the original objects
68 # this is necessary to avoid name conflicts when adding the new object
69 attr_name = u'name'
70 attr_value = u' API TEST'
71
72 # modify the orig_objs to add attr_value to attr_name
73 for x in orig_objs:
74     new_attr = getattr(x, attr_name)
75     new_attr += attr_value
76     setattr(x, attr_name, new_attr)
77
78 # delete the object in case it already exists
79 del_kwargs = {}
80 del_kwargs[attr_name] = new_attr
81 del_kwargs['objtype'] = u'sensor'
82
83 print "...CALLING: handler.delete() with args: {}".format(del_kwargs)
84 try:
85     handler.delete(**del_kwargs)
86 except Exception as e:
87     print "...EXCEPTION: {}".format(e)
88
89 # export orig_objs to a json file
90 export_kwargs = {}

```

```
91 export_kwargs['obj'] = orig_objs
92 export_kwargs['export_format'] = 'json'
93 export_kwargs['report_dir'] = tempfile.gettempdir()
94
95 print "...CALLING: handler.export_to_report_file() with args: {}".format(export_kwargs)
96 json_file, results = handler.export_to_report_file(**export_kwargs)
97
98 # create the object from the exported JSON file
99 create_kwargs = {}
100 create_kwargs['objtype'] = u'sensor'
101 create_kwargs['json_file'] = json_file
102
103 print "...CALLING: handler.create_from_json() with args {}".format(create_kwargs)
104 response = handler.create_from_json(**create_kwargs)
105
106 print "...OUTPUT: Type of response: ", type(response)
107
108 print "...OUTPUT: print of response:"
109 print response
110
111 # call the export_obj() method to convert response to JSON and store it in out
112 export_kwargs = {}
113 export_kwargs['obj'] = response
114 export_kwargs['export_format'] = 'json'
115
116 print "...CALLING: handler.export_obj() with args {}".format(export_kwargs)
117 out = handler.export_obj(**export_kwargs)
118
119 # trim the output if it is more than 15 lines long
120 if len(out.splitlines()) > 15:
121     out = out.splitlines()[0:15]
122     out.append('..trimmed for brevity..')
123     out = '\n'.join(out)
124
125 print "...OUTPUT: print the objects returned in JSON format:"
126 print out
```

## 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.

- STDOUT from Example Python Code
- STDERR from Example Python Code
- Example Python Code

```
1 # import the basic python packages we need
2 import os
3 import sys
4 import tempfile
5 import pprint
6 import traceback
7
8 # disable python from generating a .pyc file
9 sys.dont_write_bytecode = True
10
```

```

11 # change me to the path of pytan if this script is not running from EXAMPLES/PYTAN_API
12 pytan_loc = "~/gh/pytan"
13 pytan_static_path = os.path.join(os.path.expanduser(pytan_loc), 'lib')
14
15 # Determine our script name, script dir
16 my_file = os.path.abspath(sys.argv[0])
17 my_dir = os.path.dirname(my_file)
18
19 # try to automatically determine the pytan lib directory by assuming it is in '../..lib/'
20 parent_dir = os.path.dirname(my_dir)
21 pytan_root_dir = os.path.dirname(parent_dir)
22 lib_dir = os.path.join(pytan_root_dir, 'lib')
23
24 # add pytan_loc and lib_dir to the PYTHONPATH variable
25 path_adds = [lib_dir, pytan_static_path]
26 [sys.path.append(aa) for aa in path_adds if aa not in sys.path]
27
28 # import pytan
29 import pytan
30
31 # create a dictionary of arguments for the pytan handler
32 handler_args = {}
33
34 # establish our connection info for the Tanium Server
35 handler_args['username'] = "Administrator"
36 handler_args['password'] = "Tanium2015!"
37 handler_args['host'] = "10.0.1.240"
38 handler_args['port'] = "443" # optional
39
40 # optional, level 0 is no output except warnings/errors
41 # level 1 through 12 are more and more verbose
42 handler_args['loglevel'] = 1
43
44 # optional, use a debug format for the logging output (uses two lines per log entry)
45 handler_args['debugformat'] = False
46
47 # optional, this saves all response objects to handler.session.ALL_REQUESTS_RESPONSES
48 # very useful for capturing the full exchange of XML requests and responses
49 handler_args['record_all_requests'] = True
50
51 # instantiate a handler using all of the arguments in the handler_args dictionary
52 print "...CALLING: pytan.handler() with args: {}".format(handler_args)
53 handler = pytan.Handler(**handler_args)
54
55 # print out the handler string
56 print "...OUTPUT: handler string: {}".format(handler)
57
58 # setup the arguments for the handler.get() method
59 get_kwargs = {}
60 get_kwargs["objtype"] = u'question'
61 get_kwargs["id"] = 1
62
63 # get objects to use as an export to JSON file
64 print "...CALLING: handler.get() with args: {}".format(get_kwargs)
65 orig_objs = handler.get(**get_kwargs)
66
67 # export orig_objs to a json file
68 export_kwargs = {}

```

```
69 export_kwargs['obj'] = orig_objs
70 export_kwargs['export_format'] = 'json'
71 export_kwargs['report_dir'] = tempfile.gettempdir()
72
73 print "...CALLING: handler.export_to_report_file() with args: {}".format(export_kwargs)
74 json_file, results = handler.export_to_report_file(**export_kwargs)
75
76 # create the object from the exported JSON file
77 create_kwargs = {}
78 create_kwargs['objtype'] = u'question'
79 create_kwargs['json_file'] = json_file
80
81 print "...CALLING: handler.create_from_json() with args {}".format(create_kwargs)
82 response = handler.create_from_json(**create_kwargs)
83
84 print "...OUTPUT: Type of response: ", type(response)
85
86 print "...OUTPUT: print of response:"
87 print response
88
89 # call the export_obj() method to convert response to JSON and store it in out
90 export_kwargs = {}
91 export_kwargs['obj'] = response
92 export_kwargs['export_format'] = 'json'
93
94 print "...CALLING: handler.export_obj() with args {}".format(export_kwargs)
95 out = handler.export_obj(**export_kwargs)
96
97 # trim the output if it is more than 15 lines long
98 if len(out.splitlines()) > 15:
99     out = out.splitlines()[0:15]
100     out.append('..trimmed for brevity..')
101     out = '\n'.join(out)
102
103 print "...OUTPUT: print the objects returned in JSON format:"
104 print out
```

### 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

- STDOUT from Example Python Code
- STDERR from Example Python Code
- Example Python Code

```
1 # import the basic python packages we need
2 import os
3 import sys
4 import tempfile
5 import pprint
6 import traceback
7
8 # disable python from generating a .pyc file
9 sys.dont_write_bytecode = True
```

```

10
11 # change me to the path of pytan if this script is not running from EXAMPLES/PYTAN_API
12 pytan_loc = "~/gh/pytan"
13 pytan_static_path = os.path.join(os.path.expanduser(pytan_loc), 'lib')
14
15 # Determine our script name, script dir
16 my_file = os.path.abspath(sys.argv[0])
17 my_dir = os.path.dirname(my_file)
18
19 # try to automatically determine the pytan lib directory by assuming it is in '../..lib/'
20 parent_dir = os.path.dirname(my_dir)
21 pytan_root_dir = os.path.dirname(parent_dir)
22 lib_dir = os.path.join(pytan_root_dir, 'lib')
23
24 # add pytan_loc and lib_dir to the PYTHONPATH variable
25 path_adds = [lib_dir, pytan_static_path]
26 [sys.path.append(aa) for aa in path_adds if aa not in sys.path]
27
28 # import pytan
29 import pytan
30
31 # create a dictionary of arguments for the pytan handler
32 handler_args = {}
33
34 # establish our connection info for the Tanium Server
35 handler_args['username'] = "Administrator"
36 handler_args['password'] = "Tanium2015!"
37 handler_args['host'] = "10.0.1.240"
38 handler_args['port'] = "443" # optional
39
40 # optional, level 0 is no output except warnings/errors
41 # level 1 through 12 are more and more verbose
42 handler_args['loglevel'] = 1
43
44 # optional, use a debug format for the logging output (uses two lines per log entry)
45 handler_args['debugformat'] = False
46
47 # optional, this saves all response objects to handler.session.ALL_REQUESTS_RESPONSES
48 # very useful for capturing the full exchange of XML requests and responses
49 handler_args['record_all_requests'] = True
50
51 # instantiate a handler using all of the arguments in the handler_args dictionary
52 print "...CALLING: pytan.handler() with args: {}".format(handler_args)
53 handler = pytan.Handler(**handler_args)
54
55 # print out the handler string
56 print "...OUTPUT: handler string: {}".format(handler)
57
58 # setup the arguments for the handler.get() method
59 get_kwargs = {}
60 get_kwargs["objtype"] = u'whitelisted_url'
61 get_kwargs["url_regex"] = u'test1'
62
63 # get objects to use as an export to JSON file
64 print "...CALLING: handler.get() with args: {}".format(get_kwargs)
65 orig_objs = handler.get(**get_kwargs)
66
67 # set the attribute name and value we want to add to the original objects

```

```
68 # this is necessary to avoid name conflicts when adding the new object
69 attr_name = u'url_regex'
70 attr_value = u' API TEST'
71
72 # modify the orig_objs to add attr_value to attr_name
73 for x in orig_objs:
74     new_attr = getattr(x, attr_name)
75     new_attr += attr_value
76     setattr(x, attr_name, new_attr)
77
78     # delete the object in case it already exists
79     del_kwargs = {}
80     del_kwargs[attr_name] = new_attr
81     del_kwargs['objtype'] = u'whitelisted_url'
82
83     print "...CALLING: handler.delete() with args: {}".format(del_kwargs)
84     try:
85         handler.delete(**del_kwargs)
86     except Exception as e:
87         print "...EXCEPTION: {}".format(e)
88
89 # export orig_objs to a json file
90 export_kwargs = {}
91 export_kwargs['obj'] = orig_objs
92 export_kwargs['export_format'] = 'json'
93 export_kwargs['report_dir'] = tempfile.gettempdir()
94
95 print "...CALLING: handler.export_to_report_file() with args: {}".format(export_kwargs)
96 json_file, results = handler.export_to_report_file(**export_kwargs)
97
98 # create the object from the exported JSON file
99 create_kwargs = {}
100 create_kwargs['objtype'] = u'whitelisted_url'
101 create_kwargs['json_file'] = json_file
102
103 print "...CALLING: handler.create_from_json() with args {}".format(create_kwargs)
104 response = handler.create_from_json(**create_kwargs)
105
106 print "...OUTPUT: Type of response: ", type(response)
107
108 print "...OUTPUT: print of response:"
109 print response
110
111 # call the export_obj() method to convert response to JSON and store it in out
112 export_kwargs = {}
113 export_kwargs['obj'] = response
114 export_kwargs['export_format'] = 'json'
115
116 print "...CALLING: handler.export_obj() with args {}".format(export_kwargs)
117 out = handler.export_obj(**export_kwargs)
118
119 # trim the output if it is more than 15 lines long
120 if len(out.splitlines()) > 15:
121     out = out.splitlines()[0:15]
122     out.append('..trimmed for brevity..')
123     out = '\n'.join(out)
124
125 print "...OUTPUT: print the objects returned in JSON format:"
```



```
126 print out
```

### 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

- STDOUT from Example Python Code
- STDERR from Example Python Code
- Example Python Code

```

1  # import the basic python packages we need
2  import os
3  import sys
4  import tempfile
5  import pprint
6  import traceback
7
8  # disable python from generating a .pyc file
9  sys.dont_write_bytecode = True
10
11 # change me to the path of pytan if this script is not running from EXAMPLES/PYTAN_API
12 pytan_loc = "~/gh/pytan"
13 pytan_static_path = os.path.join(os.path.expanduser(pytan_loc), 'lib')
14
15 # Determine our script name, script dir
16 my_file = os.path.abspath(sys.argv[0])
17 my_dir = os.path.dirname(my_file)
18
19 # try to automatically determine the pytan lib directory by assuming it is in '../..lib/'
20 parent_dir = os.path.dirname(my_dir)
21 pytan_root_dir = os.path.dirname(parent_dir)
22 lib_dir = os.path.join(pytan_root_dir, 'lib')
23
24 # add pytan_loc and lib_dir to the PYTHONPATH variable
25 path_adds = [lib_dir, pytan_static_path]
26 [sys.path.append(aa) for aa in path_adds if aa not in sys.path]
27
28 # import pytan
29 import pytan
30
31 # create a dictionary of arguments for the pytan handler
32 handler_args = {}
33
34 # establish our connection info for the Tanium Server
35 handler_args['username'] = "Administrator"
36 handler_args['password'] = "Tanium2015!"
37 handler_args['host'] = "10.0.1.240"
38 handler_args['port'] = "443" # optional
39
40 # optional, level 0 is no output except warnings/errors
41 # level 1 through 12 are more and more verbose
42 handler_args['loglevel'] = 1
43
44 # optional, use a debug format for the logging output (uses two lines per log entry)

```

```
45 handler_args['debugformat'] = False
46
47 # optional, this saves all response objects to handler.session.ALL_REQUESTS_RESPONSES
48 # very useful for capturing the full exchange of XML requests and responses
49 handler_args['record_all_requests'] = True
50
51 # instantiate a handler using all of the arguments in the handler_args dictionary
52 print "...CALLING: pytan.handler() with args: {}".format(handler_args)
53 handler = pytan.Handler(**handler_args)
54
55 # print out the handler string
56 print "...OUTPUT: handler string: {}".format(handler)
57
58 # setup the arguments for the handler.get() method
59 get_kwargs = {}
60 get_kwargs["objtype"] = u'group'
61 get_kwargs["name"] = u'All Computers'
62
63 # get objects to use as an export to JSON file
64 print "...CALLING: handler.get() with args: {}".format(get_kwargs)
65 orig_objs = handler.get(**get_kwargs)
66
67 # set the attribute name and value we want to add to the original objects
68 # this is necessary to avoid name conflicts when adding the new object
69 attr_name = u'name'
70 attr_value = u' API TEST'
71
72 # modify the orig_objs to add attr_value to attr_name
73 for x in orig_objs:
74     new_attr = getattr(x, attr_name)
75     new_attr += attr_value
76     setattr(x, attr_name, new_attr)
77
78 # delete the object in case it already exists
79 del_kwargs = {}
80 del_kwargs[attr_name] = new_attr
81 del_kwargs['objtype'] = u'group'
82
83 print "...CALLING: handler.delete() with args: {}".format(del_kwargs)
84 try:
85     handler.delete(**del_kwargs)
86 except Exception as e:
87     print "...EXCEPTION: {}".format(e)
88
89 # export orig_objs to a json file
90 export_kwargs = {}
91 export_kwargs['obj'] = orig_objs
92 export_kwargs['export_format'] = 'json'
93 export_kwargs['report_dir'] = tempfile.gettempdir()
94
95 print "...CALLING: handler.export_to_report_file() with args: {}".format(export_kwargs)
96 json_file, results = handler.export_to_report_file(**export_kwargs)
97
98 # create the object from the exported JSON file
99 create_kwargs = {}
100 create_kwargs['objtype'] = u'group'
101 create_kwargs['json_file'] = json_file
102
```

```

103 print "...CALLING: handler.create_from_json() with args {}".format(create_kwargs)
104 response = handler.create_from_json(**create_kwargs)
105
106 print "...OUTPUT: Type of response: ", type(response)
107
108 print "...OUTPUT: print of response:"
109 print response
110
111 # call the export_obj() method to convert response to JSON and store it in out
112 export_kwargs = {}
113 export_kwargs['obj'] = response
114 export_kwargs['export_format'] = 'json'
115
116 print "...CALLING: handler.export_obj() with args {}".format(export_kwargs)
117 out = handler.export_obj(**export_kwargs)
118
119 # trim the output if it is more than 15 lines long
120 if len(out.splitlines()) > 15:
121     out = out.splitlines()[0:15]
122     out.append('..trimmed for brevity..')
123     out = '\n'.join(out)
124
125 print "...OUTPUT: print the objects returned in JSON format:"
126 print out

```

## PyTan API Valid Deploy Action Examples

All of the PyTan API examples for Valid Deploy Action

### Deploy Action Simple

Deploy an action against all computers using human strings.

- STDOUT from Example Python Code
- STDERR from Example Python Code
- Example Python Code

```

1 # import the basic python packages we need
2 import os
3 import sys
4 import tempfile
5 import pprint
6 import traceback
7
8 # disable python from generating a .pyc file
9 sys.dont_write_bytecode = True
10
11 # change me to the path of pytan if this script is not running from EXAMPLES/PYTAN_API
12 pytan_loc = "~/gh/pytan"
13 pytan_static_path = os.path.join(os.path.expanduser(pytan_loc), 'lib')
14
15 # Determine our script name, script dir
16 my_file = os.path.abspath(sys.argv[0])
17 my_dir = os.path.dirname(my_file)
18

```

```
19 # try to automatically determine the pytan lib directory by assuming it is in '../..lib/'
20 parent_dir = os.path.dirname(my_dir)
21 pytan_root_dir = os.path.dirname(parent_dir)
22 lib_dir = os.path.join(pytan_root_dir, 'lib')
23
24 # add pytan_loc and lib_dir to the PYTHONPATH variable
25 path_adds = [lib_dir, pytan_static_path]
26 [sys.path.append(aa) for aa in path_adds if aa not in sys.path]
27
28 # import pytan
29 import pytan
30
31 # create a dictionary of arguments for the pytan handler
32 handler_args = {}
33
34 # establish our connection info for the Tanium Server
35 handler_args['username'] = "Administrator"
36 handler_args['password'] = "Tanium2015!"
37 handler_args['host'] = "10.0.1.240"
38 handler_args['port'] = "443" # optional
39
40 # optional, level 0 is no output except warnings/errors
41 # level 1 through 12 are more and more verbose
42 handler_args['loglevel'] = 1
43
44 # optional, use a debug format for the logging output (uses two lines per log entry)
45 handler_args['debugformat'] = False
46
47 # optional, this saves all response objects to handler.session.ALL_REQUESTS_RESPONSES
48 # very useful for capturing the full exchange of XML requests and responses
49 handler_args['record_all_requests'] = True
50
51 # instantiate a handler using all of the arguments in the handler_args dictionary
52 print "...CALLING: pytan.handler() with args: {}".format(handler_args)
53 handler = pytan.Handler(**handler_args)
54
55 # print out the handler string
56 print "...OUTPUT: handler string: {}".format(handler)
57
58 # setup the arguments for the handler() class
59 kwargs = {}
60 kwargs["run"] = True
61 kwargs["package"] = u'Distribute Tanium Standard Utilities'
62
63 print "...CALLING: handler.deploy_action with args: {}".format(kwargs)
64 response = handler.deploy_action(**kwargs)
65
66 print "...OUTPUT: Type of response: ", type(response)
67
68 print "...OUTPUT: Pretty print of response:"
69 print pprint.pformat(response)
70
71 print "...OUTPUT: Print of action object: "
72 print response['action_object']
73
74 # if results were returned (i.e. get_results=True was one of the kwargs passed in):
75 if response['action_results']:
76     # call the export_obj() method to convert response to CSV and store it in out
```

```

77     export_kwargs = {}
78     export_kwargs['obj'] = response['action_results']
79     export_kwargs['export_format'] = 'csv'
80     print "...CALLING: handler.export_obj() with args {}".format(export_kwargs)
81     out = handler.export_obj(**export_kwargs)
82
83     # trim the output if it is more than 15 lines long
84     if len(out.splitlines()) > 15:
85         out = out.splitlines()[0:15]
86         out.append('..trimmed for brevity..')
87         out = '\n'.join(out)
88
89     print "...OUTPUT: CSV Results of response: "
90     print out

```

### 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.

- STDOUT from Example Python Code
- STDERR from Example Python Code
- Example Python Code

```

1  # import the basic python packages we need
2  import os
3  import sys
4  import tempfile
5  import pprint
6  import traceback
7
8  # disable python from generating a .pyc file
9  sys.dont_write_bytecode = True
10
11  # change me to the path of pytan if this script is not running from EXAMPLES/PYTAN_API
12  pytan_loc = "~/gh/pytan"
13  pytan_static_path = os.path.join(os.path.expanduser(pytan_loc), 'lib')
14
15  # Determine our script name, script dir
16  my_file = os.path.abspath(sys.argv[0])
17  my_dir = os.path.dirname(my_file)
18
19  # try to automatically determine the pytan lib directory by assuming it is in '.././lib/'
20  parent_dir = os.path.dirname(my_dir)
21  pytan_root_dir = os.path.dirname(parent_dir)
22  lib_dir = os.path.join(pytan_root_dir, 'lib')
23
24  # add pytan_loc and lib_dir to the PYTHONPATH variable
25  path_adds = [lib_dir, pytan_static_path]
26  [sys.path.append(aa) for aa in path_adds if aa not in sys.path]
27
28  # import pytan
29  import pytan
30
31  # create a dictionary of arguments for the pytan handler
32  handler_args = {}

```

```
33
34 # establish our connection info for the Tanium Server
35 handler_args['username'] = "Administrator"
36 handler_args['password'] = "Tanium2015!"
37 handler_args['host'] = "10.0.1.240"
38 handler_args['port'] = "443" # optional
39
40 # optional, level 0 is no output except warnings/errors
41 # level 1 through 12 are more and more verbose
42 handler_args['loglevel'] = 1
43
44 # optional, use a debug format for the logging output (uses two lines per log entry)
45 handler_args['debugformat'] = False
46
47 # optional, this saves all response objects to handler.session.ALL_REQUESTS_RESPONSES
48 # very useful for capturing the full exchange of XML requests and responses
49 handler_args['record_all_requests'] = True
50
51 # instantiate a handler using all of the arguments in the handler_args dictionary
52 print "...CALLING: pytan.handler() with args: {}".format(handler_args)
53 handler = pytan.Handler(**handler_args)
54
55 # print out the handler string
56 print "...OUTPUT: handler string: {}".format(handler)
57
58 # setup the arguments for the handler() class
59 kwargs = {}
60 kwargs["get_results"] = False
61 kwargs["run"] = True
62 kwargs["package"] = u'Distribute Tanium Standard Utilities'
63
64 print "...CALLING: handler.deploy_action with args: {}".format(kwargs)
65 response = handler.deploy_action(**kwargs)
66
67 print "...OUTPUT: Type of response: ", type(response)
68
69 print "...OUTPUT: Pretty print of response:"
70 print pprint.pformat(response)
71
72 print "...OUTPUT: Print of action object: "
73 print response['action_object']
74
75 # if results were returned (i.e. get_results=True was one of the kwargs passed in):
76 if response['action_results']:
77     # call the export_obj() method to convert response to CSV and store it in out
78     export_kwargs = {}
79     export_kwargs['obj'] = response['action_results']
80     export_kwargs['export_format'] = 'csv'
81     print "...CALLING: handler.export_obj() with args {}".format(export_kwargs)
82     out = handler.export_obj(**export_kwargs)
83
84     # trim the output if it is more than 15 lines long
85     if len(out.splitlines()) > 15:
86         out = out.splitlines()[0:15]
87         out.append('..trimmed for brevity..')
88         out = '\n'.join(out)
89
90     print "...OUTPUT: CSV Results of response: "
```

```
91 print out
```

## Deploy Action Simple Against Windows Computers

Deploy an action against only windows computers using human strings. This requires passing in an action filter

- STDOUT from Example Python Code
- STDERR from Example Python Code
- Example Python Code

```
1  # import the basic python packages we need
2  import os
3  import sys
4  import tempfile
5  import pprint
6  import traceback
7
8  # disable python from generating a .pyc file
9  sys.dont_write_bytecode = True
10
11 # change me to the path of pytan if this script is not running from EXAMPLES/PYTAN_API
12 pytan_loc = "~/gh/pytan"
13 pytan_static_path = os.path.join(os.path.expanduser(pytan_loc), 'lib')
14
15 # Determine our script name, script dir
16 my_file = os.path.abspath(sys.argv[0])
17 my_dir = os.path.dirname(my_file)
18
19 # try to automatically determine the pytan lib directory by assuming it is in '../..lib/'
20 parent_dir = os.path.dirname(my_dir)
21 pytan_root_dir = os.path.dirname(parent_dir)
22 lib_dir = os.path.join(pytan_root_dir, 'lib')
23
24 # add pytan_loc and lib_dir to the PYTHONPATH variable
25 path_adds = [lib_dir, pytan_static_path]
26 [sys.path.append(aa) for aa in path_adds if aa not in sys.path]
27
28 # import pytan
29 import pytan
30
31 # create a dictionary of arguments for the pytan handler
32 handler_args = {}
33
34 # establish our connection info for the Tanium Server
35 handler_args['username'] = "Administrator"
36 handler_args['password'] = "Tanium2015!"
37 handler_args['host'] = "10.0.1.240"
38 handler_args['port'] = "443" # optional
39
40 # optional, level 0 is no output except warnings/errors
41 # level 1 through 12 are more and more verbose
42 handler_args['loglevel'] = 1
43
44 # optional, use a debug format for the logging output (uses two lines per log entry)
45 handler_args['debugformat'] = False
46
```

```
47 # optional, this saves all response objects to handler.session.ALL_REQUESTS_RESPONSES
48 # very useful for capturing the full exchange of XML requests and responses
49 handler_args['record_all_requests'] = True
50
51 # instantiate a handler using all of the arguments in the handler_args dictionary
52 print "...CALLING: pytan.handler() with args: {}".format(handler_args)
53 handler = pytan.Handler(**handler_args)
54
55 # print out the handler string
56 print "...OUTPUT: handler string: {}".format(handler)
57
58 # setup the arguments for the handler() class
59 kwargs = {}
60 kwargs["run"] = True
61 kwargs["action_filters"] = u'Operating System, that contains:Windows'
62 kwargs["package"] = u'Distribute Tanium Standard Utilities'
63
64 print "...CALLING: handler.deploy_action with args: {}".format(kwargs)
65 response = handler.deploy_action(**kwargs)
66
67 print "...OUTPUT: Type of response: ", type(response)
68
69 print "...OUTPUT: Pretty print of response:"
70 print pprint.pformat(response)
71
72 print "...OUTPUT: Print of action object: "
73 print response['action_object']
74
75 # if results were returned (i.e. get_results=True was one of the kwargs passed in):
76 if response['action_results']:
77     # call the export_obj() method to convert response to CSV and store it in out
78     export_kwargs = {}
79     export_kwargs['obj'] = response['action_results']
80     export_kwargs['export_format'] = 'csv'
81     print "...CALLING: handler.export_obj() with args {}".format(export_kwargs)
82     out = handler.export_obj(**export_kwargs)
83
84     # trim the output if it is more than 15 lines long
85     if len(out.splitlines()) > 15:
86         out = out.splitlines()[0:15]
87         out.append('..trimmed for brevity..')
88         out = '\n'.join(out)
89
90     print "...OUTPUT: CSV Results of response: "
91     print out
```

## 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.

- STDOUT from Example Python Code
- STDERR from Example Python Code
- Example Python Code



```

1  # import the basic python packages we need
2  import os
3  import sys
4  import tempfile
5  import pprint
6  import traceback
7
8  # disable python from generating a .pyc file
9  sys.dont_write_bytecode = True
10
11  # change me to the path of pytan if this script is not running from EXAMPLES/PYTAN_API
12  pytan_loc = "~/gh/pytan"
13  pytan_static_path = os.path.join(os.path.expanduser(pytan_loc), 'lib')
14
15  # Determine our script name, script dir
16  my_file = os.path.abspath(sys.argv[0])
17  my_dir = os.path.dirname(my_file)
18
19  # try to automatically determine the pytan lib directory by assuming it is in '../..lib/'
20  parent_dir = os.path.dirname(my_dir)
21  pytan_root_dir = os.path.dirname(parent_dir)
22  lib_dir = os.path.join(pytan_root_dir, 'lib')
23
24  # add pytan_loc and lib_dir to the PYTHONPATH variable
25  path_adds = [lib_dir, pytan_static_path]
26  [sys.path.append(aa) for aa in path_adds if aa not in sys.path]
27
28  # import pytan
29  import pytan
30
31  # create a dictionary of arguments for the pytan handler
32  handler_args = {}
33
34  # establish our connection info for the Tanium Server
35  handler_args['username'] = "Administrator"
36  handler_args['password'] = "Tanium2015!"
37  handler_args['host'] = "10.0.1.240"
38  handler_args['port'] = "443" # optional
39
40  # optional, level 0 is no output except warnings/errors
41  # level 1 through 12 are more and more verbose
42  handler_args['loglevel'] = 1
43
44  # optional, use a debug format for the logging output (uses two lines per log entry)
45  handler_args['debugformat'] = False
46
47  # optional, this saves all response objects to handler.session.ALL_REQUESTS_RESPONSES
48  # very useful for capturing the full exchange of XML requests and responses
49  handler_args['record_all_requests'] = True
50
51  # instantiate a handler using all of the arguments in the handler_args dictionary
52  print "...CALLING: pytan.handler() with args: {}".format(handler_args)
53  handler = pytan.Handler(**handler_args)
54
55  # print out the handler string
56  print "...OUTPUT: handler string: {}".format(handler)
57
58  # setup the arguments for the handler() class

```

```
59 kwargs = {}
60 kwargs["run"] = True
61 kwargs["action_filters"] = u'Operating System, that contains:Windows'
62 kwargs["package"] = u'Custom Tagging - Add Tags{$1=tag_should_be_added,$2=tag_should_be_ignore}'
63
64 print "...CALLING: handler.deploy_action with args: {}".format(kwargs)
65 response = handler.deploy_action(**kwargs)
66
67 print "...OUTPUT: Type of response: ", type(response)
68
69 print "...OUTPUT: Pretty print of response:"
70 pprint.pformat(response)
71
72 print "...OUTPUT: Print of action object: "
73 print response['action_object']
74
75 # if results were returned (i.e. get_results=True was one of the kwargs passed in):
76 if response['action_results']:
77     # call the export_obj() method to convert response to CSV and store it in out
78     export_kwargs = {}
79     export_kwargs['obj'] = response['action_results']
80     export_kwargs['export_format'] = 'csv'
81     print "...CALLING: handler.export_obj() with args {}".format(export_kwargs)
82     out = handler.export_obj(**export_kwargs)
83
84     # trim the output if it is more than 15 lines long
85     if len(out.splitlines()) > 15:
86         out = out.splitlines()[0:15]
87         out.append('..trimmed for brevity..')
88         out = '\n'.join(out)
89
90     print "...OUTPUT: CSV Results of response: "
91     print out
```

## PyTan API Valid Export Basetype Examples

All of the PyTan API examples for Valid Export Basetype

### Export Basetype CSV Default Options

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

- STDOUT from Example Python Code
- STDERR from Example Python Code
- Example Python Code

```
1 # import the basic python packages we need
2 import os
3 import sys
4 import tempfile
5 import pprint
6 import traceback
7
8 # disable python from generating a .pyc file
9 sys.dont_write_bytecode = True
```

```

10
11 # change me to the path of pytan if this script is not running from EXAMPLES/PYTAN_API
12 pytan_loc = "~/gh/pytan"
13 pytan_static_path = os.path.join(os.path.expanduser(pytan_loc), 'lib')
14
15 # Determine our script name, script dir
16 my_file = os.path.abspath(sys.argv[0])
17 my_dir = os.path.dirname(my_file)
18
19 # try to automatically determine the pytan lib directory by assuming it is in '../..lib/'
20 parent_dir = os.path.dirname(my_dir)
21 pytan_root_dir = os.path.dirname(parent_dir)
22 lib_dir = os.path.join(pytan_root_dir, 'lib')
23
24 # add pytan_loc and lib_dir to the PYTHONPATH variable
25 path_adds = [lib_dir, pytan_static_path]
26 [sys.path.append(aa) for aa in path_adds if aa not in sys.path]
27
28 # import pytan
29 import pytan
30
31 # create a dictionary of arguments for the pytan handler
32 handler_args = {}
33
34 # establish our connection info for the Tanium Server
35 handler_args['username'] = "Administrator"
36 handler_args['password'] = "Tanium2015!"
37 handler_args['host'] = "10.0.1.240"
38 handler_args['port'] = "443" # optional
39
40 # optional, level 0 is no output except warnings/errors
41 # level 1 through 12 are more and more verbose
42 handler_args['loglevel'] = 1
43
44 # optional, use a debug format for the logging output (uses two lines per log entry)
45 handler_args['debugformat'] = False
46
47 # optional, this saves all response objects to handler.session.ALL_REQUESTS_RESPONSES
48 # very useful for capturing the full exchange of XML requests and responses
49 handler_args['record_all_requests'] = True
50
51 # instantiate a handler using all of the arguments in the handler_args dictionary
52 print "...CALLING: pytan.handler() with args: {}".format(handler_args)
53 handler = pytan.Handler(**handler_args)
54
55 # print out the handler string
56 print "...OUTPUT: handler string: {}".format(handler)
57
58 # setup the arguments for the handler() class
59 kwargs = {}
60 kwargs["export_format"] = u'csv'
61
62 # setup the arguments for handler.get()
63 get_kwargs = {
64     'name': [
65         "Computer Name", "IP Route Details", "IP Address",
66         'Folder Name Search with RegEx Match',
67     ],

```

```
68     'objtype': 'sensor',
69 }
70
71 # get the objects that will provide the basetype that we want to export
72 print "...CALLING: handler.get() with args: {}".format(get_kwargs)
73 response = handler.get(**get_kwargs)
74
75 # store the basetype object as the obj we want to export
76 kwargs['obj'] = response
77
78 # export the object to a string
79 # (we could just as easily export to a file using export_to_report_file)
80 print "...CALLING: handler.export_obj() with args {}".format(kwargs)
81 out = handler.export_obj(**kwargs)
82
83 # trim the output if it is more than 15 lines long
84 if len(out.splitlines()) > 15:
85     out = out.splitlines()[0:15]
86     out.append('..trimmed for brevity..')
87     out = '\n'.join(out)
88
89 print "...OUTPUT: print the export_str returned from export_obj():"
90 print out
```

## Export Basetype JSON Type False

Export a BaseType from getting objects as JSON with false for include\_type

- [STDOUT from Example Python Code](#)
- [STDERR from Example Python Code](#)
- [Example Python Code](#)

```
1  # import the basic python packages we need
2  import os
3  import sys
4  import tempfile
5  import pprint
6  import traceback
7
8  # disable python from generating a .pyc file
9  sys.dont_write_bytecode = True
10
11 # change me to the path of pytan if this script is not running from EXAMPLES/PYTAN_API
12 pytan_loc = "~/gh/pytan"
13 pytan_static_path = os.path.join(os.path.expanduser(pytan_loc), 'lib')
14
15 # Determine our script name, script dir
16 my_file = os.path.abspath(sys.argv[0])
17 my_dir = os.path.dirname(my_file)
18
19 # try to automatically determine the pytan lib directory by assuming it is in '../..lib/'
20 parent_dir = os.path.dirname(my_dir)
21 pytan_root_dir = os.path.dirname(parent_dir)
22 lib_dir = os.path.join(pytan_root_dir, 'lib')
23
24 # add pytan_loc and lib_dir to the PYTHONPATH variable
```

```

25 path_adds = [lib_dir, pytan_static_path]
26 [sys.path.append(aa) for aa in path_adds if aa not in sys.path]
27
28 # import pytan
29 import pytan
30
31 # create a dictionary of arguments for the pytan handler
32 handler_args = {}
33
34 # establish our connection info for the Tanium Server
35 handler_args['username'] = "Administrator"
36 handler_args['password'] = "Tanium2015!"
37 handler_args['host'] = "10.0.1.240"
38 handler_args['port'] = "443" # optional
39
40 # optional, level 0 is no output except warnings/errors
41 # level 1 through 12 are more and more verbose
42 handler_args['loglevel'] = 1
43
44 # optional, use a debug format for the logging output (uses two lines per log entry)
45 handler_args['debugformat'] = False
46
47 # optional, this saves all response objects to handler.session.ALL_REQUESTS_RESPONSES
48 # very useful for capturing the full exchange of XML requests and responses
49 handler_args['record_all_requests'] = True
50
51 # instantiate a handler using all of the arguments in the handler_args dictionary
52 print "...CALLING: pytan.handler() with args: {}".format(handler_args)
53 handler = pytan.Handler(**handler_args)
54
55 # print out the handler string
56 print "...OUTPUT: handler string: {}".format(handler)
57
58 # setup the arguments for the handler() class
59 kwargs = {}
60 kwargs["export_format"] = u'json'
61 kwargs["include_type"] = False
62
63 # setup the arguments for handler.get()
64 get_kwargs = {
65     'name': [
66         "Computer Name", "IP Route Details", "IP Address",
67         'Folder Name Search with RegEx Match',
68     ],
69     'objtype': 'sensor',
70 }
71
72 # get the objects that will provide the basetype that we want to export
73 print "...CALLING: handler.get() with args: {}".format(get_kwargs)
74 response = handler.get(**get_kwargs)
75
76 # store the basetype object as the obj we want to export
77 kwargs['obj'] = response
78
79 # export the object to a string
80 # (we could just as easily export to a file using export_to_report_file)
81 print "...CALLING: handler.export_obj() with args {}".format(kwargs)
82 out = handler.export_obj(**kwargs)

```

```
83
84 # trim the output if it is more than 15 lines long
85 if len(out.splitlines()) > 15:
86     out = out.splitlines()[0:15]
87     out.append('..trimmed for brevity..')
88     out = '\n'.join(out)
89
90 print "...OUTPUT: print the export_str returned from export_obj():"
91 print out
```

## Export Basetype JSON Explode False

Export a BaseType from getting objects as JSON with false for explode\_json\_string\_values

- STDOUT from Example Python Code
- STDERR from Example Python Code
- Example Python Code

```
1 # import the basic python packages we need
2 import os
3 import sys
4 import tempfile
5 import pprint
6 import traceback
7
8 # disable python from generating a .pyc file
9 sys.dont_write_bytecode = True
10
11 # change me to the path of pytan if this script is not running from EXAMPLES/PYTAN_API
12 pytan_loc = "~/gh/pytan"
13 pytan_static_path = os.path.join(os.path.expanduser(pytan_loc), 'lib')
14
15 # Determine our script name, script dir
16 my_file = os.path.abspath(sys.argv[0])
17 my_dir = os.path.dirname(my_file)
18
19 # try to automatically determine the pytan lib directory by assuming it is in '../..lib/'
20 parent_dir = os.path.dirname(my_dir)
21 pytan_root_dir = os.path.dirname(parent_dir)
22 lib_dir = os.path.join(pytan_root_dir, 'lib')
23
24 # add pytan_loc and lib_dir to the PYTHONPATH variable
25 path_adds = [lib_dir, pytan_static_path]
26 [sys.path.append(aa) for aa in path_adds if aa not in sys.path]
27
28 # import pytan
29 import pytan
30
31 # create a dictionary of arguments for the pytan handler
32 handler_args = {}
33
34 # establish our connection info for the Tanium Server
35 handler_args['username'] = "Administrator"
36 handler_args['password'] = "Tanium2015!"
37 handler_args['host'] = "10.0.1.240"
38 handler_args['port'] = "443" # optional
```

```

39
40 # optional, level 0 is no output except warnings/errors
41 # level 1 through 12 are more and more verbose
42 handler_args['loglevel'] = 1
43
44 # optional, use a debug format for the logging output (uses two lines per log entry)
45 handler_args['debugformat'] = False
46
47 # optional, this saves all response objects to handler.session.ALL_REQUESTS_RESPONSES
48 # very useful for capturing the full exchange of XML requests and responses
49 handler_args['record_all_requests'] = True
50
51 # instantiate a handler using all of the arguments in the handler_args dictionary
52 print "...CALLING: pytan.handler() with args: {}".format(handler_args)
53 handler = pytan.Handler(**handler_args)
54
55 # print out the handler string
56 print "...OUTPUT: handler string: {}".format(handler)
57
58 # setup the arguments for the handler() class
59 kwargs = {}
60 kwargs["export_format"] = u'json'
61 kwargs["explode_json_string_values"] = False
62
63 # setup the arguments for handler.get()
64 get_kwargs = {
65     'name': [
66         "Computer Name", "IP Route Details", "IP Address",
67         'Folder Name Search with RegEx Match',
68     ],
69     'objtype': 'sensor',
70 }
71
72 # get the objects that will provide the basetype that we want to export
73 print "...CALLING: handler.get() with args: {}".format(get_kwargs)
74 response = handler.get(**get_kwargs)
75
76 # store the basetype object as the obj we want to export
77 kwargs['obj'] = response
78
79 # export the object to a string
80 # (we could just as easily export to a file using export_to_report_file)
81 print "...CALLING: handler.export_obj() with args {}".format(kwargs)
82 out = handler.export_obj(**kwargs)
83
84 # trim the output if it is more than 15 lines long
85 if len(out.splitlines()) > 15:
86     out = out.splitlines()[0:15]
87     out.append('..trimmed for brevity..')
88     out = '\n'.join(out)
89
90 print "...OUTPUT: print the export_str returned from export_obj():"
91 print out

```

### Export Basetype JSON Explode True

Export a BaseType from getting objects as JSON with true for explode\_json\_string\_values

- STDOUT from Example Python Code
- STDERR from Example Python Code
- Example Python Code

```
1  # import the basic python packages we need
2  import os
3  import sys
4  import tempfile
5  import pprint
6  import traceback
7
8  # disable python from generating a .pyc file
9  sys.dont_write_bytecode = True
10
11  # change me to the path of pytan if this script is not running from EXAMPLES/PYTAN_API
12  pytan_loc = "~/gh/pytan"
13  pytan_static_path = os.path.join(os.path.expanduser(pytan_loc), 'lib')
14
15  # Determine our script name, script dir
16  my_file = os.path.abspath(sys.argv[0])
17  my_dir = os.path.dirname(my_file)
18
19  # try to automatically determine the pytan lib directory by assuming it is in '../..lib/'
20  parent_dir = os.path.dirname(my_dir)
21  pytan_root_dir = os.path.dirname(parent_dir)
22  lib_dir = os.path.join(pytan_root_dir, 'lib')
23
24  # add pytan_loc and lib_dir to the PYTHONPATH variable
25  path_adds = [lib_dir, pytan_static_path]
26  [sys.path.append(aa) for aa in path_adds if aa not in sys.path]
27
28  # import pytan
29  import pytan
30
31  # create a dictionary of arguments for the pytan handler
32  handler_args = {}
33
34  # establish our connection info for the Tanium Server
35  handler_args['username'] = "Administrator"
36  handler_args['password'] = "Tanium2015!"
37  handler_args['host'] = "10.0.1.240"
38  handler_args['port'] = "443" # optional
39
40  # optional, level 0 is no output except warnings/errors
41  # level 1 through 12 are more and more verbose
42  handler_args['loglevel'] = 1
43
44  # optional, use a debug format for the logging output (uses two lines per log entry)
45  handler_args['debugformat'] = False
46
47  # optional, this saves all response objects to handler.session.ALL_REQUESTS_RESPONSES
48  # very useful for capturing the full exchange of XML requests and responses
49  handler_args['record_all_requests'] = True
50
51  # instantiate a handler using all of the arguments in the handler_args dictionary
52  print "...CALLING: pytan.handler() with args: {}".format(handler_args)
53  handler = pytan.Handler(**handler_args)
54
```



```

55 # print out the handler string
56 print "...OUTPUT: handler string: {}".format(handler)
57
58 # setup the arguments for the handler() class
59 kwargs = {}
60 kwargs["export_format"] = u'json'
61 kwargs["explode_json_string_values"] = True
62
63 # setup the arguments for handler.get()
64 get_kwargs = {
65     'name': [
66         "Computer Name", "IP Route Details", "IP Address",
67         'Folder Name Search with RegEx Match',
68     ],
69     'objtype': 'sensor',
70 }
71
72 # get the objects that will provide the basetype that we want to export
73 print "...CALLING: handler.get() with args: {}".format(get_kwargs)
74 response = handler.get(**get_kwargs)
75
76 # store the basetype object as the obj we want to export
77 kwargs['obj'] = response
78
79 # export the object to a string
80 # (we could just as easily export to a file using export_to_report_file)
81 print "...CALLING: handler.export_obj() with args {}".format(kwargs)
82 out = handler.export_obj(**kwargs)
83
84 # trim the output if it is more than 15 lines long
85 if len(out.splitlines()) > 15:
86     out = out.splitlines()[0:15]
87     out.append('..trimmed for brevity..')
88     out = '\n'.join(out)
89
90 print "...OUTPUT: print the export_str returned from export_obj():"
91 print out

```

### Export Basetype XML Default Options

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

- STDOUT from Example Python Code
- STDERR from Example Python Code
- Example Python Code

```

1 # import the basic python packages we need
2 import os
3 import sys
4 import tempfile
5 import pprint
6 import traceback
7
8 # disable python from generating a .pyc file
9 sys.dont_write_bytecode = True
10

```

```
11 # change me to the path of pytan if this script is not running from EXAMPLES/PYTAN_API
12 pytan_loc = "~/gh/pytan"
13 pytan_static_path = os.path.join(os.path.expanduser(pytan_loc), 'lib')
14
15 # Determine our script name, script dir
16 my_file = os.path.abspath(sys.argv[0])
17 my_dir = os.path.dirname(my_file)
18
19 # try to automatically determine the pytan lib directory by assuming it is in '../..lib/'
20 parent_dir = os.path.dirname(my_dir)
21 pytan_root_dir = os.path.dirname(parent_dir)
22 lib_dir = os.path.join(pytan_root_dir, 'lib')
23
24 # add pytan_loc and lib_dir to the PYTHONPATH variable
25 path_adds = [lib_dir, pytan_static_path]
26 [sys.path.append(aa) for aa in path_adds if aa not in sys.path]
27
28 # import pytan
29 import pytan
30
31 # create a dictionary of arguments for the pytan handler
32 handler_args = {}
33
34 # establish our connection info for the Tanium Server
35 handler_args['username'] = "Administrator"
36 handler_args['password'] = "Tanium2015!"
37 handler_args['host'] = "10.0.1.240"
38 handler_args['port'] = "443" # optional
39
40 # optional, level 0 is no output except warnings/errors
41 # level 1 through 12 are more and more verbose
42 handler_args['loglevel'] = 1
43
44 # optional, use a debug format for the logging output (uses two lines per log entry)
45 handler_args['debugformat'] = False
46
47 # optional, this saves all response objects to handler.session.ALL_REQUESTS_RESPONSES
48 # very useful for capturing the full exchange of XML requests and responses
49 handler_args['record_all_requests'] = True
50
51 # instantiate a handler using all of the arguments in the handler_args dictionary
52 print "...CALLING: pytan.handler() with args: {}".format(handler_args)
53 handler = pytan.Handler(**handler_args)
54
55 # print out the handler string
56 print "...OUTPUT: handler string: {}".format(handler)
57
58 # setup the arguments for the handler() class
59 kwargs = {}
60 kwargs["export_format"] = u'xml'
61
62 # setup the arguments for handler.get()
63 get_kwargs = {
64     'name': [
65         "Computer Name", "IP Route Details", "IP Address",
66         "Folder Name Search with RegEx Match",
67     ],
68     'objtype': 'sensor',
```

```

69 }
70
71 # get the objects that will provide the basetype that we want to export
72 print "...CALLING: handler.get() with args: {}".format(get_kwargs)
73 response = handler.get(**get_kwargs)
74
75 # store the basetype object as the obj we want to export
76 kwargs['obj'] = response
77
78 # export the object to a string
79 # (we could just as easily export to a file using export_to_report_file)
80 print "...CALLING: handler.export_obj() with args {}".format(kwargs)
81 out = handler.export_obj(**kwargs)
82
83 # trim the output if it is more than 15 lines long
84 if len(out.splitlines()) > 15:
85     out = out.splitlines()[0:15]
86     out.append('..trimmed for brevity..')
87     out = '\n'.join(out)
88
89 print "...OUTPUT: print the export_str returned from export_obj():"
90 print out

```

### Export Basetype XML Minimal False

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

- STDOUT from Example Python Code
- STDERR from Example Python Code
- Example Python Code

```

1  # import the basic python packages we need
2  import os
3  import sys
4  import tempfile
5  import pprint
6  import traceback
7
8  # disable python from generating a .pyc file
9  sys.dont_write_bytecode = True
10
11 # change me to the path of pytan if this script is not running from EXAMPLES/PYTAN_API
12 pytan_loc = "~/gh/pytan"
13 pytan_static_path = os.path.join(os.path.expanduser(pytan_loc), 'lib')
14
15 # Determine our script name, script dir
16 my_file = os.path.abspath(sys.argv[0])
17 my_dir = os.path.dirname(my_file)
18
19 # try to automatically determine the pytan lib directory by assuming it is in '../..lib/'
20 parent_dir = os.path.dirname(my_dir)
21 pytan_root_dir = os.path.dirname(parent_dir)
22 lib_dir = os.path.join(pytan_root_dir, 'lib')
23
24 # add pytan_loc and lib_dir to the PYTHONPATH variable
25 path_adds = [lib_dir, pytan_static_path]

```

```
26 [sys.path.append(aa) for aa in path_adds if aa not in sys.path]
27
28 # import pytan
29 import pytan
30
31 # create a dictionary of arguments for the pytan handler
32 handler_args = {}
33
34 # establish our connection info for the Tanium Server
35 handler_args['username'] = "Administrator"
36 handler_args['password'] = "Tanium2015!"
37 handler_args['host'] = "10.0.1.240"
38 handler_args['port'] = "443" # optional
39
40 # optional, level 0 is no output except warnings/errors
41 # level 1 through 12 are more and more verbose
42 handler_args['loglevel'] = 1
43
44 # optional, use a debug format for the logging output (uses two lines per log entry)
45 handler_args['debugformat'] = False
46
47 # optional, this saves all response objects to handler.session.ALL_REQUESTS_RESPONSES
48 # very useful for capturing the full exchange of XML requests and responses
49 handler_args['record_all_requests'] = True
50
51 # instantiate a handler using all of the arguments in the handler_args dictionary
52 print "...CALLING: pytan.handler() with args: {}".format(handler_args)
53 handler = pytan.Handler(**handler_args)
54
55 # print out the handler string
56 print "...OUTPUT: handler string: {}".format(handler)
57
58 # setup the arguments for the handler() class
59 kwargs = {}
60 kwargs["export_format"] = u'xml'
61 kwargs["minimal"] = False
62
63 # setup the arguments for handler.get()
64 get_kwargs = {
65     'name': [
66         "Computer Name", "IP Route Details", "IP Address",
67         'Folder Name Search with RegEx Match',
68     ],
69     'objtype': 'sensor',
70 }
71
72 # get the objects that will provide the basetype that we want to export
73 print "...CALLING: handler.get() with args: {}".format(get_kwargs)
74 response = handler.get(**get_kwargs)
75
76 # store the basetype object as the obj we want to export
77 kwargs['obj'] = response
78
79 # export the object to a string
80 # (we could just as easily export to a file using export_to_report_file)
81 print "...CALLING: handler.export_obj() with args {}".format(kwargs)
82 out = handler.export_obj(**kwargs)
83
```

```

84 # trim the output if it is more than 15 lines long
85 if len(out.splitlines()) > 15:
86     out = out.splitlines()[0:15]
87     out.append('..trimmed for brevity..')
88     out = '\n'.join(out)
89
90 print "...OUTPUT: print the export_str returned from export_obj():"
91 print out

```

## Export Basetype XML Minimal True

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

- STDOUT from Example Python Code
- STDERR from Example Python Code
- Example Python Code

```

1  # import the basic python packages we need
2  import os
3  import sys
4  import tempfile
5  import pprint
6  import traceback
7
8  # disable python from generating a .pyc file
9  sys.dont_write_bytecode = True
10
11 # change me to the path of pytan if this script is not running from EXAMPLES/PYTAN_API
12 pytan_loc = "~/gh/pytan"
13 pytan_static_path = os.path.join(os.path.expanduser(pytan_loc), 'lib')
14
15 # Determine our script name, script dir
16 my_file = os.path.abspath(sys.argv[0])
17 my_dir = os.path.dirname(my_file)
18
19 # try to automatically determine the pytan lib directory by assuming it is in '.././lib/'
20 parent_dir = os.path.dirname(my_dir)
21 pytan_root_dir = os.path.dirname(parent_dir)
22 lib_dir = os.path.join(pytan_root_dir, 'lib')
23
24 # add pytan_loc and lib_dir to the PYTHONPATH variable
25 path_adds = [lib_dir, pytan_static_path]
26 [sys.path.append(aa) for aa in path_adds if aa not in sys.path]
27
28 # import pytan
29 import pytan
30
31 # create a dictionary of arguments for the pytan handler
32 handler_args = {}
33
34 # establish our connection info for the Tanium Server
35 handler_args['username'] = "Administrator"
36 handler_args['password'] = "Tanium2015!"
37 handler_args['host'] = "10.0.1.240"
38 handler_args['port'] = "443" # optional
39

```

```
40 # optional, level 0 is no output except warnings/errors
41 # level 1 through 12 are more and more verbose
42 handler_args['loglevel'] = 1
43
44 # optional, use a debug format for the logging output (uses two lines per log entry)
45 handler_args['debugformat'] = False
46
47 # optional, this saves all response objects to handler.session.ALL_REQUESTS_RESPONSES
48 # very useful for capturing the full exchange of XML requests and responses
49 handler_args['record_all_requests'] = True
50
51 # instantiate a handler using all of the arguments in the handler_args dictionary
52 print "...CALLING: pytan.handler() with args: {}".format(handler_args)
53 handler = pytan.Handler(**handler_args)
54
55 # print out the handler string
56 print "...OUTPUT: handler string: {}".format(handler)
57
58 # setup the arguments for the handler() class
59 kwargs = {}
60 kwargs["export_format"] = u'xml'
61 kwargs["minimal"] = True
62
63 # setup the arguments for handler.get()
64 get_kwargs = {
65     'name': [
66         "Computer Name", "IP Route Details", "IP Address",
67         "Folder Name Search with RegEx Match",
68     ],
69     'objtype': 'sensor',
70 }
71
72 # get the objects that will provide the basetype that we want to export
73 print "...CALLING: handler.get() with args: {}".format(get_kwargs)
74 response = handler.get(**get_kwargs)
75
76 # store the basetype object as the obj we want to export
77 kwargs['obj'] = response
78
79 # export the object to a string
80 # (we could just as easily export to a file using export_to_report_file)
81 print "...CALLING: handler.export_obj() with args {}".format(kwargs)
82 out = handler.export_obj(**kwargs)
83
84 # trim the output if it is more than 15 lines long
85 if len(out.splitlines()) > 15:
86     out = out.splitlines()[0:15]
87     out.append('..trimmed for brevity..')
88     out = '\n'.join(out)
89
90 print "...OUTPUT: print the export_str returned from export_obj():"
91 print out
```

### Export Basetype CSV With Explode False

Export a BaseType from getting objects as CSV with false for explode\_json\_string\_values

- STDOUT from Example Python Code
- STDERR from Example Python Code
- Example Python Code

```

1  # import the basic python packages we need
2  import os
3  import sys
4  import tempfile
5  import pprint
6  import traceback
7
8  # disable python from generating a .pyc file
9  sys.dont_write_bytecode = True
10
11 # change me to the path of pytan if this script is not running from EXAMPLES/PYTAN_API
12 pytan_loc = "~/gh/pytan"
13 pytan_static_path = os.path.join(os.path.expanduser(pytan_loc), 'lib')
14
15 # Determine our script name, script dir
16 my_file = os.path.abspath(sys.argv[0])
17 my_dir = os.path.dirname(my_file)
18
19 # try to automatically determine the pytan lib directory by assuming it is in '../..lib/'
20 parent_dir = os.path.dirname(my_dir)
21 pytan_root_dir = os.path.dirname(parent_dir)
22 lib_dir = os.path.join(pytan_root_dir, 'lib')
23
24 # add pytan_loc and lib_dir to the PYTHONPATH variable
25 path_adds = [lib_dir, pytan_static_path]
26 [sys.path.append(aa) for aa in path_adds if aa not in sys.path]
27
28 # import pytan
29 import pytan
30
31 # create a dictionary of arguments for the pytan handler
32 handler_args = {}
33
34 # establish our connection info for the Tanium Server
35 handler_args['username'] = "Administrator"
36 handler_args['password'] = "Tanium2015!"
37 handler_args['host'] = "10.0.1.240"
38 handler_args['port'] = "443" # optional
39
40 # optional, level 0 is no output except warnings/errors
41 # level 1 through 12 are more and more verbose
42 handler_args['loglevel'] = 1
43
44 # optional, use a debug format for the logging output (uses two lines per log entry)
45 handler_args['debugformat'] = False
46
47 # optional, this saves all response objects to handler.session.ALL_REQUESTS_RESPONSES
48 # very useful for capturing the full exchange of XML requests and responses
49 handler_args['record_all_requests'] = True
50
51 # instantiate a handler using all of the arguments in the handler_args dictionary
52 print "...CALLING: pytan.handler() with args: {}".format(handler_args)
53 handler = pytan.Handler(**handler_args)
54

```

```
55 # print out the handler string
56 print "...OUTPUT: handler string: {}".format(handler)
57
58 # setup the arguments for the handler() class
59 kwargs = {}
60 kwargs["export_format"] = u'csv'
61 kwargs["explode_json_string_values"] = False
62
63 # setup the arguments for handler.get()
64 get_kwargs = {
65     'name': [
66         "Computer Name", "IP Route Details", "IP Address",
67         'Folder Name Search with RegEx Match',
68     ],
69     'objtype': 'sensor',
70 }
71
72 # get the objects that will provide the basetype that we want to export
73 print "...CALLING: handler.get() with args: {}".format(get_kwargs)
74 response = handler.get(**get_kwargs)
75
76 # store the basetype object as the obj we want to export
77 kwargs['obj'] = response
78
79 # export the object to a string
80 # (we could just as easily export to a file using export_to_report_file)
81 print "...CALLING: handler.export_obj() with args {}".format(kwargs)
82 out = handler.export_obj(**kwargs)
83
84 # trim the output if it is more than 15 lines long
85 if len(out.splitlines()) > 15:
86     out = out.splitlines()[0:15]
87     out.append('..trimmed for brevity..')
88     out = '\n'.join(out)
89
90 print "...OUTPUT: print the export_str returned from export_obj():"
91 print out
```

### Export Basetype CSV With Explode True

Export a BaseType from getting objects as CSV with true for explode\_json\_string\_values

- STDOUT from Example Python Code
- STDERR from Example Python Code
- Example Python Code

```
1 # import the basic python packages we need
2 import os
3 import sys
4 import tempfile
5 import pprint
6 import traceback
7
8 # disable python from generating a .pyc file
9 sys.dont_write_bytecode = True
10
```



```

11 # change me to the path of pytan if this script is not running from EXAMPLES/PYTAN_API
12 pytan_loc = "~/gh/pytan"
13 pytan_static_path = os.path.join(os.path.expanduser(pytan_loc), 'lib')
14
15 # Determine our script name, script dir
16 my_file = os.path.abspath(sys.argv[0])
17 my_dir = os.path.dirname(my_file)
18
19 # try to automatically determine the pytan lib directory by assuming it is in '../..lib/'
20 parent_dir = os.path.dirname(my_dir)
21 pytan_root_dir = os.path.dirname(parent_dir)
22 lib_dir = os.path.join(pytan_root_dir, 'lib')
23
24 # add pytan_loc and lib_dir to the PYTHONPATH variable
25 path_adds = [lib_dir, pytan_static_path]
26 [sys.path.append(aa) for aa in path_adds if aa not in sys.path]
27
28 # import pytan
29 import pytan
30
31 # create a dictionary of arguments for the pytan handler
32 handler_args = {}
33
34 # establish our connection info for the Tanium Server
35 handler_args['username'] = "Administrator"
36 handler_args['password'] = "Tanium2015!"
37 handler_args['host'] = "10.0.1.240"
38 handler_args['port'] = "443" # optional
39
40 # optional, level 0 is no output except warnings/errors
41 # level 1 through 12 are more and more verbose
42 handler_args['loglevel'] = 1
43
44 # optional, use a debug format for the logging output (uses two lines per log entry)
45 handler_args['debugformat'] = False
46
47 # optional, this saves all response objects to handler.session.ALL_REQUESTS_RESPONSES
48 # very useful for capturing the full exchange of XML requests and responses
49 handler_args['record_all_requests'] = True
50
51 # instantiate a handler using all of the arguments in the handler_args dictionary
52 print "...CALLING: pytan.handler() with args: {}".format(handler_args)
53 handler = pytan.Handler(**handler_args)
54
55 # print out the handler string
56 print "...OUTPUT: handler string: {}".format(handler)
57
58 # setup the arguments for the handler() class
59 kwargs = {}
60 kwargs["export_format"] = u'csv'
61 kwargs["explode_json_string_values"] = True
62
63 # setup the arguments for handler.get()
64 get_kwargs = {
65     'name': [
66         "Computer Name", "IP Route Details", "IP Address",
67         'Folder Name Search with RegEx Match',
68     ],

```

```
69     'objtype': 'sensor',
70 }
71
72 # get the objects that will provide the basetype that we want to export
73 print "...CALLING: handler.get() with args: {}".format(get_kwargs)
74 response = handler.get(**get_kwargs)
75
76 # store the basetype object as the obj we want to export
77 kwargs['obj'] = response
78
79 # export the object to a string
80 # (we could just as easily export to a file using export_to_report_file)
81 print "...CALLING: handler.export_obj() with args {}".format(kwargs)
82 out = handler.export_obj(**kwargs)
83
84 # trim the output if it is more than 15 lines long
85 if len(out.splitlines()) > 15:
86     out = out.splitlines()[0:15]
87     out.append('..trimmed for brevity..')
88     out = '\n'.join(out)
89
90 print "...OUTPUT: print the export_str returned from export_obj():"
91 print out
```

### Export Basetype CSV With Sort Empty List

Export a BaseType from getting objects as CSV with an empty list for header\_sort

- [STDOUT from Example Python Code](#)
- [STDERR from Example Python Code](#)
- [Example Python Code](#)

```
1  # import the basic python packages we need
2  import os
3  import sys
4  import tempfile
5  import pprint
6  import traceback
7
8  # disable python from generating a .pyc file
9  sys.dont_write_bytecode = True
10
11 # change me to the path of pytan if this script is not running from EXAMPLES/PYTAN_API
12 pytan_loc = "~/gh/pytan"
13 pytan_static_path = os.path.join(os.path.expanduser(pytan_loc), 'lib')
14
15 # Determine our script name, script dir
16 my_file = os.path.abspath(sys.argv[0])
17 my_dir = os.path.dirname(my_file)
18
19 # try to automatically determine the pytan lib directory by assuming it is in '../..lib/'
20 parent_dir = os.path.dirname(my_dir)
21 pytan_root_dir = os.path.dirname(parent_dir)
22 lib_dir = os.path.join(pytan_root_dir, 'lib')
23
24 # add pytan_loc and lib_dir to the PYTHONPATH variable
```

```

25 path_adds = [lib_dir, pytan_static_path]
26 [sys.path.append(aa) for aa in path_adds if aa not in sys.path]
27
28 # import pytan
29 import pytan
30
31 # create a dictionary of arguments for the pytan handler
32 handler_args = {}
33
34 # establish our connection info for the Tanium Server
35 handler_args['username'] = "Administrator"
36 handler_args['password'] = "Tanium2015!"
37 handler_args['host'] = "10.0.1.240"
38 handler_args['port'] = "443" # optional
39
40 # optional, level 0 is no output except warnings/errors
41 # level 1 through 12 are more and more verbose
42 handler_args['loglevel'] = 1
43
44 # optional, use a debug format for the logging output (uses two lines per log entry)
45 handler_args['debugformat'] = False
46
47 # optional, this saves all response objects to handler.session.ALL_REQUESTS_RESPONSES
48 # very useful for capturing the full exchange of XML requests and responses
49 handler_args['record_all_requests'] = True
50
51 # instantiate a handler using all of the arguments in the handler_args dictionary
52 print "...CALLING: pytan.handler() with args: {}".format(handler_args)
53 handler = pytan.Handler(**handler_args)
54
55 # print out the handler string
56 print "...OUTPUT: handler string: {}".format(handler)
57
58 # setup the arguments for the handler() class
59 kwargs = {}
60 kwargs["export_format"] = u'csv'
61 kwargs["header_sort"] = []
62
63 # setup the arguments for handler.get()
64 get_kwargs = {
65     'name': [
66         "Computer Name", "IP Route Details", "IP Address",
67         'Folder Name Search with RegEx Match',
68     ],
69     'objtype': 'sensor',
70 }
71
72 # get the objects that will provide the basetype that we want to export
73 print "...CALLING: handler.get() with args: {}".format(get_kwargs)
74 response = handler.get(**get_kwargs)
75
76 # store the basetype object as the obj we want to export
77 kwargs['obj'] = response
78
79 # export the object to a string
80 # (we could just as easily export to a file using export_to_report_file)
81 print "...CALLING: handler.export_obj() with args {}".format(kwargs)
82 out = handler.export_obj(**kwargs)

```

```
83
84 # trim the output if it is more than 15 lines long
85 if len(out.splitlines()) > 15:
86     out = out.splitlines()[0:15]
87     out.append('..trimmed for brevity..')
88     out = '\n'.join(out)
89
90 print "...OUTPUT: print the export_str returned from export_obj():"
91 print out
```

## Export Basetype CSV With Sort True

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

- STDOUT from Example Python Code
- STDERR from Example Python Code
- Example Python Code

```
1 # import the basic python packages we need
2 import os
3 import sys
4 import tempfile
5 import pprint
6 import traceback
7
8 # disable python from generating a .pyc file
9 sys.dont_write_bytecode = True
10
11 # change me to the path of pytan if this script is not running from EXAMPLES/PYTAN_API
12 pytan_loc = "~/gh/pytan"
13 pytan_static_path = os.path.join(os.path.expanduser(pytan_loc), 'lib')
14
15 # Determine our script name, script dir
16 my_file = os.path.abspath(sys.argv[0])
17 my_dir = os.path.dirname(my_file)
18
19 # try to automatically determine the pytan lib directory by assuming it is in '.././lib/'
20 parent_dir = os.path.dirname(my_dir)
21 pytan_root_dir = os.path.dirname(parent_dir)
22 lib_dir = os.path.join(pytan_root_dir, 'lib')
23
24 # add pytan_loc and lib_dir to the PYTHONPATH variable
25 path_adds = [lib_dir, pytan_static_path]
26 [sys.path.append(aa) for aa in path_adds if aa not in sys.path]
27
28 # import pytan
29 import pytan
30
31 # create a dictionary of arguments for the pytan handler
32 handler_args = {}
33
34 # establish our connection info for the Tanium Server
35 handler_args['username'] = "Administrator"
36 handler_args['password'] = "Tanium2015!"
37 handler_args['host'] = "10.0.1.240"
38 handler_args['port'] = "443" # optional
```

```

39
40 # optional, level 0 is no output except warnings/errors
41 # level 1 through 12 are more and more verbose
42 handler_args['loglevel'] = 1
43
44 # optional, use a debug format for the logging output (uses two lines per log entry)
45 handler_args['debugformat'] = False
46
47 # optional, this saves all response objects to handler.session.ALL_REQUESTS_RESPONSES
48 # very useful for capturing the full exchange of XML requests and responses
49 handler_args['record_all_requests'] = True
50
51 # instantiate a handler using all of the arguments in the handler_args dictionary
52 print "...CALLING: pytan.handler() with args: {}".format(handler_args)
53 handler = pytan.Handler(**handler_args)
54
55 # print out the handler string
56 print "...OUTPUT: handler string: {}".format(handler)
57
58 # setup the arguments for the handler() class
59 kwargs = {}
60 kwargs["export_format"] = u'csv'
61 kwargs["header_sort"] = True
62
63 # setup the arguments for handler.get()
64 get_kwargs = {
65     'name': [
66         "Computer Name", "IP Route Details", "IP Address",
67         'Folder Name Search with RegEx Match',
68     ],
69     'objtype': 'sensor',
70 }
71
72 # get the objects that will provide the basetype that we want to export
73 print "...CALLING: handler.get() with args: {}".format(get_kwargs)
74 response = handler.get(**get_kwargs)
75
76 # store the basetype object as the obj we want to export
77 kwargs['obj'] = response
78
79 # export the object to a string
80 # (we could just as easily export to a file using export_to_report_file)
81 print "...CALLING: handler.export_obj() with args {}".format(kwargs)
82 out = handler.export_obj(**kwargs)
83
84 # trim the output if it is more than 15 lines long
85 if len(out.splitlines()) > 15:
86     out = out.splitlines()[0:15]
87     out.append('..trimmed for brevity..')
88     out = '\n'.join(out)
89
90 print "...OUTPUT: print the export_str returned from export_obj():"
91 print out

```

## Export Basetype CSV With Sort List

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

- STDOUT from Example Python Code
- STDERR from Example Python Code
- Example Python Code

```
1  # import the basic python packages we need
2  import os
3  import sys
4  import tempfile
5  import pprint
6  import traceback
7
8  # disable python from generating a .pyc file
9  sys.dont_write_bytecode = True
10
11  # change me to the path of pytan if this script is not running from EXAMPLES/PYTAN_API
12  pytan_loc = "~/gh/pytan"
13  pytan_static_path = os.path.join(os.path.expanduser(pytan_loc), 'lib')
14
15  # Determine our script name, script dir
16  my_file = os.path.abspath(sys.argv[0])
17  my_dir = os.path.dirname(my_file)
18
19  # try to automatically determine the pytan lib directory by assuming it is in '../..lib/'
20  parent_dir = os.path.dirname(my_dir)
21  pytan_root_dir = os.path.dirname(parent_dir)
22  lib_dir = os.path.join(pytan_root_dir, 'lib')
23
24  # add pytan_loc and lib_dir to the PYTHONPATH variable
25  path_adds = [lib_dir, pytan_static_path]
26  [sys.path.append(aa) for aa in path_adds if aa not in sys.path]
27
28  # import pytan
29  import pytan
30
31  # create a dictionary of arguments for the pytan handler
32  handler_args = {}
33
34  # establish our connection info for the Tanium Server
35  handler_args['username'] = "Administrator"
36  handler_args['password'] = "Tanium2015!"
37  handler_args['host'] = "10.0.1.240"
38  handler_args['port'] = "443" # optional
39
40  # optional, level 0 is no output except warnings/errors
41  # level 1 through 12 are more and more verbose
42  handler_args['loglevel'] = 1
43
44  # optional, use a debug format for the logging output (uses two lines per log entry)
45  handler_args['debugformat'] = False
46
47  # optional, this saves all response objects to handler.session.ALL_REQUESTS_RESPONSES
48  # very useful for capturing the full exchange of XML requests and responses
49  handler_args['record_all_requests'] = True
50
51  # instantiate a handler using all of the arguments in the handler_args dictionary
52  print "...CALLING: pytan.handler() with args: {}".format(handler_args)
53  handler = pytan.Handler(**handler_args)
54
```

```

55 # print out the handler string
56 print "...OUTPUT: handler string: {}".format(handler)
57
58 # setup the arguments for the handler() class
59 kwargs = {}
60 kwargs["export_format"] = u'csv'
61 kwargs["header_sort"] = [u'name', u'description']
62
63 # setup the arguments for handler.get()
64 get_kwargs = {
65     'name': [
66         "Computer Name", "IP Route Details", "IP Address",
67         'Folder Name Search with RegEx Match',
68     ],
69     'objtype': 'sensor',
70 }
71
72 # get the objects that will provide the basetype that we want to export
73 print "...CALLING: handler.get() with args: {}".format(get_kwargs)
74 response = handler.get(**get_kwargs)
75
76 # store the basetype object as the obj we want to export
77 kwargs['obj'] = response
78
79 # export the object to a string
80 # (we could just as easily export to a file using export_to_report_file)
81 print "...CALLING: handler.export_obj() with args {}".format(kwargs)
82 out = handler.export_obj(**kwargs)
83
84 # trim the output if it is more than 15 lines long
85 if len(out.splitlines()) > 15:
86     out = out.splitlines()[0:15]
87     out.append('..trimmed for brevity..')
88     out = '\n'.join(out)
89
90 print "...OUTPUT: print the export_str returned from export_obj():"
91 print out

```

## Export Basetype JSON Default Options

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

- STDOUT from Example Python Code
- STDERR from Example Python Code
- Example Python Code

```

1 # import the basic python packages we need
2 import os
3 import sys
4 import tempfile
5 import pprint
6 import traceback
7
8 # disable python from generating a .pyc file
9 sys.dont_write_bytecode = True
10

```

```
11 # change me to the path of pytan if this script is not running from EXAMPLES/PYTAN_API
12 pytan_loc = "~/gh/pytan"
13 pytan_static_path = os.path.join(os.path.expanduser(pytan_loc), 'lib')
14
15 # Determine our script name, script dir
16 my_file = os.path.abspath(sys.argv[0])
17 my_dir = os.path.dirname(my_file)
18
19 # try to automatically determine the pytan lib directory by assuming it is in '../..lib/'
20 parent_dir = os.path.dirname(my_dir)
21 pytan_root_dir = os.path.dirname(parent_dir)
22 lib_dir = os.path.join(pytan_root_dir, 'lib')
23
24 # add pytan_loc and lib_dir to the PYTHONPATH variable
25 path_adds = [lib_dir, pytan_static_path]
26 [sys.path.append(aa) for aa in path_adds if aa not in sys.path]
27
28 # import pytan
29 import pytan
30
31 # create a dictionary of arguments for the pytan handler
32 handler_args = {}
33
34 # establish our connection info for the Tanium Server
35 handler_args['username'] = "Administrator"
36 handler_args['password'] = "Tanium2015!"
37 handler_args['host'] = "10.0.1.240"
38 handler_args['port'] = "443" # optional
39
40 # optional, level 0 is no output except warnings/errors
41 # level 1 through 12 are more and more verbose
42 handler_args['loglevel'] = 1
43
44 # optional, use a debug format for the logging output (uses two lines per log entry)
45 handler_args['debugformat'] = False
46
47 # optional, this saves all response objects to handler.session.ALL_REQUESTS_RESPONSES
48 # very useful for capturing the full exchange of XML requests and responses
49 handler_args['record_all_requests'] = True
50
51 # instantiate a handler using all of the arguments in the handler_args dictionary
52 print "...CALLING: pytan.handler() with args: {}".format(handler_args)
53 handler = pytan.Handler(**handler_args)
54
55 # print out the handler string
56 print "...OUTPUT: handler string: {}".format(handler)
57
58 # setup the arguments for the handler() class
59 kwargs = {}
60 kwargs["export_format"] = u'json'
61
62 # setup the arguments for handler.get()
63 get_kwargs = {
64     'name': [
65         "Computer Name", "IP Route Details", "IP Address",
66         "Folder Name Search with RegEx Match",
67     ],
68     'objtype': 'sensor',
```



```

69 }
70
71 # get the objects that will provide the basetype that we want to export
72 print "...CALLING: handler.get() with args: {}".format(get_kwargs)
73 response = handler.get(**get_kwargs)
74
75 # store the basetype object as the obj we want to export
76 kwargs['obj'] = response
77
78 # export the object to a string
79 # (we could just as easily export to a file using export_to_report_file)
80 print "...CALLING: handler.export_obj() with args {}".format(kwargs)
81 out = handler.export_obj(**kwargs)
82
83 # trim the output if it is more than 15 lines long
84 if len(out.splitlines()) > 15:
85     out = out.splitlines()[0:15]
86     out.append('..trimmed for brevity..')
87     out = '\n'.join(out)
88
89 print "...OUTPUT: print the export_str returned from export_obj():"
90 print out

```

### Export Basetype JSON Type True

Export a BaseType from getting objects as JSON with true for include\_type

- STDOUT from Example Python Code
- STDERR from Example Python Code
- Example Python Code

```

1  # import the basic python packages we need
2  import os
3  import sys
4  import tempfile
5  import pprint
6  import traceback
7
8  # disable python from generating a .pyc file
9  sys.dont_write_bytecode = True
10
11 # change me to the path of pytan if this script is not running from EXAMPLES/PYTAN_API
12 pytan_loc = "~/gh/pytan"
13 pytan_static_path = os.path.join(os.path.expanduser(pytan_loc), 'lib')
14
15 # Determine our script name, script dir
16 my_file = os.path.abspath(sys.argv[0])
17 my_dir = os.path.dirname(my_file)
18
19 # try to automatically determine the pytan lib directory by assuming it is in '../..lib/'
20 parent_dir = os.path.dirname(my_dir)
21 pytan_root_dir = os.path.dirname(parent_dir)
22 lib_dir = os.path.join(pytan_root_dir, 'lib')
23
24 # add pytan_loc and lib_dir to the PYTHONPATH variable
25 path_adds = [lib_dir, pytan_static_path]

```

```
26 [sys.path.append(aa) for aa in path_adds if aa not in sys.path]
27
28 # import pytan
29 import pytan
30
31 # create a dictionary of arguments for the pytan handler
32 handler_args = {}
33
34 # establish our connection info for the Tanium Server
35 handler_args['username'] = "Administrator"
36 handler_args['password'] = "Tanium2015!"
37 handler_args['host'] = "10.0.1.240"
38 handler_args['port'] = "443" # optional
39
40 # optional, level 0 is no output except warnings/errors
41 # level 1 through 12 are more and more verbose
42 handler_args['loglevel'] = 1
43
44 # optional, use a debug format for the logging output (uses two lines per log entry)
45 handler_args['debugformat'] = False
46
47 # optional, this saves all response objects to handler.session.ALL_REQUESTS_RESPONSES
48 # very useful for capturing the full exchange of XML requests and responses
49 handler_args['record_all_requests'] = True
50
51 # instantiate a handler using all of the arguments in the handler_args dictionary
52 print "...CALLING: pytan.handler() with args: {}".format(handler_args)
53 handler = pytan.Handler(**handler_args)
54
55 # print out the handler string
56 print "...OUTPUT: handler string: {}".format(handler)
57
58 # setup the arguments for the handler() class
59 kwargs = {}
60 kwargs["export_format"] = u'json'
61 kwargs["include_type"] = True
62
63 # setup the arguments for handler.get()
64 get_kwargs = {
65     'name': [
66         "Computer Name", "IP Route Details", "IP Address",
67         'Folder Name Search with RegEx Match',
68     ],
69     'objtype': 'sensor',
70 }
71
72 # get the objects that will provide the basetype that we want to export
73 print "...CALLING: handler.get() with args: {}".format(get_kwargs)
74 response = handler.get(**get_kwargs)
75
76 # store the basetype object as the obj we want to export
77 kwargs['obj'] = response
78
79 # export the object to a string
80 # (we could just as easily export to a file using export_to_report_file)
81 print "...CALLING: handler.export_obj() with args {}".format(kwargs)
82 out = handler.export_obj(**kwargs)
83
```

```

84 # trim the output if it is more than 15 lines long
85 if len(out.splitlines()) > 15:
86     out = out.splitlines()[0:15]
87     out.append('..trimmed for brevity..')
88     out = '\n'.join(out)
89
90 print "...OUTPUT: print the export_str returned from export_obj():"
91 print out

```

## PyTan API Valid Export ResultSet Examples

All of the PyTan API examples for Valid Export ResultSet

### Export ResultSet CSV Default Options

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

- STDOUT from Example Python Code
- STDERR from Example Python Code
- Example Python Code

```

1  # import the basic python packages we need
2  import os
3  import sys
4  import tempfile
5  import pprint
6  import traceback
7
8  # disable python from generating a .pyc file
9  sys.dont_write_bytecode = True
10
11 # change me to the path of pytan if this script is not running from EXAMPLES/PYTAN_API
12 pytan_loc = "~/gh/pytan"
13 pytan_static_path = os.path.join(os.path.expanduser(pytan_loc), 'lib')
14
15 # Determine our script name, script dir
16 my_file = os.path.abspath(sys.argv[0])
17 my_dir = os.path.dirname(my_file)
18
19 # try to automatically determine the pytan lib directory by assuming it is in '../..lib/'
20 parent_dir = os.path.dirname(my_dir)
21 pytan_root_dir = os.path.dirname(parent_dir)
22 lib_dir = os.path.join(pytan_root_dir, 'lib')
23
24 # add pytan_loc and lib_dir to the PYTHONPATH variable
25 path_adds = [lib_dir, pytan_static_path]
26 [sys.path.append(aa) for aa in path_adds if aa not in sys.path]
27
28 # import pytan
29 import pytan
30
31 # create a dictionary of arguments for the pytan handler
32 handler_args = {}
33
34 # establish our connection info for the Tanium Server

```

```
35 handler_args['username'] = "Administrator"
36 handler_args['password'] = "Tanium2015!"
37 handler_args['host'] = "10.0.1.240"
38 handler_args['port'] = "443" # optional
39
40 # optional, level 0 is no output except warnings/errors
41 # level 1 through 12 are more and more verbose
42 handler_args['loglevel'] = 1
43
44 # optional, use a debug format for the logging output (uses two lines per log entry)
45 handler_args['debugformat'] = False
46
47 # optional, this saves all response objects to handler.session.ALL_REQUESTS_RESPONSES
48 # very useful for capturing the full exchange of XML requests and responses
49 handler_args['record_all_requests'] = True
50
51 # instantiate a handler using all of the arguments in the handler_args dictionary
52 print "...CALLING: pytan.handler() with args: {}".format(handler_args)
53 handler = pytan.Handler(**handler_args)
54
55 # print out the handler string
56 print "...OUTPUT: handler string: {}".format(handler)
57
58 # setup the arguments for the handler() class
59 kwargs = {}
60 kwargs["export_format"] = u'csv'
61
62 # setup the arguments for handler.ask()
63 ask_kwargs = {
64     'qtype': 'manual',
65     'sensors': [
66         "Computer Name", "IP Route Details", "IP Address",
67         'Folder Name Search with RegEx Match{dirname=Program Files,regex=.*Shared.*}',
68     ],
69 }
70
71 # ask the question that will provide the resultset that we want to use
72 print "...CALLING: handler.ask() with args {}".format(ask_kwargs)
73 response = handler.ask(**ask_kwargs)
74
75 # store the resultset object as the obj we want to export into kwargs
76 kwargs['obj'] = response['question_results']
77
78 # export the object to a string
79 # (we could just as easily export to a file using export_to_report_file)
80 print "...CALLING: handler.export_obj() with args {}".format(kwargs)
81 out = handler.export_obj(**kwargs)
82
83 # trim the output if it is more than 15 lines long
84 if len(out.splitlines()) > 15:
85     out = out.splitlines()[0:15]
86     out.append('..trimmed for brevity..')
87     out = '\n'.join(out)
88
89 print "...OUTPUT: print the export_str returned from export_obj():"
90 print out
```

## Export ResultSet CSV Expand False

Export a ResultSet from asking a question as CSV with false for expand\_grouped\_columns

- STDOUT from Example Python Code
- STDERR from Example Python Code
- Example Python Code

```

1  # import the basic python packages we need
2  import os
3  import sys
4  import tempfile
5  import pprint
6  import traceback
7
8  # disable python from generating a .pyc file
9  sys.dont_write_bytecode = True
10
11 # change me to the path of pytan if this script is not running from EXAMPLES/PYTAN_API
12 pytan_loc = "~/gh/pytan"
13 pytan_static_path = os.path.join(os.path.expanduser(pytan_loc), 'lib')
14
15 # Determine our script name, script dir
16 my_file = os.path.abspath(sys.argv[0])
17 my_dir = os.path.dirname(my_file)
18
19 # try to automatically determine the pytan lib directory by assuming it is in '../..lib/'
20 parent_dir = os.path.dirname(my_dir)
21 pytan_root_dir = os.path.dirname(parent_dir)
22 lib_dir = os.path.join(pytan_root_dir, 'lib')
23
24 # add pytan_loc and lib_dir to the PYTHONPATH variable
25 path_adds = [lib_dir, pytan_static_path]
26 [sys.path.append(aa) for aa in path_adds if aa not in sys.path]
27
28 # import pytan
29 import pytan
30
31 # create a dictionary of arguments for the pytan handler
32 handler_args = {}
33
34 # establish our connection info for the Tanium Server
35 handler_args['username'] = "Administrator"
36 handler_args['password'] = "Tanium2015!"
37 handler_args['host'] = "10.0.1.240"
38 handler_args['port'] = "443" # optional
39
40 # optional, level 0 is no output except warnings/errors
41 # level 1 through 12 are more and more verbose
42 handler_args['loglevel'] = 1
43
44 # optional, use a debug format for the logging output (uses two lines per log entry)
45 handler_args['debugformat'] = False
46
47 # optional, this saves all response objects to handler.session.ALL_REQUESTS_RESPONSES
48 # very useful for capturing the full exchange of XML requests and responses
49 handler_args['record_all_requests'] = True

```

```
50
51 # instantiate a handler using all of the arguments in the handler_args dictionary
52 print "...CALLING: pytan.handler() with args: {}".format(handler_args)
53 handler = pytan.Handler(**handler_args)
54
55 # print out the handler string
56 print "...OUTPUT: handler string: {}".format(handler)
57
58 # setup the arguments for the handler() class
59 kwargs = {}
60 kwargs["export_format"] = u'csv'
61 kwargs["expand_grouped_columns"] = False
62
63 # setup the arguments for handler.ask()
64 ask_kwargs = {
65     'qtype': 'manual',
66     'sensors': [
67         "Computer Name", "IP Route Details", "IP Address",
68         'Folder Name Search with RegEx Match{dirname=Program Files,regex=.*Shared.*}',
69     ],
70 }
71
72 # ask the question that will provide the resultset that we want to use
73 print "...CALLING: handler.ask() with args {}".format(ask_kwargs)
74 response = handler.ask(**ask_kwargs)
75
76 # store the resultset object as the obj we want to export into kwargs
77 kwargs['obj'] = response['question_results']
78
79 # export the object to a string
80 # (we could just as easily export to a file using export_to_report_file)
81 print "...CALLING: handler.export_obj() with args {}".format(kwargs)
82 out = handler.export_obj(**kwargs)
83
84 # trim the output if it is more than 15 lines long
85 if len(out.splitlines()) > 15:
86     out = out.splitlines()[0:15]
87     out.append('..trimmed for brevity..')
88     out = '\n'.join(out)
89
90 print "...OUTPUT: print the export_str returned from export_obj():"
91 print out
```

### Export ResultSet CSV Expand True

Export a ResultSet from asking a question as CSV with true for expand\_grouped\_columns

- STDOUT from Example Python Code
- STDERR from Example Python Code
- Example Python Code

```
1 # import the basic python packages we need
2 import os
3 import sys
4 import tempfile
5 import pprint
```

```

6 import traceback
7
8 # disable python from generating a .pyc file
9 sys.dont_write_bytecode = True
10
11 # change me to the path of pytan if this script is not running from EXAMPLES/PYTAN_API
12 pytan_loc = "~/gh/pytan"
13 pytan_static_path = os.path.join(os.path.expanduser(pytan_loc), 'lib')
14
15 # Determine our script name, script dir
16 my_file = os.path.abspath(sys.argv[0])
17 my_dir = os.path.dirname(my_file)
18
19 # try to automatically determine the pytan lib directory by assuming it is in '../..lib/'
20 parent_dir = os.path.dirname(my_dir)
21 pytan_root_dir = os.path.dirname(parent_dir)
22 lib_dir = os.path.join(pytan_root_dir, 'lib')
23
24 # add pytan_loc and lib_dir to the PYTHONPATH variable
25 path_adds = [lib_dir, pytan_static_path]
26 [sys.path.append(aa) for aa in path_adds if aa not in sys.path]
27
28 # import pytan
29 import pytan
30
31 # create a dictionary of arguments for the pytan handler
32 handler_args = {}
33
34 # establish our connection info for the Tanium Server
35 handler_args['username'] = "Administrator"
36 handler_args['password'] = "Tanium2015!"
37 handler_args['host'] = "10.0.1.240"
38 handler_args['port'] = "443" # optional
39
40 # optional, level 0 is no output except warnings/errors
41 # level 1 through 12 are more and more verbose
42 handler_args['loglevel'] = 1
43
44 # optional, use a debug format for the logging output (uses two lines per log entry)
45 handler_args['debugformat'] = False
46
47 # optional, this saves all response objects to handler.session.ALL_REQUESTS_RESPONSES
48 # very useful for capturing the full exchange of XML requests and responses
49 handler_args['record_all_requests'] = True
50
51 # instantiate a handler using all of the arguments in the handler_args dictionary
52 print "...CALLING: pytan.handler() with args: {}".format(handler_args)
53 handler = pytan.Handler(**handler_args)
54
55 # print out the handler string
56 print "...OUTPUT: handler string: {}".format(handler)
57
58 # setup the arguments for the handler() class
59 kwargs = {}
60 kwargs["export_format"] = u'csv'
61 kwargs["expand_grouped_columns"] = True
62
63 # setup the arguments for handler.ask()

```

```
64 ask_kwargs = {
65     'qtype': 'manual',
66     'sensors': [
67         "Computer Name", "IP Route Details", "IP Address",
68         'Folder Name Search with RegEx Match{dirname=Program Files,regex=.*Shared.*}',
69     ],
70 }
71
72 # ask the question that will provide the resultset that we want to use
73 print "...CALLING: handler.ask() with args {}".format(ask_kwargs)
74 response = handler.ask(**ask_kwargs)
75
76 # store the resultset object as the obj we want to export into kwargs
77 kwargs['obj'] = response['question_results']
78
79 # export the object to a string
80 # (we could just as easily export to a file using export_to_report_file)
81 print "...CALLING: handler.export_obj() with args {}".format(kwargs)
82 out = handler.export_obj(**kwargs)
83
84 # trim the output if it is more than 15 lines long
85 if len(out.splitlines()) > 15:
86     out = out.splitlines()[0:15]
87     out.append('...trimmed for brevity..')
88     out = '\n'.join(out)
89
90 print "...OUTPUT: print the export_str returned from export_obj():"
91 print out
```

### 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

- STDOUT from Example Python Code
- STDERR from Example Python Code
- Example Python Code

```
1 # import the basic python packages we need
2 import os
3 import sys
4 import tempfile
5 import pprint
6 import traceback
7
8 # disable python from generating a .pyc file
9 sys.dont_write_bytecode = True
10
11 # change me to the path of pytan if this script is not running from EXAMPLES/PYTAN_API
12 pytan_loc = "~/gh/pytan"
13 pytan_static_path = os.path.join(os.path.expanduser(pytan_loc), 'lib')
14
15 # Determine our script name, script dir
16 my_file = os.path.abspath(sys.argv[0])
17 my_dir = os.path.dirname(my_file)
18
```



```

19 # try to automatically determine the pytan lib directory by assuming it is in '../..lib/'
20 parent_dir = os.path.dirname(my_dir)
21 pytan_root_dir = os.path.dirname(parent_dir)
22 lib_dir = os.path.join(pytan_root_dir, 'lib')
23
24 # add pytan_loc and lib_dir to the PYTHONPATH variable
25 path_adds = [lib_dir, pytan_static_path]
26 [sys.path.append(aa) for aa in path_adds if aa not in sys.path]
27
28 # import pytan
29 import pytan
30
31 # create a dictionary of arguments for the pytan handler
32 handler_args = {}
33
34 # establish our connection info for the Tanium Server
35 handler_args['username'] = "Administrator"
36 handler_args['password'] = "Tanium2015!"
37 handler_args['host'] = "10.0.1.240"
38 handler_args['port'] = "443" # optional
39
40 # optional, level 0 is no output except warnings/errors
41 # level 1 through 12 are more and more verbose
42 handler_args['loglevel'] = 1
43
44 # optional, use a debug format for the logging output (uses two lines per log entry)
45 handler_args['debugformat'] = False
46
47 # optional, this saves all response objects to handler.session.ALL_REQUESTS_RESPONSES
48 # very useful for capturing the full exchange of XML requests and responses
49 handler_args['record_all_requests'] = True
50
51 # instantiate a handler using all of the arguments in the handler_args dictionary
52 print "...CALLING: pytan.handler() with args: {}".format(handler_args)
53 handler = pytan.Handler(**handler_args)
54
55 # print out the handler string
56 print "...OUTPUT: handler string: {}".format(handler)
57
58 # setup the arguments for the handler() class
59 kwargs = {}
60 kwargs["header_sort"] = True
61 kwargs["export_format"] = u'csv'
62 kwargs["header_add_type"] = True
63 kwargs["expand_grouped_columns"] = True
64 kwargs["header_add_sensor"] = True
65
66 # setup the arguments for handler.ask()
67 ask_kwargs = {
68     'qtype': 'manual',
69     'sensors': [
70         "Computer Name", "IP Route Details", "IP Address",
71         'Folder Name Search with RegEx Match{dirname=Program Files,regex=.*Shared.*}',
72     ],
73 }
74
75 # ask the question that will provide the resultset that we want to use
76 print "...CALLING: handler.ask() with args {}".format(ask_kwargs)

```

```
77 response = handler.ask(**ask_kwargs)
78
79 # store the resultset object as the obj we want to export into kwargs
80 kwargs['obj'] = response['question_results']
81
82 # export the object to a string
83 # (we could just as easily export to a file using export_to_report_file)
84 print "...CALLING: handler.export_obj() with args {}".format(kwargs)
85 out = handler.export_obj(**kwargs)
86
87 # trim the output if it is more than 15 lines long
88 if len(out.splitlines()) > 15:
89     out = out.splitlines()[0:15]
90     out.append('..trimmed for brevity..')
91     out = '\n'.join(out)
92
93 print "...OUTPUT: print the export_str returned from export_obj():"
94 print out
```

## Export ResultSet JSON

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

- STDOUT from Example Python Code
- STDERR from Example Python Code
- Example Python Code

```
1 # import the basic python packages we need
2 import os
3 import sys
4 import tempfile
5 import pprint
6 import traceback
7
8 # disable python from generating a .pyc file
9 sys.dont_write_bytecode = True
10
11 # change me to the path of pytan if this script is not running from EXAMPLES/PYTAN_API
12 pytan_loc = "~/gh/pytan"
13 pytan_static_path = os.path.join(os.path.expanduser(pytan_loc), 'lib')
14
15 # Determine our script name, script dir
16 my_file = os.path.abspath(sys.argv[0])
17 my_dir = os.path.dirname(my_file)
18
19 # try to automatically determine the pytan lib directory by assuming it is in '../..lib/'
20 parent_dir = os.path.dirname(my_dir)
21 pytan_root_dir = os.path.dirname(parent_dir)
22 lib_dir = os.path.join(pytan_root_dir, 'lib')
23
24 # add pytan_loc and lib_dir to the PYTHONPATH variable
25 path_adds = [lib_dir, pytan_static_path]
26 [sys.path.append(aa) for aa in path_adds if aa not in sys.path]
27
28 # import pytan
29 import pytan
```

```

30
31 # create a dictionary of arguments for the pytan handler
32 handler_args = {}
33
34 # establish our connection info for the Tanium Server
35 handler_args['username'] = "Administrator"
36 handler_args['password'] = "Tanium2015!"
37 handler_args['host'] = "10.0.1.240"
38 handler_args['port'] = "443" # optional
39
40 # optional, level 0 is no output except warnings/errors
41 # level 1 through 12 are more and more verbose
42 handler_args['loglevel'] = 1
43
44 # optional, use a debug format for the logging output (uses two lines per log entry)
45 handler_args['debugformat'] = False
46
47 # optional, this saves all response objects to handler.session.ALL_REQUESTS_RESPONSES
48 # very useful for capturing the full exchange of XML requests and responses
49 handler_args['record_all_requests'] = True
50
51 # instantiate a handler using all of the arguments in the handler_args dictionary
52 print "...CALLING: pytan.handler() with args: {}".format(handler_args)
53 handler = pytan.Handler(**handler_args)
54
55 # print out the handler string
56 print "...OUTPUT: handler string: {}".format(handler)
57
58 # setup the arguments for the handler() class
59 kwargs = {}
60 kwargs["export_format"] = u'json'
61
62 # setup the arguments for handler.ask()
63 ask_kwargs = {
64     'qtype': 'manual',
65     'sensors': [
66         "Computer Name", "IP Route Details", "IP Address",
67         'Folder Name Search with RegEx Match{dirname=Program Files,regex=.*Shared.*}',
68     ],
69 }
70
71 # ask the question that will provide the resultset that we want to use
72 print "...CALLING: handler.ask() with args {}".format(ask_kwargs)
73 response = handler.ask(**ask_kwargs)
74
75 # store the resultset object as the obj we want to export into kwargs
76 kwargs['obj'] = response['question_results']
77
78 # export the object to a string
79 # (we could just as easily export to a file using export_to_report_file)
80 print "...CALLING: handler.export_obj() with args {}".format(kwargs)
81 out = handler.export_obj(**kwargs)
82
83 # trim the output if it is more than 15 lines long
84 if len(out.splitlines()) > 15:
85     out = out.splitlines()[0:15]
86     out.append('..trimmed for brevity..')
87     out = '\n'.join(out)

```

```
88
89 print "...OUTPUT: print the export_str returned from export_obj():"
90 print out
```

### Export ResultSet CSV Sort Empty

Export a ResultSet from asking a question as CSV with an empty list for header\_sort

- STDOUT from Example Python Code
- STDERR from Example Python Code
- Example Python Code

```
1  # import the basic python packages we need
2  import os
3  import sys
4  import tempfile
5  import pprint
6  import traceback
7
8  # disable python from generating a .pyc file
9  sys.dont_write_bytecode = True
10
11  # change me to the path of pytan if this script is not running from EXAMPLES/PYTAN_API
12  pytan_loc = "~/gh/pytan"
13  pytan_static_path = os.path.join(os.path.expanduser(pytan_loc), 'lib')
14
15  # Determine our script name, script dir
16  my_file = os.path.abspath(sys.argv[0])
17  my_dir = os.path.dirname(my_file)
18
19  # try to automatically determine the pytan lib directory by assuming it is in '../..lib/'
20  parent_dir = os.path.dirname(my_dir)
21  pytan_root_dir = os.path.dirname(parent_dir)
22  lib_dir = os.path.join(pytan_root_dir, 'lib')
23
24  # add pytan_loc and lib_dir to the PYTHONPATH variable
25  path_adds = [lib_dir, pytan_static_path]
26  [sys.path.append(aa) for aa in path_adds if aa not in sys.path]
27
28  # import pytan
29  import pytan
30
31  # create a dictionary of arguments for the pytan handler
32  handler_args = {}
33
34  # establish our connection info for the Tanium Server
35  handler_args['username'] = "Administrator"
36  handler_args['password'] = "Tanium2015!"
37  handler_args['host'] = "10.0.1.240"
38  handler_args['port'] = "443" # optional
39
40  # optional, level 0 is no output except warnings/errors
41  # level 1 through 12 are more and more verbose
42  handler_args['loglevel'] = 1
43
44  # optional, use a debug format for the logging output (uses two lines per log entry)
```

```

45 handler_args['debugformat'] = False
46
47 # optional, this saves all response objects to handler.session.ALL_REQUESTS_RESPONSES
48 # very useful for capturing the full exchange of XML requests and responses
49 handler_args['record_all_requests'] = True
50
51 # instantiate a handler using all of the arguments in the handler_args dictionary
52 print "...CALLING: pytan.handler() with args: {}".format(handler_args)
53 handler = pytan.Handler(**handler_args)
54
55 # print out the handler string
56 print "...OUTPUT: handler string: {}".format(handler)
57
58 # setup the arguments for the handler() class
59 kwargs = {}
60 kwargs["export_format"] = u'csv'
61 kwargs["header_sort"] = []
62
63 # setup the arguments for handler.ask()
64 ask_kwargs = {
65     'qtype': 'manual',
66     'sensors': [
67         "Computer Name", "IP Route Details", "IP Address",
68         'Folder Name Search with RegEx Match{dirname=Program Files,regex=.*Shared.*}',
69     ],
70 }
71
72 # ask the question that will provide the resultset that we want to use
73 print "...CALLING: handler.ask() with args {}".format(ask_kwargs)
74 response = handler.ask(**ask_kwargs)
75
76 # store the resultset object as the obj we want to export into kwargs
77 kwargs['obj'] = response['question_results']
78
79 # export the object to a string
80 # (we could just as easily export to a file using export_to_report_file)
81 print "...CALLING: handler.export_obj() with args {}".format(kwargs)
82 out = handler.export_obj(**kwargs)
83
84 # trim the output if it is more than 15 lines long
85 if len(out.splitlines()) > 15:
86     out = out.splitlines()[0:15]
87     out.append('..trimmed for brevity..')
88     out = '\n'.join(out)
89
90 print "...OUTPUT: print the export_str returned from export_obj():"
91 print out

```

### Export ResultSet CSV Sort True

Export a ResultSet from asking a question as CSV with true for header\_sort

- STDOUT from Example Python Code
- STDERR from Example Python Code
- Example Python Code

```
1  # import the basic python packages we need
2  import os
3  import sys
4  import tempfile
5  import pprint
6  import traceback
7
8  # disable python from generating a .pyc file
9  sys.dont_write_bytecode = True
10
11  # change me to the path of pytan if this script is not running from EXAMPLES/PYTAN_API
12  pytan_loc = "~/gh/pytan"
13  pytan_static_path = os.path.join(os.path.expanduser(pytan_loc), 'lib')
14
15  # Determine our script name, script dir
16  my_file = os.path.abspath(sys.argv[0])
17  my_dir = os.path.dirname(my_file)
18
19  # try to automatically determine the pytan lib directory by assuming it is in '../..lib/'
20  parent_dir = os.path.dirname(my_dir)
21  pytan_root_dir = os.path.dirname(parent_dir)
22  lib_dir = os.path.join(pytan_root_dir, 'lib')
23
24  # add pytan_loc and lib_dir to the PYTHONPATH variable
25  path_adds = [lib_dir, pytan_static_path]
26  [sys.path.append(aa) for aa in path_adds if aa not in sys.path]
27
28  # import pytan
29  import pytan
30
31  # create a dictionary of arguments for the pytan handler
32  handler_args = {}
33
34  # establish our connection info for the Tanium Server
35  handler_args['username'] = "Administrator"
36  handler_args['password'] = "Tanium2015!"
37  handler_args['host'] = "10.0.1.240"
38  handler_args['port'] = "443" # optional
39
40  # optional, level 0 is no output except warnings/errors
41  # level 1 through 12 are more and more verbose
42  handler_args['loglevel'] = 1
43
44  # optional, use a debug format for the logging output (uses two lines per log entry)
45  handler_args['debugformat'] = False
46
47  # optional, this saves all response objects to handler.session.ALL_REQUESTS_RESPONSES
48  # very useful for capturing the full exchange of XML requests and responses
49  handler_args['record_all_requests'] = True
50
51  # instantiate a handler using all of the arguments in the handler_args dictionary
52  print "...CALLING: pytan.handler() with args: {}".format(handler_args)
53  handler = pytan.Handler(**handler_args)
54
55  # print out the handler string
56  print "...OUTPUT: handler string: {}".format(handler)
57
58  # setup the arguments for the handler() class
```

```

59 kwargs = {}
60 kwargs["export_format"] = u'csv'
61 kwargs["header_sort"] = True
62
63 # setup the arguments for handler.ask()
64 ask_kwargs = {
65     'qtype': 'manual',
66     'sensors': [
67         "Computer Name", "IP Route Details", "IP Address",
68         'Folder Name Search with RegEx Match{dirname=Program Files,regex=.*Shared.*}',
69     ],
70 }
71
72 # ask the question that will provide the resultset that we want to use
73 print "...CALLING: handler.ask() with args {}".format(ask_kwargs)
74 response = handler.ask(**ask_kwargs)
75
76 # store the resultset object as the obj we want to export into kwargs
77 kwargs['obj'] = response['question_results']
78
79 # export the object to a string
80 # (we could just as easily export to a file using export_to_report_file)
81 print "...CALLING: handler.export_obj() with args {}".format(kwargs)
82 out = handler.export_obj(**kwargs)
83
84 # trim the output if it is more than 15 lines long
85 if len(out.splitlines()) > 15:
86     out = out.splitlines()[0:15]
87     out.append('..trimmed for brevity..')
88     out = '\n'.join(out)
89
90 print "...OUTPUT: print the export_str returned from export_obj():"
91 print out

```

### Export ResultSet CSV Sort False

Export a ResultSet from asking a question as CSV with false for header\_sort

- STDOUT from Example Python Code
- STDERR from Example Python Code
- Example Python Code

```

1  # import the basic python packages we need
2  import os
3  import sys
4  import tempfile
5  import pprint
6  import traceback
7
8  # disable python from generating a .pyc file
9  sys.dont_write_bytecode = True
10
11 # change me to the path of pytan if this script is not running from EXAMPLES/PYTAN_API
12 pytan_loc = "~/gh/pytan"
13 pytan_static_path = os.path.join(os.path.expanduser(pytan_loc), 'lib')
14

```

```
15 # Determine our script name, script dir
16 my_file = os.path.abspath(sys.argv[0])
17 my_dir = os.path.dirname(my_file)
18
19 # try to automatically determine the pytan lib directory by assuming it is in '../../lib/'
20 parent_dir = os.path.dirname(my_dir)
21 pytan_root_dir = os.path.dirname(parent_dir)
22 lib_dir = os.path.join(pytan_root_dir, 'lib')
23
24 # add pytan_loc and lib_dir to the PYTHONPATH variable
25 path_adds = [lib_dir, pytan_static_path]
26 [sys.path.append(aa) for aa in path_adds if aa not in sys.path]
27
28 # import pytan
29 import pytan
30
31 # create a dictionary of arguments for the pytan handler
32 handler_args = {}
33
34 # establish our connection info for the Tanium Server
35 handler_args['username'] = "Administrator"
36 handler_args['password'] = "Tanium2015!"
37 handler_args['host'] = "10.0.1.240"
38 handler_args['port'] = "443" # optional
39
40 # optional, level 0 is no output except warnings/errors
41 # level 1 through 12 are more and more verbose
42 handler_args['loglevel'] = 1
43
44 # optional, use a debug format for the logging output (uses two lines per log entry)
45 handler_args['debugformat'] = False
46
47 # optional, this saves all response objects to handler.session.ALL_REQUESTS_RESPONSES
48 # very useful for capturing the full exchange of XML requests and responses
49 handler_args['record_all_requests'] = True
50
51 # instantiate a handler using all of the arguments in the handler_args dictionary
52 print "...CALLING: pytan.handler() with args: {}".format(handler_args)
53 handler = pytan.Handler(**handler_args)
54
55 # print out the handler string
56 print "...OUTPUT: handler string: {}".format(handler)
57
58 # setup the arguments for the handler() class
59 kwargs = {}
60 kwargs["export_format"] = u'csv'
61 kwargs["header_sort"] = False
62
63 # setup the arguments for handler.ask()
64 ask_kwargs = {
65     'qtype': 'manual',
66     'sensors': [
67         "Computer Name", "IP Route Details", "IP Address",
68         "Folder Name Search with RegEx Match{dirname=Program Files,regex=.*Shared.*}",
69     ],
70 }
71
72 # ask the question that will provide the resultset that we want to use
```



```

73 print "...CALLING: handler.ask() with args {}".format(ask_kwargs)
74 response = handler.ask(**ask_kwargs)
75
76 # store the resultset object as the obj we want to export into kwargs
77 kwargs['obj'] = response['question_results']
78
79 # export the object to a string
80 # (we could just as easily export to a file using export_to_report_file)
81 print "...CALLING: handler.export_obj() with args {}".format(kwargs)
82 out = handler.export_obj(**kwargs)
83
84 # trim the output if it is more than 15 lines long
85 if len(out.splitlines()) > 15:
86     out = out.splitlines()[0:15]
87     out.append('..trimmed for brevity..')
88     out = '\n'.join(out)
89
90 print "...OUTPUT: print the export_str returned from export_obj():"
91 print out

```

### Export ResultSet CSV Sort List

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

- STDOUT from Example Python Code
- STDERR from Example Python Code
- Example Python Code

```

1  # import the basic python packages we need
2  import os
3  import sys
4  import tempfile
5  import pprint
6  import traceback
7
8  # disable python from generating a .pyc file
9  sys.dont_write_bytecode = True
10
11 # change me to the path of pytan if this script is not running from EXAMPLES/PYTAN_API
12 pytan_loc = "~/gh/pytan"
13 pytan_static_path = os.path.join(os.path.expanduser(pytan_loc), 'lib')
14
15 # Determine our script name, script dir
16 my_file = os.path.abspath(sys.argv[0])
17 my_dir = os.path.dirname(my_file)
18
19 # try to automatically determine the pytan lib directory by assuming it is in '.././lib/'
20 parent_dir = os.path.dirname(my_dir)
21 pytan_root_dir = os.path.dirname(parent_dir)
22 lib_dir = os.path.join(pytan_root_dir, 'lib')
23
24 # add pytan_loc and lib_dir to the PYTHONPATH variable
25 path_adds = [lib_dir, pytan_static_path]
26 [sys.path.append(aa) for aa in path_adds if aa not in sys.path]
27
28 # import pytan

```

```
29 import pytan
30
31 # create a dictionary of arguments for the pytan handler
32 handler_args = {}
33
34 # establish our connection info for the Tanium Server
35 handler_args['username'] = "Administrator"
36 handler_args['password'] = "Tanium2015!"
37 handler_args['host'] = "10.0.1.240"
38 handler_args['port'] = "443" # optional
39
40 # optional, level 0 is no output except warnings/errors
41 # level 1 through 12 are more and more verbose
42 handler_args['loglevel'] = 1
43
44 # optional, use a debug format for the logging output (uses two lines per log entry)
45 handler_args['debugformat'] = False
46
47 # optional, this saves all response objects to handler.session.ALL_REQUESTS_RESPONSES
48 # very useful for capturing the full exchange of XML requests and responses
49 handler_args['record_all_requests'] = True
50
51 # instantiate a handler using all of the arguments in the handler_args dictionary
52 print "...CALLING: pytan.handler() with args: {}".format(handler_args)
53 handler = pytan.Handler(**handler_args)
54
55 # print out the handler string
56 print "...OUTPUT: handler string: {}".format(handler)
57
58 # setup the arguments for the handler() class
59 kwargs = {}
60 kwargs["export_format"] = u'csv'
61 kwargs["header_sort"] = [u'Computer Name', u'IP Address']
62
63 # setup the arguments for handler.ask()
64 ask_kwargs = {
65     'qtype': 'manual',
66     'sensors': [
67         "Computer Name", "IP Route Details", "IP Address",
68         'Folder Name Search with RegEx Match{dirname=Program Files,regex=.*Shared.*}',
69     ],
70 }
71
72 # ask the question that will provide the resultset that we want to use
73 print "...CALLING: handler.ask() with args {}".format(ask_kwargs)
74 response = handler.ask(**ask_kwargs)
75
76 # store the resultset object as the obj we want to export into kwargs
77 kwargs['obj'] = response['question_results']
78
79 # export the object to a string
80 # (we could just as easily export to a file using export_to_report_file)
81 print "...CALLING: handler.export_obj() with args {}".format(kwargs)
82 out = handler.export_obj(**kwargs)
83
84 # trim the output if it is more than 15 lines long
85 if len(out.splitlines()) > 15:
86     out = out.splitlines()[0:15]
```

```

87     out.append('..trimmed for brevity..')
88     out = '\n'.join(out)
89
90 print "...OUTPUT: print the export_str returned from export_obj():"
91 print out

```

### Export ResultSet CSV Type False

Export a ResultSet from asking a question as CSV with false for header\_add\_type

- STDOUT from Example Python Code
- STDERR from Example Python Code
- Example Python Code

```

1  # import the basic python packages we need
2  import os
3  import sys
4  import tempfile
5  import pprint
6  import traceback
7
8  # disable python from generating a .pyc file
9  sys.dont_write_bytecode = True
10
11 # change me to the path of pytan if this script is not running from EXAMPLES/PYTAN_API
12 pytan_loc = "~/gh/pytan"
13 pytan_static_path = os.path.join(os.path.expanduser(pytan_loc), 'lib')
14
15 # Determine our script name, script dir
16 my_file = os.path.abspath(sys.argv[0])
17 my_dir = os.path.dirname(my_file)
18
19 # try to automatically determine the pytan lib directory by assuming it is in '../..lib/'
20 parent_dir = os.path.dirname(my_dir)
21 pytan_root_dir = os.path.dirname(parent_dir)
22 lib_dir = os.path.join(pytan_root_dir, 'lib')
23
24 # add pytan_loc and lib_dir to the PYTHONPATH variable
25 path_adds = [lib_dir, pytan_static_path]
26 [sys.path.append(aa) for aa in path_adds if aa not in sys.path]
27
28 # import pytan
29 import pytan
30
31 # create a dictionary of arguments for the pytan handler
32 handler_args = {}
33
34 # establish our connection info for the Tanium Server
35 handler_args['username'] = "Administrator"
36 handler_args['password'] = "Tanium2015!"
37 handler_args['host'] = "10.0.1.240"
38 handler_args['port'] = "443" # optional
39
40 # optional, level 0 is no output except warnings/errors
41 # level 1 through 12 are more and more verbose
42 handler_args['loglevel'] = 1

```

```
43 # optional, use a debug format for the logging output (uses two lines per log entry)
44 handler_args['debugformat'] = False
45
46 # optional, this saves all response objects to handler.session.ALL_REQUESTS_RESPONSES
47 # very useful for capturing the full exchange of XML requests and responses
48 handler_args['record_all_requests'] = True
49
50 # instantiate a handler using all of the arguments in the handler_args dictionary
51 print "...CALLING: pytan.handler() with args: {}".format(handler_args)
52 handler = pytan.Handler(**handler_args)
53
54 # print out the handler string
55 print "...OUTPUT: handler string: {}".format(handler)
56
57 # setup the arguments for the handler() class
58 kwargs = {}
59 kwargs["export_format"] = u'csv'
60 kwargs["header_add_type"] = False
61
62 # setup the arguments for handler.ask()
63 ask_kwargs = {
64     'qtype': 'manual',
65     'sensors': [
66         "Computer Name", "IP Route Details", "IP Address",
67         'Folder Name Search with RegEx Match{dirname=Program Files,regex=.*Shared.*}',
68     ],
69 }
70
71 # ask the question that will provide the resultset that we want to use
72 print "...CALLING: handler.ask() with args {}".format(ask_kwargs)
73 response = handler.ask(**ask_kwargs)
74
75 # store the resultset object as the obj we want to export into kwargs
76 kwargs['obj'] = response['question_results']
77
78 # export the object to a string
79 # (we could just as easily export to a file using export_to_report_file)
80 print "...CALLING: handler.export_obj() with args {}".format(kwargs)
81 out = handler.export_obj(**kwargs)
82
83 # trim the output if it is more than 15 lines long
84 if len(out.splitlines()) > 15:
85     out = out.splitlines()[0:15]
86     out.append('..trimmed for brevity..')
87     out = '\n'.join(out)
88
89 print "...OUTPUT: print the export_str returned from export_obj():"
90 print out
```

### Export ResultSet CSV Type True

Export a ResultSet from asking a question as CSV with true for header\_add\_type

- STDOUT from Example Python Code
- STDERR from Example Python Code

- Example Python Code

```

1  # import the basic python packages we need
2  import os
3  import sys
4  import tempfile
5  import pprint
6  import traceback
7
8  # disable python from generating a .pyc file
9  sys.dont_write_bytecode = True
10
11 # change me to the path of pytan if this script is not running from EXAMPLES/PYTAN_API
12 pytan_loc = "~/gh/pytan"
13 pytan_static_path = os.path.join(os.path.expanduser(pytan_loc), 'lib')
14
15 # Determine our script name, script dir
16 my_file = os.path.abspath(sys.argv[0])
17 my_dir = os.path.dirname(my_file)
18
19 # try to automatically determine the pytan lib directory by assuming it is in '../..lib/'
20 parent_dir = os.path.dirname(my_dir)
21 pytan_root_dir = os.path.dirname(parent_dir)
22 lib_dir = os.path.join(pytan_root_dir, 'lib')
23
24 # add pytan_loc and lib_dir to the PYTHONPATH variable
25 path_adds = [lib_dir, pytan_static_path]
26 [sys.path.append(aa) for aa in path_adds if aa not in sys.path]
27
28 # import pytan
29 import pytan
30
31 # create a dictionary of arguments for the pytan handler
32 handler_args = {}
33
34 # establish our connection info for the Tanium Server
35 handler_args['username'] = "Administrator"
36 handler_args['password'] = "Tanium2015!"
37 handler_args['host'] = "10.0.1.240"
38 handler_args['port'] = "443" # optional
39
40 # optional, level 0 is no output except warnings/errors
41 # level 1 through 12 are more and more verbose
42 handler_args['loglevel'] = 1
43
44 # optional, use a debug format for the logging output (uses two lines per log entry)
45 handler_args['debugformat'] = False
46
47 # optional, this saves all response objects to handler.session.ALL_REQUESTS_RESPONSES
48 # very useful for capturing the full exchange of XML requests and responses
49 handler_args['record_all_requests'] = True
50
51 # instantiate a handler using all of the arguments in the handler_args dictionary
52 print "...CALLING: pytan.handler() with args: {}".format(handler_args)
53 handler = pytan.Handler(**handler_args)
54
55 # print out the handler string
56 print "...OUTPUT: handler string: {}".format(handler)
57

```

```
58 # setup the arguments for the handler() class
59 kwargs = {}
60 kwargs["export_format"] = u'csv'
61 kwargs["header_add_type"] = True
62
63 # setup the arguments for handler.ask()
64 ask_kwargs = {
65     'qtype': 'manual',
66     'sensors': [
67         "Computer Name", "IP Route Details", "IP Address",
68         'Folder Name Search with RegEx Match{dirname=Program Files,regex=.*Shared.*}',
69     ],
70 }
71
72 # ask the question that will provide the resultset that we want to use
73 print "...CALLING: handler.ask() with args {}".format(ask_kwargs)
74 response = handler.ask(**ask_kwargs)
75
76 # store the resultset object as the obj we want to export into kwargs
77 kwargs['obj'] = response['question_results']
78
79 # export the object to a string
80 # (we could just as easily export to a file using export_to_report_file)
81 print "...CALLING: handler.export_obj() with args {}".format(kwargs)
82 out = handler.export_obj(**kwargs)
83
84 # trim the output if it is more than 15 lines long
85 if len(out.splitlines()) > 15:
86     out = out.splitlines()[0:15]
87     out.append('..trimmed for brevity..')
88     out = '\n'.join(out)
89
90 print "...OUTPUT: print the export_str returned from export_obj():"
91 print out
```

## Export ResultSet CSV Sensor False

Export a ResultSet from asking a question as CSV with false for header\_add\_sensor

- STDOUT from Example Python Code
- STDERR from Example Python Code
- Example Python Code

```
1 # import the basic python packages we need
2 import os
3 import sys
4 import tempfile
5 import pprint
6 import traceback
7
8 # disable python from generating a .pyc file
9 sys.dont_write_bytecode = True
10
11 # change me to the path of pytan if this script is not running from EXAMPLES/PYTAN_API
12 pytan_loc = "~/gh/pytan"
13 pytan_static_path = os.path.join(os.path.expanduser(pytan_loc), 'lib')
```

```

14
15 # Determine our script name, script dir
16 my_file = os.path.abspath(sys.argv[0])
17 my_dir = os.path.dirname(my_file)
18
19 # try to automatically determine the pytan lib directory by assuming it is in '../..lib/'
20 parent_dir = os.path.dirname(my_dir)
21 pytan_root_dir = os.path.dirname(parent_dir)
22 lib_dir = os.path.join(pytan_root_dir, 'lib')
23
24 # add pytan_loc and lib_dir to the PYTHONPATH variable
25 path_adds = [lib_dir, pytan_static_path]
26 [sys.path.append(aa) for aa in path_adds if aa not in sys.path]
27
28 # import pytan
29 import pytan
30
31 # create a dictionary of arguments for the pytan handler
32 handler_args = {}
33
34 # establish our connection info for the Tanium Server
35 handler_args['username'] = "Administrator"
36 handler_args['password'] = "Tanium2015!"
37 handler_args['host'] = "10.0.1.240"
38 handler_args['port'] = "443" # optional
39
40 # optional, level 0 is no output except warnings/errors
41 # level 1 through 12 are more and more verbose
42 handler_args['loglevel'] = 1
43
44 # optional, use a debug format for the logging output (uses two lines per log entry)
45 handler_args['debugformat'] = False
46
47 # optional, this saves all response objects to handler.session.ALL_REQUESTS_RESPONSES
48 # very useful for capturing the full exchange of XML requests and responses
49 handler_args['record_all_requests'] = True
50
51 # instantiate a handler using all of the arguments in the handler_args dictionary
52 print "...CALLING: pytan.handler() with args: {}".format(handler_args)
53 handler = pytan.Handler(**handler_args)
54
55 # print out the handler string
56 print "...OUTPUT: handler string: {}".format(handler)
57
58 # setup the arguments for the handler() class
59 kwargs = {}
60 kwargs["export_format"] = u'csv'
61 kwargs["header_add_sensor"] = False
62
63 # setup the arguments for handler.ask()
64 ask_kwargs = {
65     'qtype': 'manual',
66     'sensors': [
67         "Computer Name", "IP Route Details", "IP Address",
68         "Folder Name Search with RegEx Match{dirname=Program Files,regex=.*Shared.*}",
69     ],
70 }
71

```

```
72 # ask the question that will provide the resultset that we want to use
73 print "...CALLING: handler.ask() with args {}".format(ask_kwargs)
74 response = handler.ask(**ask_kwargs)
75
76 # store the resultset object as the obj we want to export into kwargs
77 kwargs['obj'] = response['question_results']
78
79 # export the object to a string
80 # (we could just as easily export to a file using export_to_report_file)
81 print "...CALLING: handler.export_obj() with args {}".format(kwargs)
82 out = handler.export_obj(**kwargs)
83
84 # trim the output if it is more than 15 lines long
85 if len(out.splitlines()) > 15:
86     out = out.splitlines()[0:15]
87     out.append('..trimmed for brevity..')
88     out = '\n'.join(out)
89
90 print "...OUTPUT: print the export_str returned from export_obj():"
91 print out
```

## Export ResultSet CSV Sensor True

Export a ResultSet from asking a question as CSV with true for header\_add\_sensor

- STDOUT from Example Python Code
- STDERR from Example Python Code
- Example Python Code

```
1 # import the basic python packages we need
2 import os
3 import sys
4 import tempfile
5 import pprint
6 import traceback
7
8 # disable python from generating a .pyc file
9 sys.dont_write_bytecode = True
10
11 # change me to the path of pytan if this script is not running from EXAMPLES/PYTAN_API
12 pytan_loc = "~/gh/pytan"
13 pytan_static_path = os.path.join(os.path.expanduser(pytan_loc), 'lib')
14
15 # Determine our script name, script dir
16 my_file = os.path.abspath(sys.argv[0])
17 my_dir = os.path.dirname(my_file)
18
19 # try to automatically determine the pytan lib directory by assuming it is in '.././lib/'
20 parent_dir = os.path.dirname(my_dir)
21 pytan_root_dir = os.path.dirname(parent_dir)
22 lib_dir = os.path.join(pytan_root_dir, 'lib')
23
24 # add pytan_loc and lib_dir to the PYTHONPATH variable
25 path_adds = [lib_dir, pytan_static_path]
26 [sys.path.append(aa) for aa in path_adds if aa not in sys.path]
27
```



```

28 # import pytan
29 import pytan
30
31 # create a dictionary of arguments for the pytan handler
32 handler_args = {}
33
34 # establish our connection info for the Tanium Server
35 handler_args['username'] = "Administrator"
36 handler_args['password'] = "Tanium2015!"
37 handler_args['host'] = "10.0.1.240"
38 handler_args['port'] = "443" # optional
39
40 # optional, level 0 is no output except warnings/errors
41 # level 1 through 12 are more and more verbose
42 handler_args['loglevel'] = 1
43
44 # optional, use a debug format for the logging output (uses two lines per log entry)
45 handler_args['debugformat'] = False
46
47 # optional, this saves all response objects to handler.session.ALL_REQUESTS_RESPONSES
48 # very useful for capturing the full exchange of XML requests and responses
49 handler_args['record_all_requests'] = True
50
51 # instantiate a handler using all of the arguments in the handler_args dictionary
52 print "...CALLING: pytan.handler() with args: {}".format(handler_args)
53 handler = pytan.Handler(**handler_args)
54
55 # print out the handler string
56 print "...OUTPUT: handler string: {}".format(handler)
57
58 # setup the arguments for the handler() class
59 kwargs = {}
60 kwargs["export_format"] = u'csv'
61 kwargs["header_add_sensor"] = True
62
63 # setup the arguments for handler.ask()
64 ask_kwargs = {
65     'qtype': 'manual',
66     'sensors': [
67         "Computer Name", "IP Route Details", "IP Address",
68         'Folder Name Search with RegEx Match{dirname=Program Files,regex=.*Shared.*}',
69     ],
70 }
71
72 # ask the question that will provide the resultset that we want to use
73 print "...CALLING: handler.ask() with args {}".format(ask_kwargs)
74 response = handler.ask(**ask_kwargs)
75
76 # store the resultset object as the obj we want to export into kwargs
77 kwargs['obj'] = response['question_results']
78
79 # export the object to a string
80 # (we could just as easily export to a file using export_to_report_file)
81 print "...CALLING: handler.export_obj() with args {}".format(kwargs)
82 out = handler.export_obj(**kwargs)
83
84 # trim the output if it is more than 15 lines long
85 if len(out.splitlines()) > 15:

```

```
86     out = out.splitlines()[0:15]
87     out.append('..trimmed for brevity..')
88     out = '\n'.join(out)
89
90 print "...OUTPUT: print the export_str returned from export_obj():"
91 print out
```

## PyTan API Valid Get Object Examples

All of the PyTan API examples for Valid Get Object

### Get Action By Id

Get an action by id

- STDOUT from Example Python Code
- STDERR from Example Python Code
- Example Python Code

```
1  # import the basic python packages we need
2  import os
3  import sys
4  import tempfile
5  import pprint
6  import traceback
7
8  # disable python from generating a .pyc file
9  sys.dont_write_bytecode = True
10
11 # change me to the path of pytan if this script is not running from EXAMPLES/PYTAN_API
12 pytan_loc = "~/gh/pytan"
13 pytan_static_path = os.path.join(os.path.expanduser(pytan_loc), 'lib')
14
15 # Determine our script name, script dir
16 my_file = os.path.abspath(sys.argv[0])
17 my_dir = os.path.dirname(my_file)
18
19 # try to automatically determine the pytan lib directory by assuming it is in '.././lib/'
20 parent_dir = os.path.dirname(my_dir)
21 pytan_root_dir = os.path.dirname(parent_dir)
22 lib_dir = os.path.join(pytan_root_dir, 'lib')
23
24 # add pytan_loc and lib_dir to the PYTHONPATH variable
25 path_adds = [lib_dir, pytan_static_path]
26 [sys.path.append(aa) for aa in path_adds if aa not in sys.path]
27
28 # import pytan
29 import pytan
30
31 # create a dictionary of arguments for the pytan handler
32 handler_args = {}
33
34 # establish our connection info for the Tanium Server
35 handler_args['username'] = "Administrator"
36 handler_args['password'] = "Tanium2015!"
```

```

37 handler_args['host'] = "10.0.1.240"
38 handler_args['port'] = "443" # optional
39
40 # optional, level 0 is no output except warnings/errors
41 # level 1 through 12 are more and more verbose
42 handler_args['loglevel'] = 1
43
44 # optional, use a debug format for the logging output (uses two lines per log entry)
45 handler_args['debugformat'] = False
46
47 # optional, this saves all response objects to handler.session.ALL_REQUESTS_RESPONSES
48 # very useful for capturing the full exchange of XML requests and responses
49 handler_args['record_all_requests'] = True
50
51 # instantiate a handler using all of the arguments in the handler_args dictionary
52 print "...CALLING: pytan.handler() with args: {}".format(handler_args)
53 handler = pytan.Handler(**handler_args)
54
55 # print out the handler string
56 print "...OUTPUT: handler string: {}".format(handler)
57
58 # setup the arguments for the handler() class
59 kwargs = {}
60 kwargs["objtype"] = u'action'
61 kwargs["id"] = 1
62
63 print "...CALLING: handler.get with args: {}".format(kwargs)
64 response = handler.get(**kwargs)
65
66 print "...OUTPUT: Type of response: ", type(response)
67
68 print "...OUTPUT: print of response:"
69 print response
70
71 # call the export_obj() method to convert response to JSON and store it in out
72 export_kwargs = {}
73 export_kwargs['obj'] = response
74 export_kwargs['export_format'] = 'json'
75
76 print "...CALLING: handler.export_obj() with args {}".format(export_kwargs)
77 out = handler.export_obj(**export_kwargs)
78
79 # trim the output if it is more than 15 lines long
80 if len(out.splitlines()) > 15:
81     out = out.splitlines()[0:15]
82     out.append('..trimmed for brevity..')
83     out = '\n'.join(out)
84
85 print "...OUTPUT: print the objects returned in JSON format:"
86 print out

```

## Get Question By Id

Get a question by id

- STDOUT from Example Python Code

- STDERR from Example Python Code
- Example Python Code

```
1  # import the basic python packages we need
2  import os
3  import sys
4  import tempfile
5  import pprint
6  import traceback
7
8  # disable python from generating a .pyc file
9  sys.dont_write_bytecode = True
10
11  # change me to the path of pytan if this script is not running from EXAMPLES/PYTAN_API
12  pytan_loc = "~/gh/pytan"
13  pytan_static_path = os.path.join(os.path.expanduser(pytan_loc), 'lib')
14
15  # Determine our script name, script dir
16  my_file = os.path.abspath(sys.argv[0])
17  my_dir = os.path.dirname(my_file)
18
19  # try to automatically determine the pytan lib directory by assuming it is in '../..lib/'
20  parent_dir = os.path.dirname(my_dir)
21  pytan_root_dir = os.path.dirname(parent_dir)
22  lib_dir = os.path.join(pytan_root_dir, 'lib')
23
24  # add pytan_loc and lib_dir to the PYTHONPATH variable
25  path_adds = [lib_dir, pytan_static_path]
26  [sys.path.append(aa) for aa in path_adds if aa not in sys.path]
27
28  # import pytan
29  import pytan
30
31  # create a dictionary of arguments for the pytan handler
32  handler_args = {}
33
34  # establish our connection info for the Tanium Server
35  handler_args['username'] = "Administrator"
36  handler_args['password'] = "Tanium2015!"
37  handler_args['host'] = "10.0.1.240"
38  handler_args['port'] = "443" # optional
39
40  # optional, level 0 is no output except warnings/errors
41  # level 1 through 12 are more and more verbose
42  handler_args['loglevel'] = 1
43
44  # optional, use a debug format for the logging output (uses two lines per log entry)
45  handler_args['debugformat'] = False
46
47  # optional, this saves all response objects to handler.session.ALL_REQUESTS_RESPONSES
48  # very useful for capturing the full exchange of XML requests and responses
49  handler_args['record_all_requests'] = True
50
51  # instantiate a handler using all of the arguments in the handler_args dictionary
52  print "...CALLING: pytan.handler() with args: {}".format(handler_args)
53  handler = pytan.Handler(**handler_args)
54
55  # print out the handler string
```

```

56 print "...OUTPUT: handler string: {}".format(handler)
57
58 # setup the arguments for the handler() class
59 kwargs = {}
60 kwargs["objtype"] = u'question'
61 kwargs["id"] = 1
62
63 print "...CALLING: handler.get with args: {}".format(kwargs)
64 response = handler.get(**kwargs)
65
66 print "...OUTPUT: Type of response: ", type(response)
67
68 print "...OUTPUT: print of response:"
69 print response
70
71 # call the export_obj() method to convert response to JSON and store it in out
72 export_kwargs = {}
73 export_kwargs['obj'] = response
74 export_kwargs['export_format'] = 'json'
75
76 print "...CALLING: handler.export_obj() with args {}".format(export_kwargs)
77 out = handler.export_obj(**export_kwargs)
78
79 # trim the output if it is more than 15 lines long
80 if len(out.splitlines()) > 15:
81     out = out.splitlines()[0:15]
82     out.append('..trimmed for brevity..')
83     out = '\n'.join(out)
84
85 print "...OUTPUT: print the objects returned in JSON format:"
86 print out

```

### Get Saved Question By Names

Get two saved questions by name

- STDOUT from Example Python Code
- STDERR from Example Python Code
- Example Python Code

```

1 # import the basic python packages we need
2 import os
3 import sys
4 import tempfile
5 import pprint
6 import traceback
7
8 # disable python from generating a .pyc file
9 sys.dont_write_bytecode = True
10
11 # change me to the path of pytan if this script is not running from EXAMPLES/PYTAN_API
12 pytan_loc = "~/gh/pytan"
13 pytan_static_path = os.path.join(os.path.expanduser(pytan_loc), 'lib')
14
15 # Determine our script name, script dir
16 my_file = os.path.abspath(sys.argv[0])

```

```
17 my_dir = os.path.dirname(my_file)
18
19 # try to automatically determine the pytan lib directory by assuming it is in '../..lib/'
20 parent_dir = os.path.dirname(my_dir)
21 pytan_root_dir = os.path.dirname(parent_dir)
22 lib_dir = os.path.join(pytan_root_dir, 'lib')
23
24 # add pytan_loc and lib_dir to the PYTHONPATH variable
25 path_adds = [lib_dir, pytan_static_path]
26 [sys.path.append(aa) for aa in path_adds if aa not in sys.path]
27
28 # import pytan
29 import pytan
30
31 # create a dictionary of arguments for the pytan handler
32 handler_args = {}
33
34 # establish our connection info for the Tanium Server
35 handler_args['username'] = "Administrator"
36 handler_args['password'] = "Tanium2015!"
37 handler_args['host'] = "10.0.1.240"
38 handler_args['port'] = "443" # optional
39
40 # optional, level 0 is no output except warnings/errors
41 # level 1 through 12 are more and more verbose
42 handler_args['loglevel'] = 1
43
44 # optional, use a debug format for the logging output (uses two lines per log entry)
45 handler_args['debugformat'] = False
46
47 # optional, this saves all response objects to handler.session.ALL_REQUESTS_RESPONSES
48 # very useful for capturing the full exchange of XML requests and responses
49 handler_args['record_all_requests'] = True
50
51 # instantiate a handler using all of the arguments in the handler_args dictionary
52 print "...CALLING: pytan.handler() with args: {}".format(handler_args)
53 handler = pytan.Handler(**handler_args)
54
55 # print out the handler string
56 print "...OUTPUT: handler string: {}".format(handler)
57
58 # setup the arguments for the handler() class
59 kwargs = {}
60 kwargs["objtype"] = u'saved_question'
61 kwargs["name"] = [u'Installed Applications', u'Computer Name']
62
63 print "...CALLING: handler.get with args: {}".format(kwargs)
64 response = handler.get(**kwargs)
65
66 print "...OUTPUT: Type of response: ", type(response)
67
68 print "...OUTPUT: print of response:"
69 print response
70
71 # call the export_obj() method to convert response to JSON and store it in out
72 export_kwargs = {}
73 export_kwargs['obj'] = response
74 export_kwargs['export_format'] = 'json'
```

```

75
76 print "...CALLING: handler.export_obj() with args {}".format(export_kwargs)
77 out = handler.export_obj(**export_kwargs)
78
79 # trim the output if it is more than 15 lines long
80 if len(out.splitlines()) > 15:
81     out = out.splitlines()[0:15]
82     out.append('..trimmed for brevity..')
83     out = '\n'.join(out)
84
85 print "...OUTPUT: print the objects returned in JSON format:"
86 print out

```

## Get Userrole By Id

Get a user role by id.

- [STDOUT from Example Python Code](#)
- [STDERR from Example Python Code](#)
- [Example Python Code](#)

```

1  # import the basic python packages we need
2  import os
3  import sys
4  import tempfile
5  import pprint
6  import traceback
7
8  # disable python from generating a .pyc file
9  sys.dont_write_bytecode = True
10
11 # change me to the path of pytan if this script is not running from EXAMPLES/PYTAN_API
12 pytan_loc = "~/gh/pytan"
13 pytan_static_path = os.path.join(os.path.expanduser(pytan_loc), 'lib')
14
15 # Determine our script name, script dir
16 my_file = os.path.abspath(sys.argv[0])
17 my_dir = os.path.dirname(my_file)
18
19 # try to automatically determine the pytan lib directory by assuming it is in '../..lib/'
20 parent_dir = os.path.dirname(my_dir)
21 pytan_root_dir = os.path.dirname(parent_dir)
22 lib_dir = os.path.join(pytan_root_dir, 'lib')
23
24 # add pytan_loc and lib_dir to the PYTHONPATH variable
25 path_adds = [lib_dir, pytan_static_path]
26 [sys.path.append(aa) for aa in path_adds if aa not in sys.path]
27
28 # import pytan
29 import pytan
30
31 # create a dictionary of arguments for the pytan handler
32 handler_args = {}
33
34 # establish our connection info for the Tanium Server
35 handler_args['username'] = "Administrator"

```

```
36 handler_args['password'] = "Tanium2015!"
37 handler_args['host'] = "10.0.1.240"
38 handler_args['port'] = "443" # optional
39
40 # optional, level 0 is no output except warnings/errors
41 # level 1 through 12 are more and more verbose
42 handler_args['loglevel'] = 1
43
44 # optional, use a debug format for the logging output (uses two lines per log entry)
45 handler_args['debugformat'] = False
46
47 # optional, this saves all response objects to handler.session.ALL_REQUESTS_RESPONSES
48 # very useful for capturing the full exchange of XML requests and responses
49 handler_args['record_all_requests'] = True
50
51 # instantiate a handler using all of the arguments in the handler_args dictionary
52 print "...CALLING: pytan.handler() with args: {}".format(handler_args)
53 handler = pytan.Handler(**handler_args)
54
55 # print out the handler string
56 print "...OUTPUT: handler string: {}".format(handler)
57
58 # setup the arguments for the handler() class
59 kwargs = {}
60 kwargs["objtype"] = u'userrole'
61 kwargs["id"] = 1
62
63 print "...CALLING: handler.get with args: {}".format(kwargs)
64 response = handler.get(**kwargs)
65
66 print "...OUTPUT: Type of response: ", type(response)
67
68 print "...OUTPUT: print of response:"
69 print response
70
71 # call the export_obj() method to convert response to JSON and store it in out
72 export_kwargs = {}
73 export_kwargs['obj'] = response
74 export_kwargs['export_format'] = 'json'
75
76 print "...CALLING: handler.export_obj() with args {}".format(export_kwargs)
77 out = handler.export_obj(**export_kwargs)
78
79 # trim the output if it is more than 15 lines long
80 if len(out.splitlines()) > 15:
81     out = out.splitlines()[0:15]
82     out.append('..trimmed for brevity..')
83     out = '\n'.join(out)
84
85 print "...OUTPUT: print the objects returned in JSON format:"
86 print out
```

## Get Leader Clients

Get all clients that are Leader status

- STDOUT from Example Python Code



- STDERR from Example Python Code
- Example Python Code

```

1  # import the basic python packages we need
2  import os
3  import sys
4  import tempfile
5  import pprint
6  import traceback
7
8  # disable python from generating a .pyc file
9  sys.dont_write_bytecode = True
10
11 # change me to the path of pytan if this script is not running from EXAMPLES/PYTAN_API
12 pytan_loc = "~/gh/pytan"
13 pytan_static_path = os.path.join(os.path.expanduser(pytan_loc), 'lib')
14
15 # Determine our script name, script dir
16 my_file = os.path.abspath(sys.argv[0])
17 my_dir = os.path.dirname(my_file)
18
19 # try to automatically determine the pytan lib directory by assuming it is in '../..lib/'
20 parent_dir = os.path.dirname(my_dir)
21 pytan_root_dir = os.path.dirname(parent_dir)
22 lib_dir = os.path.join(pytan_root_dir, 'lib')
23
24 # add pytan_loc and lib_dir to the PYTHONPATH variable
25 path_adds = [lib_dir, pytan_static_path]
26 [sys.path.append(aa) for aa in path_adds if aa not in sys.path]
27
28 # import pytan
29 import pytan
30
31 # create a dictionary of arguments for the pytan handler
32 handler_args = {}
33
34 # establish our connection info for the Tanium Server
35 handler_args['username'] = "Administrator"
36 handler_args['password'] = "Tanium2015!"
37 handler_args['host'] = "10.0.1.240"
38 handler_args['port'] = "443" # optional
39
40 # optional, level 0 is no output except warnings/errors
41 # level 1 through 12 are more and more verbose
42 handler_args['loglevel'] = 1
43
44 # optional, use a debug format for the logging output (uses two lines per log entry)
45 handler_args['debugformat'] = False
46
47 # optional, this saves all response objects to handler.session.ALL_REQUESTS_RESPONSES
48 # very useful for capturing the full exchange of XML requests and responses
49 handler_args['record_all_requests'] = True
50
51 # instantiate a handler using all of the arguments in the handler_args dictionary
52 print "...CALLING: pytan.handler() with args: {}".format(handler_args)
53 handler = pytan.Handler(**handler_args)
54
55 # print out the handler string

```

```
56 print "...OUTPUT: handler string: {}".format(handler)
57
58 # setup the arguments for the handler() class
59 kwargs = {}
60 kwargs["objtype"] = u'client'
61 kwargs["status"] = u'Leader'
62
63 print "...CALLING: handler.get with args: {}".format(kwargs)
64 response = handler.get(**kwargs)
65
66 print "...OUTPUT: Type of response: ", type(response)
67
68 print "...OUTPUT: print of response:"
69 print response
70
71 # call the export_obj() method to convert response to JSON and store it in out
72 export_kwargs = {}
73 export_kwargs['obj'] = response
74 export_kwargs['export_format'] = 'json'
75
76 print "...CALLING: handler.export_obj() with args {}".format(export_kwargs)
77 out = handler.export_obj(**export_kwargs)
78
79 # trim the output if it is more than 15 lines long
80 if len(out.splitlines()) > 15:
81     out = out.splitlines()[0:15]
82     out.append('..trimmed for brevity..')
83     out = '\n'.join(out)
84
85 print "...OUTPUT: print the objects returned in JSON format:"
86 print out
```

## Get Setting By Name

Get a system setting by name

- [STDOUT from Example Python Code](#)
- [STDERR from Example Python Code](#)
- [Example Python Code](#)

```
1 # import the basic python packages we need
2 import os
3 import sys
4 import tempfile
5 import pprint
6 import traceback
7
8 # disable python from generating a .pyc file
9 sys.dont_write_bytecode = True
10
11 # change me to the path of pytan if this script is not running from EXAMPLES/PYTAN_API
12 pytan_loc = "~/gh/pytan"
13 pytan_static_path = os.path.join(os.path.expanduser(pytan_loc), 'lib')
14
15 # Determine our script name, script dir
16 my_file = os.path.abspath(sys.argv[0])
```

```

17 my_dir = os.path.dirname(my_file)
18
19 # try to automatically determine the pytan lib directory by assuming it is in '../..lib/'
20 parent_dir = os.path.dirname(my_dir)
21 pytan_root_dir = os.path.dirname(parent_dir)
22 lib_dir = os.path.join(pytan_root_dir, 'lib')
23
24 # add pytan_loc and lib_dir to the PYTHONPATH variable
25 path_adds = [lib_dir, pytan_static_path]
26 [sys.path.append(aa) for aa in path_adds if aa not in sys.path]
27
28 # import pytan
29 import pytan
30
31 # create a dictionary of arguments for the pytan handler
32 handler_args = {}
33
34 # establish our connection info for the Tanium Server
35 handler_args['username'] = "Administrator"
36 handler_args['password'] = "Tanium2015!"
37 handler_args['host'] = "10.0.1.240"
38 handler_args['port'] = "443" # optional
39
40 # optional, level 0 is no output except warnings/errors
41 # level 1 through 12 are more and more verbose
42 handler_args['loglevel'] = 1
43
44 # optional, use a debug format for the logging output (uses two lines per log entry)
45 handler_args['debugformat'] = False
46
47 # optional, this saves all response objects to handler.session.ALL_REQUESTS_RESPONSES
48 # very useful for capturing the full exchange of XML requests and responses
49 handler_args['record_all_requests'] = True
50
51 # instantiate a handler using all of the arguments in the handler_args dictionary
52 print "...CALLING: pytan.handler() with args: {}".format(handler_args)
53 handler = pytan.Handler(**handler_args)
54
55 # print out the handler string
56 print "...OUTPUT: handler string: {}".format(handler)
57
58 # setup the arguments for the handler() class
59 kwargs = {}
60 kwargs["objtype"] = u'setting'
61 kwargs["name"] = u'control_address'
62
63 print "...CALLING: handler.get with args: {}".format(kwargs)
64 response = handler.get(**kwargs)
65
66 print "...OUTPUT: Type of response: ", type(response)
67
68 print "...OUTPUT: print of response:"
69 print response
70
71 # call the export_obj() method to convert response to JSON and store it in out
72 export_kwargs = {}
73 export_kwargs['obj'] = response
74 export_kwargs['export_format'] = 'json'

```

```
75
76 print "...CALLING: handler.export_obj() with args {}".format(export_kwargs)
77 out = handler.export_obj(**export_kwargs)
78
79 # trim the output if it is more than 15 lines long
80 if len(out.splitlines()) > 15:
81     out = out.splitlines()[0:15]
82     out.append('..trimmed for brevity..')
83     out = '\n'.join(out)
84
85 print "...OUTPUT: print the objects returned in JSON format:"
86 print out
```

## Get User By Name

Get a user by name

- [STDOUT from Example Python Code](#)
- [STDERR from Example Python Code](#)
- [Example Python Code](#)

```
1 # import the basic python packages we need
2 import os
3 import sys
4 import tempfile
5 import pprint
6 import traceback
7
8 # disable python from generating a .pyc file
9 sys.dont_write_bytecode = True
10
11 # change me to the path of pytan if this script is not running from EXAMPLES/PYTAN_API
12 pytan_loc = "~/gh/pytan"
13 pytan_static_path = os.path.join(os.path.expanduser(pytan_loc), 'lib')
14
15 # Determine our script name, script dir
16 my_file = os.path.abspath(sys.argv[0])
17 my_dir = os.path.dirname(my_file)
18
19 # try to automatically determine the pytan lib directory by assuming it is in '../..lib/'
20 parent_dir = os.path.dirname(my_dir)
21 pytan_root_dir = os.path.dirname(parent_dir)
22 lib_dir = os.path.join(pytan_root_dir, 'lib')
23
24 # add pytan_loc and lib_dir to the PYTHONPATH variable
25 path_adds = [lib_dir, pytan_static_path]
26 [sys.path.append(aa) for aa in path_adds if aa not in sys.path]
27
28 # import pytan
29 import pytan
30
31 # create a dictionary of arguments for the pytan handler
32 handler_args = {}
33
34 # establish our connection info for the Tanium Server
35 handler_args['username'] = "Administrator"
```

```

36 handler_args['password'] = "Tanium2015!"
37 handler_args['host'] = "10.0.1.240"
38 handler_args['port'] = "443" # optional
39
40 # optional, level 0 is no output except warnings/errors
41 # level 1 through 12 are more and more verbose
42 handler_args['loglevel'] = 1
43
44 # optional, use a debug format for the logging output (uses two lines per log entry)
45 handler_args['debugformat'] = False
46
47 # optional, this saves all response objects to handler.session.ALL_REQUESTS_RESPONSES
48 # very useful for capturing the full exchange of XML requests and responses
49 handler_args['record_all_requests'] = True
50
51 # instantiate a handler using all of the arguments in the handler_args dictionary
52 print "...CALLING: pytan.handler() with args: {}".format(handler_args)
53 handler = pytan.Handler(**handler_args)
54
55 # print out the handler string
56 print "...OUTPUT: handler string: {}".format(handler)
57
58 # setup the arguments for the handler() class
59 kwargs = {}
60 kwargs["objtype"] = u'user'
61 kwargs["name"] = u'Administrator'
62
63 print "...CALLING: handler.get with args: {}".format(kwargs)
64 response = handler.get(**kwargs)
65
66 print "...OUTPUT: Type of response: ", type(response)
67
68 print "...OUTPUT: print of response:"
69 print response
70
71 # call the export_obj() method to convert response to JSON and store it in out
72 export_kwargs = {}
73 export_kwargs['obj'] = response
74 export_kwargs['export_format'] = 'json'
75
76 print "...CALLING: handler.export_obj() with args {}".format(export_kwargs)
77 out = handler.export_obj(**export_kwargs)
78
79 # trim the output if it is more than 15 lines long
80 if len(out.splitlines()) > 15:
81     out = out.splitlines()[0:15]
82     out.append('..trimmed for brevity..')
83     out = '\n'.join(out)
84
85 print "...OUTPUT: print the objects returned in JSON format:"
86 print out

```

## Get Sensor By Id

Get a sensor by id

- STDOUT from Example Python Code

- STDERR from Example Python Code
- Example Python Code

```
1  # import the basic python packages we need
2  import os
3  import sys
4  import tempfile
5  import pprint
6  import traceback
7
8  # disable python from generating a .pyc file
9  sys.dont_write_bytecode = True
10
11  # change me to the path of pytan if this script is not running from EXAMPLES/PYTAN_API
12  pytan_loc = "~/gh/pytan"
13  pytan_static_path = os.path.join(os.path.expanduser(pytan_loc), 'lib')
14
15  # Determine our script name, script dir
16  my_file = os.path.abspath(sys.argv[0])
17  my_dir = os.path.dirname(my_file)
18
19  # try to automatically determine the pytan lib directory by assuming it is in '../..lib/'
20  parent_dir = os.path.dirname(my_dir)
21  pytan_root_dir = os.path.dirname(parent_dir)
22  lib_dir = os.path.join(pytan_root_dir, 'lib')
23
24  # add pytan_loc and lib_dir to the PYTHONPATH variable
25  path_adds = [lib_dir, pytan_static_path]
26  [sys.path.append(aa) for aa in path_adds if aa not in sys.path]
27
28  # import pytan
29  import pytan
30
31  # create a dictionary of arguments for the pytan handler
32  handler_args = {}
33
34  # establish our connection info for the Tanium Server
35  handler_args['username'] = "Administrator"
36  handler_args['password'] = "Tanium2015!"
37  handler_args['host'] = "10.0.1.240"
38  handler_args['port'] = "443" # optional
39
40  # optional, level 0 is no output except warnings/errors
41  # level 1 through 12 are more and more verbose
42  handler_args['loglevel'] = 1
43
44  # optional, use a debug format for the logging output (uses two lines per log entry)
45  handler_args['debugformat'] = False
46
47  # optional, this saves all response objects to handler.session.ALL_REQUESTS_RESPONSES
48  # very useful for capturing the full exchange of XML requests and responses
49  handler_args['record_all_requests'] = True
50
51  # instantiate a handler using all of the arguments in the handler_args dictionary
52  print "...CALLING: pytan.handler() with args: {}".format(handler_args)
53  handler = pytan.Handler(**handler_args)
54
55  # print out the handler string
```

```

56 print "...OUTPUT: handler string: {}".format(handler)
57
58 # setup the arguments for the handler() class
59 kwargs = {}
60 kwargs["objtype"] = u'sensor'
61 kwargs["id"] = 1
62
63 print "...CALLING: handler.get with args: {}".format(kwargs)
64 response = handler.get(**kwargs)
65
66 print "...OUTPUT: Type of response: ", type(response)
67
68 print "...OUTPUT: print of response:"
69 print response
70
71 # call the export_obj() method to convert response to JSON and store it in out
72 export_kwargs = {}
73 export_kwargs['obj'] = response
74 export_kwargs['export_format'] = 'json'
75
76 print "...CALLING: handler.export_obj() with args {}".format(export_kwargs)
77 out = handler.export_obj(**export_kwargs)
78
79 # trim the output if it is more than 15 lines long
80 if len(out.splitlines()) > 15:
81     out = out.splitlines()[0:15]
82     out.append('..trimmed for brevity..')
83     out = '\n'.join(out)
84
85 print "...OUTPUT: print the objects returned in JSON format:"
86 print out

```

## Get Sensor By Mixed

Get multiple sensors by id, name, and hash

- STDOUT from Example Python Code
- STDERR from Example Python Code
- Example Python Code

```

1 # import the basic python packages we need
2 import os
3 import sys
4 import tempfile
5 import pprint
6 import traceback
7
8 # disable python from generating a .pyc file
9 sys.dont_write_bytecode = True
10
11 # change me to the path of pytan if this script is not running from EXAMPLES/PYTAN_API
12 pytan_loc = "~/gh/pytan"
13 pytan_static_path = os.path.join(os.path.expanduser(pytan_loc), 'lib')
14
15 # Determine our script name, script dir
16 my_file = os.path.abspath(sys.argv[0])

```

```
17 my_dir = os.path.dirname(my_file)
18
19 # try to automatically determine the pytan lib directory by assuming it is in '../..lib/'
20 parent_dir = os.path.dirname(my_dir)
21 pytan_root_dir = os.path.dirname(parent_dir)
22 lib_dir = os.path.join(pytan_root_dir, 'lib')
23
24 # add pytan_loc and lib_dir to the PYTHONPATH variable
25 path_adds = [lib_dir, pytan_static_path]
26 [sys.path.append(aa) for aa in path_adds if aa not in sys.path]
27
28 # import pytan
29 import pytan
30
31 # create a dictionary of arguments for the pytan handler
32 handler_args = {}
33
34 # establish our connection info for the Tanium Server
35 handler_args['username'] = "Administrator"
36 handler_args['password'] = "Tanium2015!"
37 handler_args['host'] = "10.0.1.240"
38 handler_args['port'] = "443" # optional
39
40 # optional, level 0 is no output except warnings/errors
41 # level 1 through 12 are more and more verbose
42 handler_args['loglevel'] = 1
43
44 # optional, use a debug format for the logging output (uses two lines per log entry)
45 handler_args['debugformat'] = False
46
47 # optional, this saves all response objects to handler.session.ALL_REQUESTS_RESPONSES
48 # very useful for capturing the full exchange of XML requests and responses
49 handler_args['record_all_requests'] = True
50
51 # instantiate a handler using all of the arguments in the handler_args dictionary
52 print "...CALLING: pytan.handler() with args: {}".format(handler_args)
53 handler = pytan.Handler(**handler_args)
54
55 # print out the handler string
56 print "...OUTPUT: handler string: {}".format(handler)
57
58 # setup the arguments for the handler() class
59 kwargs = {}
60 kwargs["objtype"] = u'sensor'
61 kwargs["hash"] = [u'322086833']
62 kwargs["name"] = [u'Computer Name']
63 kwargs["id"] = [1, 2]
64
65 print "...CALLING: handler.get with args: {}".format(kwargs)
66 response = handler.get(**kwargs)
67
68 print "...OUTPUT: Type of response: ", type(response)
69
70 print "...OUTPUT: print of response:"
71 print response
72
73 # call the export_obj() method to convert response to JSON and store it in out
74 export_kwargs = {}
```



```

75 export_kwargs['obj'] = response
76 export_kwargs['export_format'] = 'json'
77
78 print "...CALLING: handler.export_obj() with args {}".format(export_kwargs)
79 out = handler.export_obj(**export_kwargs)
80
81 # trim the output if it is more than 15 lines long
82 if len(out.splitlines()) > 15:
83     out = out.splitlines()[0:15]
84     out.append('..trimmed for brevity..')
85     out = '\n'.join(out)
86
87 print "...OUTPUT: print the objects returned in JSON format:"
88 print out

```

## Get Whitelisted Url By Id

Get a whitelisted url by id

- STDOUT from Example Python Code
- STDERR from Example Python Code
- Example Python Code

```

1  # import the basic python packages we need
2  import os
3  import sys
4  import tempfile
5  import pprint
6  import traceback
7
8  # disable python from generating a .pyc file
9  sys.dont_write_bytecode = True
10
11 # change me to the path of pytan if this script is not running from EXAMPLES/PYTAN_API
12 pytan_loc = "~/gh/pytan"
13 pytan_static_path = os.path.join(os.path.expanduser(pytan_loc), 'lib')
14
15 # Determine our script name, script dir
16 my_file = os.path.abspath(sys.argv[0])
17 my_dir = os.path.dirname(my_file)
18
19 # try to automatically determine the pytan lib directory by assuming it is in '../..lib/'
20 parent_dir = os.path.dirname(my_dir)
21 pytan_root_dir = os.path.dirname(parent_dir)
22 lib_dir = os.path.join(pytan_root_dir, 'lib')
23
24 # add pytan_loc and lib_dir to the PYTHONPATH variable
25 path_adds = [lib_dir, pytan_static_path]
26 [sys.path.append(aa) for aa in path_adds if aa not in sys.path]
27
28 # import pytan
29 import pytan
30
31 # create a dictionary of arguments for the pytan handler
32 handler_args = {}
33

```

```
34 # establish our connection info for the Tanium Server
35 handler_args['username'] = "Administrator"
36 handler_args['password'] = "Tanium2015!"
37 handler_args['host'] = "10.0.1.240"
38 handler_args['port'] = "443" # optional
39
40 # optional, level 0 is no output except warnings/errors
41 # level 1 through 12 are more and more verbose
42 handler_args['loglevel'] = 1
43
44 # optional, use a debug format for the logging output (uses two lines per log entry)
45 handler_args['debugformat'] = False
46
47 # optional, this saves all response objects to handler.session.ALL_REQUESTS_RESPONSES
48 # very useful for capturing the full exchange of XML requests and responses
49 handler_args['record_all_requests'] = True
50
51 # instantiate a handler using all of the arguments in the handler_args dictionary
52 print "...CALLING: pytan.handler() with args: {}".format(handler_args)
53 handler = pytan.Handler(**handler_args)
54
55 # print out the handler string
56 print "...OUTPUT: handler string: {}".format(handler)
57
58 # setup the arguments for the handler() class
59 kwargs = {}
60 kwargs["objtype"] = u'whitelisted_url'
61 kwargs["id"] = 1
62
63 print "...CALLING: handler.get with args: {}".format(kwargs)
64 response = handler.get(**kwargs)
65
66 print "...OUTPUT: Type of response: ", type(response)
67
68 print "...OUTPUT: print of response:"
69 print response
70
71 # call the export_obj() method to convert response to JSON and store it in out
72 export_kwargs = {}
73 export_kwargs['obj'] = response
74 export_kwargs['export_format'] = 'json'
75
76 print "...CALLING: handler.export_obj() with args {}".format(export_kwargs)
77 out = handler.export_obj(**export_kwargs)
78
79 # trim the output if it is more than 15 lines long
80 if len(out.splitlines()) > 15:
81     out = out.splitlines()[0:15]
82     out.append('..trimmed for brevity..')
83     out = '\n'.join(out)
84
85 print "...OUTPUT: print the objects returned in JSON format:"
86 print out
```

## Get Group By Name

Get a group by name

- STDOUT from Example Python Code
- STDERR from Example Python Code
- Example Python Code

```

1  # import the basic python packages we need
2  import os
3  import sys
4  import tempfile
5  import pprint
6  import traceback
7
8  # disable python from generating a .pyc file
9  sys.dont_write_bytecode = True
10
11 # change me to the path of pytan if this script is not running from EXAMPLES/PYTAN_API
12 pytan_loc = "~/gh/pytan"
13 pytan_static_path = os.path.join(os.path.expanduser(pytan_loc), 'lib')
14
15 # Determine our script name, script dir
16 my_file = os.path.abspath(sys.argv[0])
17 my_dir = os.path.dirname(my_file)
18
19 # try to automatically determine the pytan lib directory by assuming it is in '../..lib/'
20 parent_dir = os.path.dirname(my_dir)
21 pytan_root_dir = os.path.dirname(parent_dir)
22 lib_dir = os.path.join(pytan_root_dir, 'lib')
23
24 # add pytan_loc and lib_dir to the PYTHONPATH variable
25 path_adds = [lib_dir, pytan_static_path]
26 [sys.path.append(aa) for aa in path_adds if aa not in sys.path]
27
28 # import pytan
29 import pytan
30
31 # create a dictionary of arguments for the pytan handler
32 handler_args = {}
33
34 # establish our connection info for the Tanium Server
35 handler_args['username'] = "Administrator"
36 handler_args['password'] = "Tanium2015!"
37 handler_args['host'] = "10.0.1.240"
38 handler_args['port'] = "443" # optional
39
40 # optional, level 0 is no output except warnings/errors
41 # level 1 through 12 are more and more verbose
42 handler_args['loglevel'] = 1
43
44 # optional, use a debug format for the logging output (uses two lines per log entry)
45 handler_args['debugformat'] = False
46
47 # optional, this saves all response objects to handler.session.ALL_REQUESTS_RESPONSES
48 # very useful for capturing the full exchange of XML requests and responses
49 handler_args['record_all_requests'] = True
50
51 # instantiate a handler using all of the arguments in the handler_args dictionary
52 print "...CALLING: pytan.handler() with args: {}".format(handler_args)
53 handler = pytan.Handler(**handler_args)
54

```

```
55 # print out the handler string
56 print "...OUTPUT: handler string: {}".format(handler)
57
58 # setup the arguments for the handler() class
59 kwargs = {}
60 kwargs["objtype"] = u'group'
61 kwargs["name"] = u'All Computers'
62
63 print "...CALLING: handler.get with args: {}".format(kwargs)
64 response = handler.get(**kwargs)
65
66 print "...OUTPUT: Type of response: ", type(response)
67
68 print "...OUTPUT: print of response:"
69 print response
70
71 # call the export_obj() method to convert response to JSON and store it in out
72 export_kwargs = {}
73 export_kwargs['obj'] = response
74 export_kwargs['export_format'] = 'json'
75
76 print "...CALLING: handler.export_obj() with args {}".format(export_kwargs)
77 out = handler.export_obj(**export_kwargs)
78
79 # trim the output if it is more than 15 lines long
80 if len(out.splitlines()) > 15:
81     out = out.splitlines()[0:15]
82     out.append('..trimmed for brevity..')
83     out = '\n'.join(out)
84
85 print "...OUTPUT: print the objects returned in JSON format:"
86 print out
```

## Get Sensor By Hash

Get a sensor by hash

- [STDOUT from Example Python Code](#)
- [STDERR from Example Python Code](#)
- [Example Python Code](#)

```
1 # import the basic python packages we need
2 import os
3 import sys
4 import tempfile
5 import pprint
6 import traceback
7
8 # disable python from generating a .pyc file
9 sys.dont_write_bytecode = True
10
11 # change me to the path of pytan if this script is not running from EXAMPLES/PYTAN_API
12 pytan_loc = "~/gh/pytan"
13 pytan_static_path = os.path.join(os.path.expanduser(pytan_loc), 'lib')
14
15 # Determine our script name, script dir
```

```

16 my_file = os.path.abspath(sys.argv[0])
17 my_dir = os.path.dirname(my_file)
18
19 # try to automatically determine the pytan lib directory by assuming it is in '../..lib/'
20 parent_dir = os.path.dirname(my_dir)
21 pytan_root_dir = os.path.dirname(parent_dir)
22 lib_dir = os.path.join(pytan_root_dir, 'lib')
23
24 # add pytan_loc and lib_dir to the PYTHONPATH variable
25 path_adds = [lib_dir, pytan_static_path]
26 [sys.path.append(aa) for aa in path_adds if aa not in sys.path]
27
28 # import pytan
29 import pytan
30
31 # create a dictionary of arguments for the pytan handler
32 handler_args = {}
33
34 # establish our connection info for the Tanium Server
35 handler_args['username'] = "Administrator"
36 handler_args['password'] = "Tanium2015!"
37 handler_args['host'] = "10.0.1.240"
38 handler_args['port'] = "443" # optional
39
40 # optional, level 0 is no output except warnings/errors
41 # level 1 through 12 are more and more verbose
42 handler_args['loglevel'] = 1
43
44 # optional, use a debug format for the logging output (uses two lines per log entry)
45 handler_args['debugformat'] = False
46
47 # optional, this saves all response objects to handler.session.ALL_REQUESTS_RESPONSES
48 # very useful for capturing the full exchange of XML requests and responses
49 handler_args['record_all_requests'] = True
50
51 # instantiate a handler using all of the arguments in the handler_args dictionary
52 print "...CALLING: pytan.handler() with args: {}".format(handler_args)
53 handler = pytan.Handler(**handler_args)
54
55 # print out the handler string
56 print "...OUTPUT: handler string: {}".format(handler)
57
58 # setup the arguments for the handler() class
59 kwargs = {}
60 kwargs["objtype"] = u'sensor'
61 kwargs["hash"] = u'322086833'
62
63 print "...CALLING: handler.get with args: {}".format(kwargs)
64 response = handler.get(**kwargs)
65
66 print "...OUTPUT: Type of response: ", type(response)
67
68 print "...OUTPUT: print of response:"
69 print response
70
71 # call the export_obj() method to convert response to JSON and store it in out
72 export_kwargs = {}
73 export_kwargs['obj'] = response

```

```
74 export_kwargs['export_format'] = 'json'
75
76 print "...CALLING: handler.export_obj() with args {}".format(export_kwargs)
77 out = handler.export_obj(**export_kwargs)
78
79 # trim the output if it is more than 15 lines long
80 if len(out.splitlines()) > 15:
81     out = out.splitlines()[0:15]
82     out.append('..trimmed for brevity..')
83     out = '\n'.join(out)
84
85 print "...OUTPUT: print the objects returned in JSON format:"
86 print out
```

## Get Package By Name

Get a package by name

- [STDOUT from Example Python Code](#)
- [STDERR from Example Python Code](#)
- [Example Python Code](#)

```
1  # import the basic python packages we need
2  import os
3  import sys
4  import tempfile
5  import pprint
6  import traceback
7
8  # disable python from generating a .pyc file
9  sys.dont_write_bytecode = True
10
11 # change me to the path of pytan if this script is not running from EXAMPLES/PYTAN_API
12 pytan_loc = "~/gh/pytan"
13 pytan_static_path = os.path.join(os.path.expanduser(pytan_loc), 'lib')
14
15 # Determine our script name, script dir
16 my_file = os.path.abspath(sys.argv[0])
17 my_dir = os.path.dirname(my_file)
18
19 # try to automatically determine the pytan lib directory by assuming it is in '.././lib/'
20 parent_dir = os.path.dirname(my_dir)
21 pytan_root_dir = os.path.dirname(parent_dir)
22 lib_dir = os.path.join(pytan_root_dir, 'lib')
23
24 # add pytan_loc and lib_dir to the PYTHONPATH variable
25 path_adds = [lib_dir, pytan_static_path]
26 [sys.path.append(aa) for aa in path_adds if aa not in sys.path]
27
28 # import pytan
29 import pytan
30
31 # create a dictionary of arguments for the pytan handler
32 handler_args = {}
33
34 # establish our connection info for the Tanium Server
```

```

35 handler_args['username'] = "Administrator"
36 handler_args['password'] = "Tanium2015!"
37 handler_args['host'] = "10.0.1.240"
38 handler_args['port'] = "443" # optional
39
40 # optional, level 0 is no output except warnings/errors
41 # level 1 through 12 are more and more verbose
42 handler_args['loglevel'] = 1
43
44 # optional, use a debug format for the logging output (uses two lines per log entry)
45 handler_args['debugformat'] = False
46
47 # optional, this saves all response objects to handler.session.ALL_REQUESTS_RESPONSES
48 # very useful for capturing the full exchange of XML requests and responses
49 handler_args['record_all_requests'] = True
50
51 # instantiate a handler using all of the arguments in the handler_args dictionary
52 print "...CALLING: pytan.handler() with args: {}".format(handler_args)
53 handler = pytan.Handler(**handler_args)
54
55 # print out the handler string
56 print "...OUTPUT: handler string: {}".format(handler)
57
58 # setup the arguments for the handler() class
59 kwargs = {}
60 kwargs["objtype"] = u'package'
61 kwargs["name"] = u'Distribute Tanium Standard Utilities'
62
63 print "...CALLING: handler.get with args: {}".format(kwargs)
64 response = handler.get(**kwargs)
65
66 print "...OUTPUT: Type of response: ", type(response)
67
68 print "...OUTPUT: print of response:"
69 print response
70
71 # call the export_obj() method to convert response to JSON and store it in out
72 export_kwargs = {}
73 export_kwargs['obj'] = response
74 export_kwargs['export_format'] = 'json'
75
76 print "...CALLING: handler.export_obj() with args {}".format(export_kwargs)
77 out = handler.export_obj(**export_kwargs)
78
79 # trim the output if it is more than 15 lines long
80 if len(out.splitlines()) > 15:
81     out = out.splitlines()[0:15]
82     out.append('...trimmed for brevity..')
83     out = '\n'.join(out)
84
85 print "...OUTPUT: print the objects returned in JSON format:"
86 print out

```

## Get Sensor By Names

Get multiple sensors by name

- STDOUT from Example Python Code
- STDERR from Example Python Code
- Example Python Code

```
1  # import the basic python packages we need
2  import os
3  import sys
4  import tempfile
5  import pprint
6  import traceback
7
8  # disable python from generating a .pyc file
9  sys.dont_write_bytecode = True
10
11  # change me to the path of pytan if this script is not running from EXAMPLES/PYTAN_API
12  pytan_loc = "~/gh/pytan"
13  pytan_static_path = os.path.join(os.path.expanduser(pytan_loc), 'lib')
14
15  # Determine our script name, script dir
16  my_file = os.path.abspath(sys.argv[0])
17  my_dir = os.path.dirname(my_file)
18
19  # try to automatically determine the pytan lib directory by assuming it is in '../..lib/'
20  parent_dir = os.path.dirname(my_dir)
21  pytan_root_dir = os.path.dirname(parent_dir)
22  lib_dir = os.path.join(pytan_root_dir, 'lib')
23
24  # add pytan_loc and lib_dir to the PYTHONPATH variable
25  path_adds = [lib_dir, pytan_static_path]
26  [sys.path.append(aa) for aa in path_adds if aa not in sys.path]
27
28  # import pytan
29  import pytan
30
31  # create a dictionary of arguments for the pytan handler
32  handler_args = {}
33
34  # establish our connection info for the Tanium Server
35  handler_args['username'] = "Administrator"
36  handler_args['password'] = "Tanium2015!"
37  handler_args['host'] = "10.0.1.240"
38  handler_args['port'] = "443" # optional
39
40  # optional, level 0 is no output except warnings/errors
41  # level 1 through 12 are more and more verbose
42  handler_args['loglevel'] = 1
43
44  # optional, use a debug format for the logging output (uses two lines per log entry)
45  handler_args['debugformat'] = False
46
47  # optional, this saves all response objects to handler.session.ALL_REQUESTS_RESPONSES
48  # very useful for capturing the full exchange of XML requests and responses
49  handler_args['record_all_requests'] = True
50
51  # instantiate a handler using all of the arguments in the handler_args dictionary
52  print "...CALLING: pytan.handler() with args: {}".format(handler_args)
53  handler = pytan.Handler(**handler_args)
54
```



```

55 # print out the handler string
56 print "...OUTPUT: handler string: {}".format(handler)
57
58 # setup the arguments for the handler() class
59 kwargs = {}
60 kwargs["objtype"] = u'sensor'
61 kwargs["name"] = [u'Computer Name', u'Action Statuses']
62
63 print "...CALLING: handler.get with args: {}".format(kwargs)
64 response = handler.get(**kwargs)
65
66 print "...OUTPUT: Type of response: ", type(response)
67
68 print "...OUTPUT: print of response:"
69 print response
70
71 # call the export_obj() method to convert response to JSON and store it in out
72 export_kwargs = {}
73 export_kwargs['obj'] = response
74 export_kwargs['export_format'] = 'json'
75
76 print "...CALLING: handler.export_obj() with args {}".format(export_kwargs)
77 out = handler.export_obj(**export_kwargs)
78
79 # trim the output if it is more than 15 lines long
80 if len(out.splitlines()) > 15:
81     out = out.splitlines()[0:15]
82     out.append('..trimmed for brevity..')
83     out = '\n'.join(out)
84
85 print "...OUTPUT: print the objects returned in JSON format:"
86 print out

```

## Get Saved Question By Name

Get saved question by name

- STDOUT from Example Python Code
- STDERR from Example Python Code
- Example Python Code

```

1 # import the basic python packages we need
2 import os
3 import sys
4 import tempfile
5 import pprint
6 import traceback
7
8 # disable python from generating a .pyc file
9 sys.dont_write_bytecode = True
10
11 # change me to the path of pytan if this script is not running from EXAMPLES/PYTAN_API
12 pytan_loc = "~/gh/pytan"
13 pytan_static_path = os.path.join(os.path.expanduser(pytan_loc), 'lib')
14
15 # Determine our script name, script dir

```

```
16 my_file = os.path.abspath(sys.argv[0])
17 my_dir = os.path.dirname(my_file)
18
19 # try to automatically determine the pytan lib directory by assuming it is in '../..lib/'
20 parent_dir = os.path.dirname(my_dir)
21 pytan_root_dir = os.path.dirname(parent_dir)
22 lib_dir = os.path.join(pytan_root_dir, 'lib')
23
24 # add pytan_loc and lib_dir to the PYTHONPATH variable
25 path_adds = [lib_dir, pytan_static_path]
26 [sys.path.append(aa) for aa in path_adds if aa not in sys.path]
27
28 # import pytan
29 import pytan
30
31 # create a dictionary of arguments for the pytan handler
32 handler_args = {}
33
34 # establish our connection info for the Tanium Server
35 handler_args['username'] = "Administrator"
36 handler_args['password'] = "Tanium2015!"
37 handler_args['host'] = "10.0.1.240"
38 handler_args['port'] = "443" # optional
39
40 # optional, level 0 is no output except warnings/errors
41 # level 1 through 12 are more and more verbose
42 handler_args['loglevel'] = 1
43
44 # optional, use a debug format for the logging output (uses two lines per log entry)
45 handler_args['debugformat'] = False
46
47 # optional, this saves all response objects to handler.session.ALL_REQUESTS_RESPONSES
48 # very useful for capturing the full exchange of XML requests and responses
49 handler_args['record_all_requests'] = True
50
51 # instantiate a handler using all of the arguments in the handler_args dictionary
52 print "...CALLING: pytan.handler() with args: {}".format(handler_args)
53 handler = pytan.Handler(**handler_args)
54
55 # print out the handler string
56 print "...OUTPUT: handler string: {}".format(handler)
57
58 # setup the arguments for the handler() class
59 kwargs = {}
60 kwargs["objtype"] = u'saved_question'
61 kwargs["name"] = u'Installed Applications'
62
63 print "...CALLING: handler.get with args: {}".format(kwargs)
64 response = handler.get(**kwargs)
65
66 print "...OUTPUT: Type of response: ", type(response)
67
68 print "...OUTPUT: print of response:"
69 print response
70
71 # call the export_obj() method to convert response to JSON and store it in out
72 export_kwargs = {}
73 export_kwargs['obj'] = response
```

```

74 export_kwargs['export_format'] = 'json'
75
76 print "...CALLING: handler.export_obj() with args {}".format(export_kwargs)
77 out = handler.export_obj(**export_kwargs)
78
79 # trim the output if it is more than 15 lines long
80 if len(out.splitlines()) > 15:
81     out = out.splitlines()[0:15]
82     out.append('..trimmed for brevity..')
83     out = '\n'.join(out)
84
85 print "...OUTPUT: print the objects returned in JSON format:"
86 print out

```

## Get User By Id

Get a user by id

- [STDOUT from Example Python Code](#)
- [STDERR from Example Python Code](#)
- [Example Python Code](#)

```

1  # import the basic python packages we need
2  import os
3  import sys
4  import tempfile
5  import pprint
6  import traceback
7
8  # disable python from generating a .pyc file
9  sys.dont_write_bytecode = True
10
11 # change me to the path of pytan if this script is not running from EXAMPLES/PYTAN_API
12 pytan_loc = "~/gh/pytan"
13 pytan_static_path = os.path.join(os.path.expanduser(pytan_loc), 'lib')
14
15 # Determine our script name, script dir
16 my_file = os.path.abspath(sys.argv[0])
17 my_dir = os.path.dirname(my_file)
18
19 # try to automatically determine the pytan lib directory by assuming it is in '../..lib/'
20 parent_dir = os.path.dirname(my_dir)
21 pytan_root_dir = os.path.dirname(parent_dir)
22 lib_dir = os.path.join(pytan_root_dir, 'lib')
23
24 # add pytan_loc and lib_dir to the PYTHONPATH variable
25 path_adds = [lib_dir, pytan_static_path]
26 [sys.path.append(aa) for aa in path_adds if aa not in sys.path]
27
28 # import pytan
29 import pytan
30
31 # create a dictionary of arguments for the pytan handler
32 handler_args = {}
33
34 # establish our connection info for the Tanium Server

```

```
35 handler_args['username'] = "Administrator"
36 handler_args['password'] = "Tanium2015!"
37 handler_args['host'] = "10.0.1.240"
38 handler_args['port'] = "443" # optional
39
40 # optional, level 0 is no output except warnings/errors
41 # level 1 through 12 are more and more verbose
42 handler_args['loglevel'] = 1
43
44 # optional, use a debug format for the logging output (uses two lines per log entry)
45 handler_args['debugformat'] = False
46
47 # optional, this saves all response objects to handler.session.ALL_REQUESTS_RESPONSES
48 # very useful for capturing the full exchange of XML requests and responses
49 handler_args['record_all_requests'] = True
50
51 # instantiate a handler using all of the arguments in the handler_args dictionary
52 print "...CALLING: pytan.handler() with args: {}".format(handler_args)
53 handler = pytan.Handler(**handler_args)
54
55 # print out the handler string
56 print "...OUTPUT: handler string: {}".format(handler)
57
58 # setup the arguments for the handler() class
59 kwargs = {}
60 kwargs["objtype"] = u'user'
61 kwargs["id"] = 1
62
63 print "...CALLING: handler.get with args: {}".format(kwargs)
64 response = handler.get(**kwargs)
65
66 print "...OUTPUT: Type of response: ", type(response)
67
68 print "...OUTPUT: print of response:"
69 print response
70
71 # call the export_obj() method to convert response to JSON and store it in out
72 export_kwargs = {}
73 export_kwargs['obj'] = response
74 export_kwargs['export_format'] = 'json'
75
76 print "...CALLING: handler.export_obj() with args {}".format(export_kwargs)
77 out = handler.export_obj(**export_kwargs)
78
79 # trim the output if it is more than 15 lines long
80 if len(out.splitlines()) > 15:
81     out = out.splitlines()[0:15]
82     out.append('...trimmed for brevity..')
83     out = '\n'.join(out)
84
85 print "...OUTPUT: print the objects returned in JSON format:"
86 print out
```

## Get Sensor By Name

Get a sensor by name

- STDOUT from Example Python Code
- STDERR from Example Python Code
- Example Python Code

```

1  # import the basic python packages we need
2  import os
3  import sys
4  import tempfile
5  import pprint
6  import traceback
7
8  # disable python from generating a .pyc file
9  sys.dont_write_bytecode = True
10
11 # change me to the path of pytan if this script is not running from EXAMPLES/PYTAN_API
12 pytan_loc = "~/gh/pytan"
13 pytan_static_path = os.path.join(os.path.expanduser(pytan_loc), 'lib')
14
15 # Determine our script name, script dir
16 my_file = os.path.abspath(sys.argv[0])
17 my_dir = os.path.dirname(my_file)
18
19 # try to automatically determine the pytan lib directory by assuming it is in '../..lib/'
20 parent_dir = os.path.dirname(my_dir)
21 pytan_root_dir = os.path.dirname(parent_dir)
22 lib_dir = os.path.join(pytan_root_dir, 'lib')
23
24 # add pytan_loc and lib_dir to the PYTHONPATH variable
25 path_adds = [lib_dir, pytan_static_path]
26 [sys.path.append(aa) for aa in path_adds if aa not in sys.path]
27
28 # import pytan
29 import pytan
30
31 # create a dictionary of arguments for the pytan handler
32 handler_args = {}
33
34 # establish our connection info for the Tanium Server
35 handler_args['username'] = "Administrator"
36 handler_args['password'] = "Tanium2015!"
37 handler_args['host'] = "10.0.1.240"
38 handler_args['port'] = "443" # optional
39
40 # optional, level 0 is no output except warnings/errors
41 # level 1 through 12 are more and more verbose
42 handler_args['loglevel'] = 1
43
44 # optional, use a debug format for the logging output (uses two lines per log entry)
45 handler_args['debugformat'] = False
46
47 # optional, this saves all response objects to handler.session.ALL_REQUESTS_RESPONSES
48 # very useful for capturing the full exchange of XML requests and responses
49 handler_args['record_all_requests'] = True
50
51 # instantiate a handler using all of the arguments in the handler_args dictionary
52 print "...CALLING: pytan.handler() with args: {}".format(handler_args)
53 handler = pytan.Handler(**handler_args)
54

```

```
55 # print out the handler string
56 print "...OUTPUT: handler string: {}".format(handler)
57
58 # setup the arguments for the handler() class
59 kwargs = {}
60 kwargs["objtype"] = u'sensor'
61 kwargs["name"] = u'Computer Name'
62
63 print "...CALLING: handler.get with args: {}".format(kwargs)
64 response = handler.get(**kwargs)
65
66 print "...OUTPUT: Type of response: ", type(response)
67
68 print "...OUTPUT: print of response:"
69 print response
70
71 # call the export_obj() method to convert response to JSON and store it in out
72 export_kwargs = {}
73 export_kwargs['obj'] = response
74 export_kwargs['export_format'] = 'json'
75
76 print "...CALLING: handler.export_obj() with args {}".format(export_kwargs)
77 out = handler.export_obj(**export_kwargs)
78
79 # trim the output if it is more than 15 lines long
80 if len(out.splitlines()) > 15:
81     out = out.splitlines()[0:15]
82     out.append('..trimmed for brevity..')
83     out = '\n'.join(out)
84
85 print "...OUTPUT: print the objects returned in JSON format:"
86 print out
```

## Get Saved Action By Name

Get a saved action by name

- [STDOUT from Example Python Code](#)
- [STDERR from Example Python Code](#)
- [Example Python Code](#)

```
1 # import the basic python packages we need
2 import os
3 import sys
4 import tempfile
5 import pprint
6 import traceback
7
8 # disable python from generating a .pyc file
9 sys.dont_write_bytecode = True
10
11 # change me to the path of pytan if this script is not running from EXAMPLES/PYTAN_API
12 pytan_loc = "~/gh/pytan"
13 pytan_static_path = os.path.join(os.path.expanduser(pytan_loc), 'lib')
14
15 # Determine our script name, script dir
```

```

16 my_file = os.path.abspath(sys.argv[0])
17 my_dir = os.path.dirname(my_file)
18
19 # try to automatically determine the pytan lib directory by assuming it is in '../..lib/'
20 parent_dir = os.path.dirname(my_dir)
21 pytan_root_dir = os.path.dirname(parent_dir)
22 lib_dir = os.path.join(pytan_root_dir, 'lib')
23
24 # add pytan_loc and lib_dir to the PYTHONPATH variable
25 path_adds = [lib_dir, pytan_static_path]
26 [sys.path.append(aa) for aa in path_adds if aa not in sys.path]
27
28 # import pytan
29 import pytan
30
31 # create a dictionary of arguments for the pytan handler
32 handler_args = {}
33
34 # establish our connection info for the Tanium Server
35 handler_args['username'] = "Administrator"
36 handler_args['password'] = "Tanium2015!"
37 handler_args['host'] = "10.0.1.240"
38 handler_args['port'] = "443" # optional
39
40 # optional, level 0 is no output except warnings/errors
41 # level 1 through 12 are more and more verbose
42 handler_args['loglevel'] = 1
43
44 # optional, use a debug format for the logging output (uses two lines per log entry)
45 handler_args['debugformat'] = False
46
47 # optional, this saves all response objects to handler.session.ALL_REQUESTS_RESPONSES
48 # very useful for capturing the full exchange of XML requests and responses
49 handler_args['record_all_requests'] = True
50
51 # instantiate a handler using all of the arguments in the handler_args dictionary
52 print "...CALLING: pytan.handler() with args: {}".format(handler_args)
53 handler = pytan.Handler(**handler_args)
54
55 # print out the handler string
56 print "...OUTPUT: handler string: {}".format(handler)
57
58 # setup the arguments for the handler() class
59 kwargs = {}
60 kwargs["objtype"] = u'saved_action'
61 kwargs["name"] = u'Distribute Tanium Standard Utilities'
62
63 print "...CALLING: handler.get with args: {}".format(kwargs)
64 response = handler.get(**kwargs)
65
66 print "...OUTPUT: Type of response: ", type(response)
67
68 print "...OUTPUT: print of response:"
69 print response
70
71 # call the export_obj() method to convert response to JSON and store it in out
72 export_kwargs = {}
73 export_kwargs['obj'] = response

```

```
74 export_kwargs['export_format'] = 'json'
75
76 print "...CALLING: handler.export_obj() with args {}".format(export_kwargs)
77 out = handler.export_obj(**export_kwargs)
78
79 # trim the output if it is more than 15 lines long
80 if len(out.splitlines()) > 15:
81     out = out.splitlines()[0:15]
82     out.append('..trimmed for brevity..')
83     out = '\n'.join(out)
84
85 print "...OUTPUT: print the objects returned in JSON format:"
86 print out
```

## Get All Users

Get all users

- [STDOUT from Example Python Code](#)
- [STDERR from Example Python Code](#)
- [Example Python Code](#)

```
1  # import the basic python packages we need
2  import os
3  import sys
4  import tempfile
5  import pprint
6  import traceback
7
8  # disable python from generating a .pyc file
9  sys.dont_write_bytecode = True
10
11 # change me to the path of pytan if this script is not running from EXAMPLES/PYTAN_API
12 pytan_loc = "~/gh/pytan"
13 pytan_static_path = os.path.join(os.path.expanduser(pytan_loc), 'lib')
14
15 # Determine our script name, script dir
16 my_file = os.path.abspath(sys.argv[0])
17 my_dir = os.path.dirname(my_file)
18
19 # try to automatically determine the pytan lib directory by assuming it is in '.././lib/'
20 parent_dir = os.path.dirname(my_dir)
21 pytan_root_dir = os.path.dirname(parent_dir)
22 lib_dir = os.path.join(pytan_root_dir, 'lib')
23
24 # add pytan_loc and lib_dir to the PYTHONPATH variable
25 path_adds = [lib_dir, pytan_static_path]
26 [sys.path.append(aa) for aa in path_adds if aa not in sys.path]
27
28 # import pytan
29 import pytan
30
31 # create a dictionary of arguments for the pytan handler
32 handler_args = {}
33
34 # establish our connection info for the Tanium Server
```



```

35 handler_args['username'] = "Administrator"
36 handler_args['password'] = "Tanium2015!"
37 handler_args['host'] = "10.0.1.240"
38 handler_args['port'] = "443" # optional
39
40 # optional, level 0 is no output except warnings/errors
41 # level 1 through 12 are more and more verbose
42 handler_args['loglevel'] = 1
43
44 # optional, use a debug format for the logging output (uses two lines per log entry)
45 handler_args['debugformat'] = False
46
47 # optional, this saves all response objects to handler.session.ALL_REQUESTS_RESPONSES
48 # very useful for capturing the full exchange of XML requests and responses
49 handler_args['record_all_requests'] = True
50
51 # instantiate a handler using all of the arguments in the handler_args dictionary
52 print "...CALLING: pytan.handler() with args: {}".format(handler_args)
53 handler = pytan.Handler(**handler_args)
54
55 # print out the handler string
56 print "...OUTPUT: handler string: {}".format(handler)
57
58 # setup the arguments for the handler() class
59 kwargs = {}
60 kwargs["objtype"] = u'user'
61
62 print "...CALLING: handler.get_all with args: {}".format(kwargs)
63 response = handler.get_all(**kwargs)
64
65 print "...OUTPUT: Type of response: ", type(response)
66
67 print "...OUTPUT: print of response:"
68 print response
69
70 # call the export_obj() method to convert response to JSON and store it in out
71 export_kwargs = {}
72 export_kwargs['obj'] = response
73 export_kwargs['export_format'] = 'json'
74
75 print "...CALLING: handler.export_obj() with args {}".format(export_kwargs)
76 out = handler.export_obj(**export_kwargs)
77
78 # trim the output if it is more than 15 lines long
79 if len(out.splitlines()) > 15:
80     out = out.splitlines()[0:15]
81     out.append('..trimmed for brevity..')
82     out = '\n'.join(out)
83
84 print "...OUTPUT: print the objects returned in JSON format:"
85 print out

```

## Get All Saved Actions

Get all saved actions

- STDOUT from Example Python Code

- STDERR from Example Python Code
- Example Python Code

```
1  # import the basic python packages we need
2  import os
3  import sys
4  import tempfile
5  import pprint
6  import traceback
7
8  # disable python from generating a .pyc file
9  sys.dont_write_bytecode = True
10
11  # change me to the path of pytan if this script is not running from EXAMPLES/PYTAN_API
12  pytan_loc = "~/gh/pytan"
13  pytan_static_path = os.path.join(os.path.expanduser(pytan_loc), 'lib')
14
15  # Determine our script name, script dir
16  my_file = os.path.abspath(sys.argv[0])
17  my_dir = os.path.dirname(my_file)
18
19  # try to automatically determine the pytan lib directory by assuming it is in '../..lib/'
20  parent_dir = os.path.dirname(my_dir)
21  pytan_root_dir = os.path.dirname(parent_dir)
22  lib_dir = os.path.join(pytan_root_dir, 'lib')
23
24  # add pytan_loc and lib_dir to the PYTHONPATH variable
25  path_adds = [lib_dir, pytan_static_path]
26  [sys.path.append(aa) for aa in path_adds if aa not in sys.path]
27
28  # import pytan
29  import pytan
30
31  # create a dictionary of arguments for the pytan handler
32  handler_args = {}
33
34  # establish our connection info for the Tanium Server
35  handler_args['username'] = "Administrator"
36  handler_args['password'] = "Tanium2015!"
37  handler_args['host'] = "10.0.1.240"
38  handler_args['port'] = "443" # optional
39
40  # optional, level 0 is no output except warnings/errors
41  # level 1 through 12 are more and more verbose
42  handler_args['loglevel'] = 1
43
44  # optional, use a debug format for the logging output (uses two lines per log entry)
45  handler_args['debugformat'] = False
46
47  # optional, this saves all response objects to handler.session.ALL_REQUESTS_RESPONSES
48  # very useful for capturing the full exchange of XML requests and responses
49  handler_args['record_all_requests'] = True
50
51  # instantiate a handler using all of the arguments in the handler_args dictionary
52  print "...CALLING: pytan.handler() with args: {}".format(handler_args)
53  handler = pytan.Handler(**handler_args)
54
55  # print out the handler string
```

```

56 print "...OUTPUT: handler string: {}".format(handler)
57
58 # setup the arguments for the handler() class
59 kwargs = {}
60 kwargs["objtype"] = u'saved_action'
61
62 print "...CALLING: handler.get_all with args: {}".format(kwargs)
63 response = handler.get_all(**kwargs)
64
65 print "...OUTPUT: Type of response: ", type(response)
66
67 print "...OUTPUT: print of response:"
68 print response
69
70 # call the export_obj() method to convert response to JSON and store it in out
71 export_kwargs = {}
72 export_kwargs['obj'] = response
73 export_kwargs['export_format'] = 'json'
74
75 print "...CALLING: handler.export_obj() with args {}".format(export_kwargs)
76 out = handler.export_obj(**export_kwargs)
77
78 # trim the output if it is more than 15 lines long
79 if len(out.splitlines()) > 15:
80     out = out.splitlines()[0:15]
81     out.append('..trimmed for brevity..')
82     out = '\n'.join(out)
83
84 print "...OUTPUT: print the objects returned in JSON format:"
85 print out

```

## Get All Settings

Get all system settings

- [STDOUT from Example Python Code](#)
- [STDERR from Example Python Code](#)
- [Example Python Code](#)

```

1 # import the basic python packages we need
2 import os
3 import sys
4 import tempfile
5 import pprint
6 import traceback
7
8 # disable python from generating a .pyc file
9 sys.dont_write_bytecode = True
10
11 # change me to the path of pytan if this script is not running from EXAMPLES/PYTAN_API
12 pytan_loc = "~/gh/pytan"
13 pytan_static_path = os.path.join(os.path.expanduser(pytan_loc), 'lib')
14
15 # Determine our script name, script dir
16 my_file = os.path.abspath(sys.argv[0])
17 my_dir = os.path.dirname(my_file)

```

```
18
19 # try to automatically determine the pytan lib directory by assuming it is in '../..lib/'
20 parent_dir = os.path.dirname(my_dir)
21 pytan_root_dir = os.path.dirname(parent_dir)
22 lib_dir = os.path.join(pytan_root_dir, 'lib')
23
24 # add pytan_loc and lib_dir to the PYTHONPATH variable
25 path_adds = [lib_dir, pytan_static_path]
26 [sys.path.append(aa) for aa in path_adds if aa not in sys.path]
27
28 # import pytan
29 import pytan
30
31 # create a dictionary of arguments for the pytan handler
32 handler_args = {}
33
34 # establish our connection info for the Tanium Server
35 handler_args['username'] = "Administrator"
36 handler_args['password'] = "Tanium2015!"
37 handler_args['host'] = "10.0.1.240"
38 handler_args['port'] = "443" # optional
39
40 # optional, level 0 is no output except warnings/errors
41 # level 1 through 12 are more and more verbose
42 handler_args['loglevel'] = 1
43
44 # optional, use a debug format for the logging output (uses two lines per log entry)
45 handler_args['debugformat'] = False
46
47 # optional, this saves all response objects to handler.session.ALL_REQUESTS_RESPONSES
48 # very useful for capturing the full exchange of XML requests and responses
49 handler_args['record_all_requests'] = True
50
51 # instantiate a handler using all of the arguments in the handler_args dictionary
52 print "...CALLING: pytan.handler() with args: {}".format(handler_args)
53 handler = pytan.Handler(**handler_args)
54
55 # print out the handler string
56 print "...OUTPUT: handler string: {}".format(handler)
57
58 # setup the arguments for the handler() class
59 kwargs = {}
60 kwargs["objtype"] = u'setting'
61
62 print "...CALLING: handler.get_all with args: {}".format(kwargs)
63 response = handler.get_all(**kwargs)
64
65 print "...OUTPUT: Type of response: ", type(response)
66
67 print "...OUTPUT: print of response:"
68 print response
69
70 # call the export_obj() method to convert response to JSON and store it in out
71 export_kwargs = {}
72 export_kwargs['obj'] = response
73 export_kwargs['export_format'] = 'json'
74
75 print "...CALLING: handler.export_obj() with args {}".format(export_kwargs)
```

```

76 out = handler.export_obj(**export_kwargs)
77
78 # trim the output if it is more than 15 lines long
79 if len(out.splitlines()) > 15:
80     out = out.splitlines()[0:15]
81     out.append('..trimmed for brevity..')
82     out = '\n'.join(out)
83
84 print "...OUTPUT: print the objects returned in JSON format:"
85 print out

```

## Get All Saved Questions

Get all saved questions

- STDOUT from Example Python Code
- STDERR from Example Python Code
- Example Python Code

```

1  # import the basic python packages we need
2  import os
3  import sys
4  import tempfile
5  import pprint
6  import traceback
7
8  # disable python from generating a .pyc file
9  sys.dont_write_bytecode = True
10
11 # change me to the path of pytan if this script is not running from EXAMPLES/PYTAN_API
12 pytan_loc = "~/gh/pytan"
13 pytan_static_path = os.path.join(os.path.expanduser(pytan_loc), 'lib')
14
15 # Determine our script name, script dir
16 my_file = os.path.abspath(sys.argv[0])
17 my_dir = os.path.dirname(my_file)
18
19 # try to automatically determine the pytan lib directory by assuming it is in '.././lib/'
20 parent_dir = os.path.dirname(my_dir)
21 pytan_root_dir = os.path.dirname(parent_dir)
22 lib_dir = os.path.join(pytan_root_dir, 'lib')
23
24 # add pytan_loc and lib_dir to the PYTHONPATH variable
25 path_adds = [lib_dir, pytan_static_path]
26 [sys.path.append(aa) for aa in path_adds if aa not in sys.path]
27
28 # import pytan
29 import pytan
30
31 # create a dictionary of arguments for the pytan handler
32 handler_args = {}
33
34 # establish our connection info for the Tanium Server
35 handler_args['username'] = "Administrator"
36 handler_args['password'] = "Tanium2015!"
37 handler_args['host'] = "10.0.1.240"

```

```
38 handler_args['port'] = "443" # optional
39
40 # optional, level 0 is no output except warnings/errors
41 # level 1 through 12 are more and more verbose
42 handler_args['loglevel'] = 1
43
44 # optional, use a debug format for the logging output (uses two lines per log entry)
45 handler_args['debugformat'] = False
46
47 # optional, this saves all response objects to handler.session.ALL_REQUESTS_RESPONSES
48 # very useful for capturing the full exchange of XML requests and responses
49 handler_args['record_all_requests'] = True
50
51 # instantiate a handler using all of the arguments in the handler_args dictionary
52 print "...CALLING: pytan.handler() with args: {}".format(handler_args)
53 handler = pytan.Handler(**handler_args)
54
55 # print out the handler string
56 print "...OUTPUT: handler string: {}".format(handler)
57
58 # setup the arguments for the handler() class
59 kwargs = {}
60 kwargs["objtype"] = u'saved_question'
61
62 print "...CALLING: handler.get_all with args: {}".format(kwargs)
63 response = handler.get_all(**kwargs)
64
65 print "...OUTPUT: Type of response: ", type(response)
66
67 print "...OUTPUT: print of response:"
68 print response
69
70 # call the export_obj() method to convert response to JSON and store it in out
71 export_kwargs = {}
72 export_kwargs['obj'] = response
73 export_kwargs['export_format'] = 'json'
74
75 print "...CALLING: handler.export_obj() with args {}".format(export_kwargs)
76 out = handler.export_obj(**export_kwargs)
77
78 # trim the output if it is more than 15 lines long
79 if len(out.splitlines()) > 15:
80     out = out.splitlines()[0:15]
81     out.append('..trimmed for brevity..')
82     out = '\n'.join(out)
83
84 print "...OUTPUT: print the objects returned in JSON format:"
85 print out
```

## Get All Userroles

Get all user roles

- [STDOUT from Example Python Code](#)
- [STDERR from Example Python Code](#)
- [Example Python Code](#)

```

1  # import the basic python packages we need
2  import os
3  import sys
4  import tempfile
5  import pprint
6  import traceback
7
8  # disable python from generating a .pyc file
9  sys.dont_write_bytecode = True
10
11  # change me to the path of pytan if this script is not running from EXAMPLES/PYTAN_API
12  pytan_loc = "~/gh/pytan"
13  pytan_static_path = os.path.join(os.path.expanduser(pytan_loc), 'lib')
14
15  # Determine our script name, script dir
16  my_file = os.path.abspath(sys.argv[0])
17  my_dir = os.path.dirname(my_file)
18
19  # try to automatically determine the pytan lib directory by assuming it is in '../..lib/'
20  parent_dir = os.path.dirname(my_dir)
21  pytan_root_dir = os.path.dirname(parent_dir)
22  lib_dir = os.path.join(pytan_root_dir, 'lib')
23
24  # add pytan_loc and lib_dir to the PYTHONPATH variable
25  path_adds = [lib_dir, pytan_static_path]
26  [sys.path.append(aa) for aa in path_adds if aa not in sys.path]
27
28  # import pytan
29  import pytan
30
31  # create a dictionary of arguments for the pytan handler
32  handler_args = {}
33
34  # establish our connection info for the Tanium Server
35  handler_args['username'] = "Administrator"
36  handler_args['password'] = "Tanium2015!"
37  handler_args['host'] = "10.0.1.240"
38  handler_args['port'] = "443" # optional
39
40  # optional, level 0 is no output except warnings/errors
41  # level 1 through 12 are more and more verbose
42  handler_args['loglevel'] = 1
43
44  # optional, use a debug format for the logging output (uses two lines per log entry)
45  handler_args['debugformat'] = False
46
47  # optional, this saves all response objects to handler.session.ALL_REQUESTS_RESPONSES
48  # very useful for capturing the full exchange of XML requests and responses
49  handler_args['record_all_requests'] = True
50
51  # instantiate a handler using all of the arguments in the handler_args dictionary
52  print "...CALLING: pytan.handler() with args: {}".format(handler_args)
53  handler = pytan.Handler(**handler_args)
54
55  # print out the handler string
56  print "...OUTPUT: handler string: {}".format(handler)
57
58  # setup the arguments for the handler() class

```

```
59 kwargs = {}
60 kwargs["objtype"] = u'userrole'
61
62 print "...CALLING: handler.get_all with args: {}".format(kwargs)
63 response = handler.get_all(**kwargs)
64
65 print "...OUTPUT: Type of response: ", type(response)
66
67 print "...OUTPUT: print of response:"
68 print response
69
70 # call the export_obj() method to convert response to JSON and store it in out
71 export_kwargs = {}
72 export_kwargs['obj'] = response
73 export_kwargs['export_format'] = 'json'
74
75 print "...CALLING: handler.export_obj() with args {}".format(export_kwargs)
76 out = handler.export_obj(**export_kwargs)
77
78 # trim the output if it is more than 15 lines long
79 if len(out.splitlines()) > 15:
80     out = out.splitlines()[0:15]
81     out.append('..trimmed for brevity..')
82     out = '\n'.join(out)
83
84 print "...OUTPUT: print the objects returned in JSON format:"
85 print out
```

## Get All Questions

Get all questions

- [STDOUT from Example Python Code](#)
- [STDERR from Example Python Code](#)
- [Example Python Code](#)

```
1 # import the basic python packages we need
2 import os
3 import sys
4 import tempfile
5 import pprint
6 import traceback
7
8 # disable python from generating a .pyc file
9 sys.dont_write_bytecode = True
10
11 # change me to the path of pytan if this script is not running from EXAMPLES/PYTAN_API
12 pytan_loc = "~/gh/pytan"
13 pytan_static_path = os.path.join(os.path.expanduser(pytan_loc), 'lib')
14
15 # Determine our script name, script dir
16 my_file = os.path.abspath(sys.argv[0])
17 my_dir = os.path.dirname(my_file)
18
19 # try to automatically determine the pytan lib directory by assuming it is in '../..lib/'
20 parent_dir = os.path.dirname(my_dir)
```



```

21 pytan_root_dir = os.path.dirname(parent_dir)
22 lib_dir = os.path.join(pytan_root_dir, 'lib')
23
24 # add pytan_loc and lib_dir to the PYTHONPATH variable
25 path_adds = [lib_dir, pytan_static_path]
26 [sys.path.append(aa) for aa in path_adds if aa not in sys.path]
27
28 # import pytan
29 import pytan
30
31 # create a dictionary of arguments for the pytan handler
32 handler_args = {}
33
34 # establish our connection info for the Tanium Server
35 handler_args['username'] = "Administrator"
36 handler_args['password'] = "Tanium2015!"
37 handler_args['host'] = "10.0.1.240"
38 handler_args['port'] = "443" # optional
39
40 # optional, level 0 is no output except warnings/errors
41 # level 1 through 12 are more and more verbose
42 handler_args['loglevel'] = 1
43
44 # optional, use a debug format for the logging output (uses two lines per log entry)
45 handler_args['debugformat'] = False
46
47 # optional, this saves all response objects to handler.session.ALL_REQUESTS_RESPONSES
48 # very useful for capturing the full exchange of XML requests and responses
49 handler_args['record_all_requests'] = True
50
51 # instantiate a handler using all of the arguments in the handler_args dictionary
52 print "...CALLING: pytan.handler() with args: {}".format(handler_args)
53 handler = pytan.Handler(**handler_args)
54
55 # print out the handler string
56 print "...OUTPUT: handler string: {}".format(handler)
57
58 # setup the arguments for the handler() class
59 kwargs = {}
60 kwargs["objtype"] = u'question'
61
62 print "...CALLING: handler.get_all with args: {}".format(kwargs)
63 response = handler.get_all(**kwargs)
64
65 print "...OUTPUT: Type of response: ", type(response)
66
67 print "...OUTPUT: print of response:"
68 print response
69
70 # call the export_obj() method to convert response to JSON and store it in out
71 export_kwargs = {}
72 export_kwargs['obj'] = response
73 export_kwargs['export_format'] = 'json'
74
75 print "...CALLING: handler.export_obj() with args {}".format(export_kwargs)
76 out = handler.export_obj(**export_kwargs)
77
78 # trim the output if it is more than 15 lines long

```

```
79 if len(out.splitlines()) > 15:
80     out = out.splitlines()[0:15]
81     out.append('..trimmed for brevity..')
82     out = '\n'.join(out)
83
84 print "...OUTPUT: print the objects returned in JSON format:"
85 print out
```

## Get All Groups

Get all groups

- [STDOUT from Example Python Code](#)
- [STDERR from Example Python Code](#)
- [Example Python Code](#)

```
1  # import the basic python packages we need
2  import os
3  import sys
4  import tempfile
5  import pprint
6  import traceback
7
8  # disable python from generating a .pyc file
9  sys.dont_write_bytecode = True
10
11 # change me to the path of pytan if this script is not running from EXAMPLES/PYTAN_API
12 pytan_loc = "~/gh/pytan"
13 pytan_static_path = os.path.join(os.path.expanduser(pytan_loc), 'lib')
14
15 # Determine our script name, script dir
16 my_file = os.path.abspath(sys.argv[0])
17 my_dir = os.path.dirname(my_file)
18
19 # try to automatically determine the pytan lib directory by assuming it is in '../..lib/'
20 parent_dir = os.path.dirname(my_dir)
21 pytan_root_dir = os.path.dirname(parent_dir)
22 lib_dir = os.path.join(pytan_root_dir, 'lib')
23
24 # add pytan_loc and lib_dir to the PYTHONPATH variable
25 path_adds = [lib_dir, pytan_static_path]
26 [sys.path.append(aa) for aa in path_adds if aa not in sys.path]
27
28 # import pytan
29 import pytan
30
31 # create a dictionary of arguments for the pytan handler
32 handler_args = {}
33
34 # establish our connection info for the Tanium Server
35 handler_args['username'] = "Administrator"
36 handler_args['password'] = "Tanium2015!"
37 handler_args['host'] = "10.0.1.240"
38 handler_args['port'] = "443" # optional
39
40 # optional, level 0 is no output except warnings/errors
```

```

41 # level 1 through 12 are more and more verbose
42 handler_args['loglevel'] = 1
43
44 # optional, use a debug format for the logging output (uses two lines per log entry)
45 handler_args['debugformat'] = False
46
47 # optional, this saves all response objects to handler.session.ALL_REQUESTS_RESPONSES
48 # very useful for capturing the full exchange of XML requests and responses
49 handler_args['record_all_requests'] = True
50
51 # instantiate a handler using all of the arguments in the handler_args dictionary
52 print "...CALLING: pytan.handler() with args: {}".format(handler_args)
53 handler = pytan.Handler(**handler_args)
54
55 # print out the handler string
56 print "...OUTPUT: handler string: {}".format(handler)
57
58 # setup the arguments for the handler() class
59 kwargs = {}
60 kwargs["objtype"] = u'group'
61
62 print "...CALLING: handler.get_all with args: {}".format(kwargs)
63 response = handler.get_all(**kwargs)
64
65 print "...OUTPUT: Type of response: ", type(response)
66
67 print "...OUTPUT: print of response:"
68 print response
69
70 # call the export_obj() method to convert response to JSON and store it in out
71 export_kwargs = {}
72 export_kwargs['obj'] = response
73 export_kwargs['export_format'] = 'json'
74
75 print "...CALLING: handler.export_obj() with args {}".format(export_kwargs)
76 out = handler.export_obj(**export_kwargs)
77
78 # trim the output if it is more than 15 lines long
79 if len(out.splitlines()) > 15:
80     out = out.splitlines()[0:15]
81     out.append('..trimmed for brevity..')
82     out = '\n'.join(out)
83
84 print "...OUTPUT: print the objects returned in JSON format:"
85 print out

```

## Get All Sensors

Get all sensors

- STDOUT from Example Python Code
- STDERR from Example Python Code
- Example Python Code

```

1 # import the basic python packages we need
2 import os

```

```
3 import sys
4 import tempfile
5 import pprint
6 import traceback
7
8 # disable python from generating a .pyc file
9 sys.dont_write_bytecode = True
10
11 # change me to the path of pytan if this script is not running from EXAMPLES/PYTAN_API
12 pytan_loc = "~/gh/pytan"
13 pytan_static_path = os.path.join(os.path.expanduser(pytan_loc), 'lib')
14
15 # Determine our script name, script dir
16 my_file = os.path.abspath(sys.argv[0])
17 my_dir = os.path.dirname(my_file)
18
19 # try to automatically determine the pytan lib directory by assuming it is in '../..lib/'
20 parent_dir = os.path.dirname(my_dir)
21 pytan_root_dir = os.path.dirname(parent_dir)
22 lib_dir = os.path.join(pytan_root_dir, 'lib')
23
24 # add pytan_loc and lib_dir to the PYTHONPATH variable
25 path_adds = [lib_dir, pytan_static_path]
26 [sys.path.append(aa) for aa in path_adds if aa not in sys.path]
27
28 # import pytan
29 import pytan
30
31 # create a dictionary of arguments for the pytan handler
32 handler_args = {}
33
34 # establish our connection info for the Tanium Server
35 handler_args['username'] = "Administrator"
36 handler_args['password'] = "Tanium2015!"
37 handler_args['host'] = "10.0.1.240"
38 handler_args['port'] = "443" # optional
39
40 # optional, level 0 is no output except warnings/errors
41 # level 1 through 12 are more and more verbose
42 handler_args['loglevel'] = 1
43
44 # optional, use a debug format for the logging output (uses two lines per log entry)
45 handler_args['debugformat'] = False
46
47 # optional, this saves all response objects to handler.session.ALL_REQUESTS_RESPONSES
48 # very useful for capturing the full exchange of XML requests and responses
49 handler_args['record_all_requests'] = True
50
51 # instantiate a handler using all of the arguments in the handler_args dictionary
52 print "...CALLING: pytan.handler() with args: {}".format(handler_args)
53 handler = pytan.Handler(**handler_args)
54
55 # print out the handler string
56 print "...OUTPUT: handler string: {}".format(handler)
57
58 # setup the arguments for the handler() class
59 kwargs = {}
60 kwargs["objtype"] = u'sensor'
```

```

61
62 print "...CALLING: handler.get_all with args: {}".format(kwargs)
63 response = handler.get_all(**kwargs)
64
65 print "...OUTPUT: Type of response: ", type(response)
66
67 print "...OUTPUT: print of response:"
68 print response
69
70 # call the export_obj() method to convert response to JSON and store it in out
71 export_kwargs = {}
72 export_kwargs['obj'] = response
73 export_kwargs['export_format'] = 'json'
74
75 print "...CALLING: handler.export_obj() with args {}".format(export_kwargs)
76 out = handler.export_obj(**export_kwargs)
77
78 # trim the output if it is more than 15 lines long
79 if len(out.splitlines()) > 15:
80     out = out.splitlines()[0:15]
81     out.append('..trimmed for brevity..')
82     out = '\n'.join(out)
83
84 print "...OUTPUT: print the objects returned in JSON format:"
85 print out

```

## Get All Whitelisted Urls

Get all whitelisted urls

- [STDOUT from Example Python Code](#)
- [STDERR from Example Python Code](#)
- [Example Python Code](#)

```

1 # import the basic python packages we need
2 import os
3 import sys
4 import tempfile
5 import pprint
6 import traceback
7
8 # disable python from generating a .pyc file
9 sys.dont_write_bytecode = True
10
11 # change me to the path of pytan if this script is not running from EXAMPLES/PYTAN_API
12 pytan_loc = "~/gh/pytan"
13 pytan_static_path = os.path.join(os.path.expanduser(pytan_loc), 'lib')
14
15 # Determine our script name, script dir
16 my_file = os.path.abspath(sys.argv[0])
17 my_dir = os.path.dirname(my_file)
18
19 # try to automatically determine the pytan lib directory by assuming it is in '../..lib/'
20 parent_dir = os.path.dirname(my_dir)
21 pytan_root_dir = os.path.dirname(parent_dir)
22 lib_dir = os.path.join(pytan_root_dir, 'lib')

```

```
23
24 # add pytan_loc and lib_dir to the PYTHONPATH variable
25 path_adds = [lib_dir, pytan_static_path]
26 [sys.path.append(aa) for aa in path_adds if aa not in sys.path]
27
28 # import pytan
29 import pytan
30
31 # create a dictionary of arguments for the pytan handler
32 handler_args = {}
33
34 # establish our connection info for the Tanium Server
35 handler_args['username'] = "Administrator"
36 handler_args['password'] = "Tanium2015!"
37 handler_args['host'] = "10.0.1.240"
38 handler_args['port'] = "443" # optional
39
40 # optional, level 0 is no output except warnings/errors
41 # level 1 through 12 are more and more verbose
42 handler_args['loglevel'] = 1
43
44 # optional, use a debug format for the logging output (uses two lines per log entry)
45 handler_args['debugformat'] = False
46
47 # optional, this saves all response objects to handler.session.ALL_REQUESTS_RESPONSES
48 # very useful for capturing the full exchange of XML requests and responses
49 handler_args['record_all_requests'] = True
50
51 # instantiate a handler using all of the arguments in the handler_args dictionary
52 print "...CALLING: pytan.handler() with args: {}".format(handler_args)
53 handler = pytan.Handler(**handler_args)
54
55 # print out the handler string
56 print "...OUTPUT: handler string: {}".format(handler)
57
58 # setup the arguments for the handler() class
59 kwargs = {}
60 kwargs["objtype"] = u'whitelisted_url'
61
62 print "...CALLING: handler.get_all with args: {}".format(kwargs)
63 response = handler.get_all(**kwargs)
64
65 print "...OUTPUT: Type of response: ", type(response)
66
67 print "...OUTPUT: print of response:"
68 print response
69
70 # call the export_obj() method to convert response to JSON and store it in out
71 export_kwargs = {}
72 export_kwargs['obj'] = response
73 export_kwargs['export_format'] = 'json'
74
75 print "...CALLING: handler.export_obj() with args {}".format(export_kwargs)
76 out = handler.export_obj(**export_kwargs)
77
78 # trim the output if it is more than 15 lines long
79 if len(out.splitlines()) > 15:
80     out = out.splitlines()[0:15]
```

```

81     out.append('..trimmed for brevity..')
82     out = '\n'.join(out)
83
84 print "...OUTPUT: print the objects returned in JSON format:"
85 print out

```

## Get All Clients

Get all clients

- STDOUT from Example Python Code
- STDERR from Example Python Code
- Example Python Code

```

1  # import the basic python packages we need
2  import os
3  import sys
4  import tempfile
5  import pprint
6  import traceback
7
8  # disable python from generating a .pyc file
9  sys.dont_write_bytecode = True
10
11 # change me to the path of pytan if this script is not running from EXAMPLES/PYTAN_API
12 pytan_loc = "~/gh/pytan"
13 pytan_static_path = os.path.join(os.path.expanduser(pytan_loc), 'lib')
14
15 # Determine our script name, script dir
16 my_file = os.path.abspath(sys.argv[0])
17 my_dir = os.path.dirname(my_file)
18
19 # try to automatically determine the pytan lib directory by assuming it is in '../..lib/'
20 parent_dir = os.path.dirname(my_dir)
21 pytan_root_dir = os.path.dirname(parent_dir)
22 lib_dir = os.path.join(pytan_root_dir, 'lib')
23
24 # add pytan_loc and lib_dir to the PYTHONPATH variable
25 path_adds = [lib_dir, pytan_static_path]
26 [sys.path.append(aa) for aa in path_adds if aa not in sys.path]
27
28 # import pytan
29 import pytan
30
31 # create a dictionary of arguments for the pytan handler
32 handler_args = {}
33
34 # establish our connection info for the Tanium Server
35 handler_args['username'] = "Administrator"
36 handler_args['password'] = "Tanium2015!"
37 handler_args['host'] = "10.0.1.240"
38 handler_args['port'] = "443" # optional
39
40 # optional, level 0 is no output except warnings/errors
41 # level 1 through 12 are more and more verbose
42 handler_args['loglevel'] = 1

```

```
43 # optional, use a debug format for the logging output (uses two lines per log entry)
44 handler_args['debugformat'] = False
45
46 # optional, this saves all response objects to handler.session.ALL_REQUESTS_RESPONSES
47 # very useful for capturing the full exchange of XML requests and responses
48 handler_args['record_all_requests'] = True
49
50 # instantiate a handler using all of the arguments in the handler_args dictionary
51 print "...CALLING: pytan.handler() with args: {}".format(handler_args)
52 handler = pytan.Handler(**handler_args)
53
54 # print out the handler string
55 print "...OUTPUT: handler string: {}".format(handler)
56
57 # setup the arguments for the handler() class
58 kwargs = {}
59 kwargs["objtype"] = u'client'
60
61 print "...CALLING: handler.get_all with args: {}".format(kwargs)
62 response = handler.get_all(**kwargs)
63
64 print "...OUTPUT: Type of response: ", type(response)
65
66 print "...OUTPUT: print of response:"
67 print response
68
69 # call the export_obj() method to convert response to JSON and store it in out
70 export_kwargs = {}
71 export_kwargs['obj'] = response
72 export_kwargs['export_format'] = 'json'
73
74 print "...CALLING: handler.export_obj() with args {}".format(export_kwargs)
75 out = handler.export_obj(**export_kwargs)
76
77 # trim the output if it is more than 15 lines long
78 if len(out.splitlines()) > 15:
79     out = out.splitlines()[0:15]
80     out.append('..trimmed for brevity..')
81     out = '\n'.join(out)
82
83 print "...OUTPUT: print the objects returned in JSON format:"
84 print out
85
```

## Get All Packages

Get all packages

- STDOUT from Example Python Code
- STDERR from Example Python Code
- Example Python Code

```
1 # import the basic python packages we need
2 import os
3 import sys
4 import tempfile
```



```

5 import pprint
6 import traceback
7
8 # disable python from generating a .pyc file
9 sys.dont_write_bytecode = True
10
11 # change me to the path of pytan if this script is not running from EXAMPLES/PYTAN_API
12 pytan_loc = "~/gh/pytan"
13 pytan_static_path = os.path.join(os.path.expanduser(pytan_loc), 'lib')
14
15 # Determine our script name, script dir
16 my_file = os.path.abspath(sys.argv[0])
17 my_dir = os.path.dirname(my_file)
18
19 # try to automatically determine the pytan lib directory by assuming it is in '../..lib/'
20 parent_dir = os.path.dirname(my_dir)
21 pytan_root_dir = os.path.dirname(parent_dir)
22 lib_dir = os.path.join(pytan_root_dir, 'lib')
23
24 # add pytan_loc and lib_dir to the PYTHONPATH variable
25 path_adds = [lib_dir, pytan_static_path]
26 [sys.path.append(aa) for aa in path_adds if aa not in sys.path]
27
28 # import pytan
29 import pytan
30
31 # create a dictionary of arguments for the pytan handler
32 handler_args = {}
33
34 # establish our connection info for the Tanium Server
35 handler_args['username'] = "Administrator"
36 handler_args['password'] = "Tanium2015!"
37 handler_args['host'] = "10.0.1.240"
38 handler_args['port'] = "443" # optional
39
40 # optional, level 0 is no output except warnings/errors
41 # level 1 through 12 are more and more verbose
42 handler_args['loglevel'] = 1
43
44 # optional, use a debug format for the logging output (uses two lines per log entry)
45 handler_args['debugformat'] = False
46
47 # optional, this saves all response objects to handler.session.ALL_REQUESTS_RESPONSES
48 # very useful for capturing the full exchange of XML requests and responses
49 handler_args['record_all_requests'] = True
50
51 # instantiate a handler using all of the arguments in the handler_args dictionary
52 print "...CALLING: pytan.handler() with args: {}".format(handler_args)
53 handler = pytan.Handler(**handler_args)
54
55 # print out the handler string
56 print "...OUTPUT: handler string: {}".format(handler)
57
58 # setup the arguments for the handler() class
59 kwargs = {}
60 kwargs["objtype"] = u'package'
61
62 print "...CALLING: handler.get_all with args: {}".format(kwargs)

```

```
63 response = handler.get_all(**kwargs)
64
65 print "...OUTPUT: Type of response: ", type(response)
66
67 print "...OUTPUT: print of response:"
68 print response
69
70 # call the export_obj() method to convert response to JSON and store it in out
71 export_kwargs = {}
72 export_kwargs['obj'] = response
73 export_kwargs['export_format'] = 'json'
74
75 print "...CALLING: handler.export_obj() with args {}".format(export_kwargs)
76 out = handler.export_obj(**export_kwargs)
77
78 # trim the output if it is more than 15 lines long
79 if len(out.splitlines()) > 15:
80     out = out.splitlines()[0:15]
81     out.append('..trimmed for brevity..')
82     out = '\n'.join(out)
83
84 print "...OUTPUT: print the objects returned in JSON format:"
85 print out
```

## Get All Actions

Get all actions

- [STDOUT from Example Python Code](#)
- [STDERR from Example Python Code](#)
- [Example Python Code](#)

```
1 # import the basic python packages we need
2 import os
3 import sys
4 import tempfile
5 import pprint
6 import traceback
7
8 # disable python from generating a .pyc file
9 sys.dont_write_bytecode = True
10
11 # change me to the path of pytan if this script is not running from EXAMPLES/PYTAN_API
12 pytan_loc = "~/gh/pytan"
13 pytan_static_path = os.path.join(os.path.expanduser(pytan_loc), 'lib')
14
15 # Determine our script name, script dir
16 my_file = os.path.abspath(sys.argv[0])
17 my_dir = os.path.dirname(my_file)
18
19 # try to automatically determine the pytan lib directory by assuming it is in '../..lib/'
20 parent_dir = os.path.dirname(my_dir)
21 pytan_root_dir = os.path.dirname(parent_dir)
22 lib_dir = os.path.join(pytan_root_dir, 'lib')
23
24 # add pytan_loc and lib_dir to the PYTHONPATH variable
```

```

25 path_adds = [lib_dir, pytan_static_path]
26 [sys.path.append(aa) for aa in path_adds if aa not in sys.path]
27
28 # import pytan
29 import pytan
30
31 # create a dictionary of arguments for the pytan handler
32 handler_args = {}
33
34 # establish our connection info for the Tanium Server
35 handler_args['username'] = "Administrator"
36 handler_args['password'] = "Tanium2015!"
37 handler_args['host'] = "10.0.1.240"
38 handler_args['port'] = "443" # optional
39
40 # optional, level 0 is no output except warnings/errors
41 # level 1 through 12 are more and more verbose
42 handler_args['loglevel'] = 1
43
44 # optional, use a debug format for the logging output (uses two lines per log entry)
45 handler_args['debugformat'] = False
46
47 # optional, this saves all response objects to handler.session.ALL_REQUESTS_RESPONSES
48 # very useful for capturing the full exchange of XML requests and responses
49 handler_args['record_all_requests'] = True
50
51 # instantiate a handler using all of the arguments in the handler_args dictionary
52 print "...CALLING: pytan.handler() with args: {}".format(handler_args)
53 handler = pytan.Handler(**handler_args)
54
55 # print out the handler string
56 print "...OUTPUT: handler string: {}".format(handler)
57
58 # setup the arguments for the handler() class
59 kwargs = {}
60 kwargs["objtype"] = u'action'
61
62 print "...CALLING: handler.get_all with args: {}".format(kwargs)
63 response = handler.get_all(**kwargs)
64
65 print "...OUTPUT: Type of response: ", type(response)
66
67 print "...OUTPUT: print of response:"
68 print response
69
70 # call the export_obj() method to convert response to JSON and store it in out
71 export_kwargs = {}
72 export_kwargs['obj'] = response
73 export_kwargs['export_format'] = 'json'
74
75 print "...CALLING: handler.export_obj() with args {}".format(export_kwargs)
76 out = handler.export_obj(**export_kwargs)
77
78 # trim the output if it is more than 15 lines long
79 if len(out.splitlines()) > 15:
80     out = out.splitlines()[0:15]
81     out.append('..trimmed for brevity..')
82     out = '\n'.join(out)

```

```
83
84 print "...OUTPUT: print the objects returned in JSON format:"
85 print out
```

## PyTan API Valid Questions Examples

All of the PyTan API examples for Valid Questions

### 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.

- STDOUT from Example Python Code
- STDERR from Example Python Code
- Example Python Code

```
1  # import the basic python packages we need
2  import os
3  import sys
4  import tempfile
5  import pprint
6  import traceback
7
8  # disable python from generating a .pyc file
9  sys.dont_write_bytecode = True
10
11 # change me to the path of pytan if this script is not running from EXAMPLES/PYTAN_API
12 pytan_loc = "~/gh/pytan"
13 pytan_static_path = os.path.join(os.path.expanduser(pytan_loc), 'lib')
14
15 # Determine our script name, script dir
16 my_file = os.path.abspath(sys.argv[0])
17 my_dir = os.path.dirname(my_file)
18
19 # try to automatically determine the pytan lib directory by assuming it is in '../..lib/'
20 parent_dir = os.path.dirname(my_dir)
21 pytan_root_dir = os.path.dirname(parent_dir)
22 lib_dir = os.path.join(pytan_root_dir, 'lib')
23
24 # add pytan_loc and lib_dir to the PYTHONPATH variable
25 path_adds = [lib_dir, pytan_static_path]
26 [sys.path.append(aa) for aa in path_adds if aa not in sys.path]
27
28 # import pytan
29 import pytan
30
31 # create a dictionary of arguments for the pytan handler
32 handler_args = {}
33
34 # establish our connection info for the Tanium Server
35 handler_args['username'] = "Administrator"
36 handler_args['password'] = "Tanium2015!"
37 handler_args['host'] = "10.0.1.240"
```

```

38 handler_args['port'] = "443" # optional
39
40 # optional, level 0 is no output except warnings/errors
41 # level 1 through 12 are more and more verbose
42 handler_args['loglevel'] = 1
43
44 # optional, use a debug format for the logging output (uses two lines per log entry)
45 handler_args['debugformat'] = False
46
47 # optional, this saves all response objects to handler.session.ALL_REQUESTS_RESPONSES
48 # very useful for capturing the full exchange of XML requests and responses
49 handler_args['record_all_requests'] = True
50
51 # instantiate a handler using all of the arguments in the handler_args dictionary
52 print "...CALLING: pytan.handler() with args: {}".format(handler_args)
53 handler = pytan.Handler(**handler_args)
54
55 # print out the handler string
56 print "...OUTPUT: handler string: {}".format(handler)
57
58 # setup the arguments for the handler() class
59 kwargs = {}
60 kwargs["sensors"] = [u'Computer Name', u'Installed Applications']
61 kwargs["qtype"] = u'manual'
62
63 print "...CALLING: handler.ask with args: {}".format(kwargs)
64 response = handler.ask(**kwargs)
65
66 print "...OUTPUT: Type of response: ", type(response)
67
68 print "...OUTPUT: Pretty print of response:"
69 print pprint.pformat(response)
70
71 print "...OUTPUT: Equivalent Question if it were to be asked in the Tanium Console: "
72 print response['question_object'].query_text
73
74 # call the export_obj() method to convert response to CSV and store it in out
75 export_kwargs = {}
76 export_kwargs['obj'] = response['question_results']
77 export_kwargs['export_format'] = 'csv'
78
79 print "...CALLING: handler.export_obj() with args {}".format(export_kwargs)
80 out = handler.export_obj(**export_kwargs)
81
82 # trim the output if it is more than 15 lines long
83 if len(out.splitlines()) > 15:
84     out = out.splitlines()[0:15]
85     out.append('...trimmed for brevity..')
86     out = '\n'.join(out)
87
88 print "...OUTPUT: CSV Results of response: "
89 print out

```

### 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.

- STDOUT from Example Python Code
- STDERR from Example Python Code
- Example Python Code

```
1  # import the basic python packages we need
2  import os
3  import sys
4  import tempfile
5  import pprint
6  import traceback
7
8  # disable python from generating a .pyc file
9  sys.dont_write_bytecode = True
10
11  # change me to the path of pytan if this script is not running from EXAMPLES/PYTAN_API
12  pytan_loc = "~/gh/pytan"
13  pytan_static_path = os.path.join(os.path.expanduser(pytan_loc), 'lib')
14
15  # Determine our script name, script dir
16  my_file = os.path.abspath(sys.argv[0])
17  my_dir = os.path.dirname(my_file)
18
19  # try to automatically determine the pytan lib directory by assuming it is in '../..lib/'
20  parent_dir = os.path.dirname(my_dir)
21  pytan_root_dir = os.path.dirname(parent_dir)
22  lib_dir = os.path.join(pytan_root_dir, 'lib')
23
24  # add pytan_loc and lib_dir to the PYTHONPATH variable
25  path_adds = [lib_dir, pytan_static_path]
26  [sys.path.append(aa) for aa in path_adds if aa not in sys.path]
27
28  # import pytan
29  import pytan
30
31  # create a dictionary of arguments for the pytan handler
32  handler_args = {}
33
34  # establish our connection info for the Tanium Server
35  handler_args['username'] = "Administrator"
36  handler_args['password'] = "Tanium2015!"
37  handler_args['host'] = "10.0.1.240"
38  handler_args['port'] = "443" # optional
39
40  # optional, level 0 is no output except warnings/errors
41  # level 1 through 12 are more and more verbose
42  handler_args['loglevel'] = 1
43
44  # optional, use a debug format for the logging output (uses two lines per log entry)
45  handler_args['debugformat'] = False
46
47  # optional, this saves all response objects to handler.session.ALL_REQUESTS_RESPONSES
48  # very useful for capturing the full exchange of XML requests and responses
49  handler_args['record_all_requests'] = True
50
51  # instantiate a handler using all of the arguments in the handler_args dictionary
52  print "...CALLING: pytan.handler() with args: {}".format(handler_args)
```

```

53 handler = pytan.Handler(**handler_args)
54
55 # print out the handler string
56 print "...OUTPUT: handler string: {}".format(handler)
57
58 # setup the arguments for the handler() class
59 kwargs = {}
60 kwargs["sensors"] = u'Computer Name'
61 kwargs["qtype"] = u'manual'
62
63 print "...CALLING: handler.ask with args: {}".format(kwargs)
64 response = handler.ask(**kwargs)
65
66 print "...OUTPUT: Type of response: ", type(response)
67
68 print "...OUTPUT: Pretty print of response:"
69 pprint.pformat(response)
70
71 print "...OUTPUT: Equivalent Question if it were to be asked in the Tanium Console: "
72 print response['question_object'].query_text
73
74 # call the export_obj() method to convert response to CSV and store it in out
75 export_kwargs = {}
76 export_kwargs['obj'] = response['question_results']
77 export_kwargs['export_format'] = 'csv'
78
79 print "...CALLING: handler.export_obj() with args {}".format(export_kwargs)
80 out = handler.export_obj(**export_kwargs)
81
82 # trim the output if it is more than 15 lines long
83 if len(out.splitlines()) > 15:
84     out = out.splitlines()[0:15]
85     out.append('..trimmed for brevity..')
86     out = '\n'.join(out)
87
88 print "...OUTPUT: CSV Results of response: "
89 print out

```

### 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.

- STDOUT from Example Python Code
- STDERR from Example Python Code
- Example Python Code

```

1 # import the basic python packages we need
2 import os
3 import sys
4 import tempfile
5 import pprint
6 import traceback
7

```

```
8 # disable python from generating a .pyc file
9 sys.dont_write_bytecode = True
10
11 # change me to the path of pytan if this script is not running from EXAMPLES/PYTAN_API
12 pytan_loc = "~/gh/pytan"
13 pytan_static_path = os.path.join(os.path.expanduser(pytan_loc), 'lib')
14
15 # Determine our script name, script dir
16 my_file = os.path.abspath(sys.argv[0])
17 my_dir = os.path.dirname(my_file)
18
19 # try to automatically determine the pytan lib directory by assuming it is in '../..lib/'
20 parent_dir = os.path.dirname(my_dir)
21 pytan_root_dir = os.path.dirname(parent_dir)
22 lib_dir = os.path.join(pytan_root_dir, 'lib')
23
24 # add pytan_loc and lib_dir to the PYTHONPATH variable
25 path_adds = [lib_dir, pytan_static_path]
26 [sys.path.append(aa) for aa in path_adds if aa not in sys.path]
27
28 # import pytan
29 import pytan
30
31 # create a dictionary of arguments for the pytan handler
32 handler_args = {}
33
34 # establish our connection info for the Tanium Server
35 handler_args['username'] = "Administrator"
36 handler_args['password'] = "Tanium2015!"
37 handler_args['host'] = "10.0.1.240"
38 handler_args['port'] = "443" # optional
39
40 # optional, level 0 is no output except warnings/errors
41 # level 1 through 12 are more and more verbose
42 handler_args['loglevel'] = 1
43
44 # optional, use a debug format for the logging output (uses two lines per log entry)
45 handler_args['debugformat'] = False
46
47 # optional, this saves all response objects to handler.session.ALL_REQUESTS_RESPONSES
48 # very useful for capturing the full exchange of XML requests and responses
49 handler_args['record_all_requests'] = True
50
51 # instantiate a handler using all of the arguments in the handler_args dictionary
52 print "...CALLING: pytan.handler() with args: {}".format(handler_args)
53 handler = pytan.Handler(**handler_args)
54
55 # print out the handler string
56 print "...OUTPUT: handler string: {}".format(handler)
57
58 # setup the arguments for the handler() class
59 kwargs = {}
60 kwargs["sensors"] = [u'name:Computer Name', u'name:Installed Applications']
61 kwargs["qtype"] = u'manual'
62
63 print "...CALLING: handler.ask with args: {}".format(kwargs)
64 response = handler.ask(**kwargs)
65
```



```

66 print "...OUTPUT: Type of response: ", type(response)
67
68 print "...OUTPUT: Pretty print of response:"
69 print pprint.pformat(response)
70
71 print "...OUTPUT: Equivalent Question if it were to be asked in the Tanium Console: "
72 print response['question_object'].query_text
73
74 # call the export_obj() method to convert response to CSV and store it in out
75 export_kwargs = {}
76 export_kwargs['obj'] = response['question_results']
77 export_kwargs['export_format'] = 'csv'
78
79 print "...CALLING: handler.export_obj() with args {}".format(export_kwargs)
80 out = handler.export_obj(**export_kwargs)
81
82 # trim the output if it is more than 15 lines long
83 if len(out.splitlines()) > 15:
84     out = out.splitlines()[0:15]
85     out.append('..trimmed for brevity..')
86     out = '\n'.join(out)
87
88 print "...OUTPUT: CSV Results of response: "
89 print out

```

### 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.

- STDOUT from Example Python Code
- STDERR from Example Python Code
- Example Python Code

```

1 # import the basic python packages we need
2 import os
3 import sys
4 import tempfile
5 import pprint
6 import traceback
7
8 # disable python from generating a .pyc file
9 sys.dont_write_bytecode = True
10
11 # change me to the path of pytan if this script is not running from EXAMPLES/PYTAN_API
12 pytan_loc = "~/gh/pytan"
13 pytan_static_path = os.path.join(os.path.expanduser(pytan_loc), 'lib')
14
15 # Determine our script name, script dir
16 my_file = os.path.abspath(sys.argv[0])
17 my_dir = os.path.dirname(my_file)
18
19 # try to automatically determine the pytan lib directory by assuming it is in '../..lib/'

```

```
20 parent_dir = os.path.dirname(my_dir)
21 pytan_root_dir = os.path.dirname(parent_dir)
22 lib_dir = os.path.join(pytan_root_dir, 'lib')
23
24 # add pytan_loc and lib_dir to the PYTHONPATH variable
25 path_adds = [lib_dir, pytan_static_path]
26 [sys.path.append(aa) for aa in path_adds if aa not in sys.path]
27
28 # import pytan
29 import pytan
30
31 # create a dictionary of arguments for the pytan handler
32 handler_args = {}
33
34 # establish our connection info for the Tanium Server
35 handler_args['username'] = "Administrator"
36 handler_args['password'] = "Tanium2015!"
37 handler_args['host'] = "10.0.1.240"
38 handler_args['port'] = "443" # optional
39
40 # optional, level 0 is no output except warnings/errors
41 # level 1 through 12 are more and more verbose
42 handler_args['loglevel'] = 1
43
44 # optional, use a debug format for the logging output (uses two lines per log entry)
45 handler_args['debugformat'] = False
46
47 # optional, this saves all response objects to handler.session.ALL_REQUESTS_RESPONSES
48 # very useful for capturing the full exchange of XML requests and responses
49 handler_args['record_all_requests'] = True
50
51 # instantiate a handler using all of the arguments in the handler_args dictionary
52 print "...CALLING: pytan.handler() with args: {}".format(handler_args)
53 handler = pytan.Handler(**handler_args)
54
55 # print out the handler string
56 print "...OUTPUT: handler string: {}".format(handler)
57
58 # setup the arguments for the handler() class
59 kwargs = {}
60 kwargs["sensors"] = u'Folder Name Search with RegEx Match{dirname=Program Files,regex=Microsoft.*}'
61 kwargs["qtype"] = u'manual'
62
63 print "...CALLING: handler.ask with args: {}".format(kwargs)
64 response = handler.ask(**kwargs)
65
66 print "...OUTPUT: Type of response: ", type(response)
67
68 print "...OUTPUT: Pretty print of response:"
69 print pprint.pformat(response)
70
71 print "...OUTPUT: Equivalent Question if it were to be asked in the Tanium Console: "
72 print response['question_object'].query_text
73
74 # call the export_obj() method to convert response to CSV and store it in out
75 export_kwargs = {}
76 export_kwargs['obj'] = response['question_results']
77 export_kwargs['export_format'] = 'csv'
```

```

78
79 print "...CALLING: handler.export_obj() with args {}".format(export_kwargs)
80 out = handler.export_obj(**export_kwargs)
81
82 # trim the output if it is more than 15 lines long
83 if len(out.splitlines()) > 15:
84     out = out.splitlines()[0:15]
85     out.append('...trimmed for brevity..')
86     out = '\n'.join(out)
87
88 print "...OUTPUT: CSV Results of response: "
89 print out

```

### 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.

- STDOUT from Example Python Code
- STDERR from Example Python Code
- Example Python Code

```

1  # import the basic python packages we need
2  import os
3  import sys
4  import tempfile
5  import pprint
6  import traceback
7
8  # disable python from generating a .pyc file
9  sys.dont_write_bytecode = True
10
11 # change me to the path of pytan if this script is not running from EXAMPLES/PYTAN_API
12 pytan_loc = "~/gh/pytan"
13 pytan_static_path = os.path.join(os.path.expanduser(pytan_loc), 'lib')
14
15 # Determine our script name, script dir
16 my_file = os.path.abspath(sys.argv[0])
17 my_dir = os.path.dirname(my_file)
18
19 # try to automatically determine the pytan lib directory by assuming it is in '../..lib/'
20 parent_dir = os.path.dirname(my_dir)
21 pytan_root_dir = os.path.dirname(parent_dir)
22 lib_dir = os.path.join(pytan_root_dir, 'lib')
23
24 # add pytan_loc and lib_dir to the PYTHONPATH variable
25 path_adds = [lib_dir, pytan_static_path]
26 [sys.path.append(aa) for aa in path_adds if aa not in sys.path]
27
28 # import pytan
29 import pytan
30

```

```
31 # create a dictionary of arguments for the pytan handler
32 handler_args = {}
33
34 # establish our connection info for the Tanium Server
35 handler_args['username'] = "Administrator"
36 handler_args['password'] = "Tanium2015!"
37 handler_args['host'] = "10.0.1.240"
38 handler_args['port'] = "443" # optional
39
40 # optional, level 0 is no output except warnings/errors
41 # level 1 through 12 are more and more verbose
42 handler_args['loglevel'] = 1
43
44 # optional, use a debug format for the logging output (uses two lines per log entry)
45 handler_args['debugformat'] = False
46
47 # optional, this saves all response objects to handler.session.ALL_REQUESTS_RESPONSES
48 # very useful for capturing the full exchange of XML requests and responses
49 handler_args['record_all_requests'] = True
50
51 # instantiate a handler using all of the arguments in the handler_args dictionary
52 print "...CALLING: pytan.handler() with args: {}".format(handler_args)
53 handler = pytan.Handler(**handler_args)
54
55 # print out the handler string
56 print "...OUTPUT: handler string: {}".format(handler)
57
58 # setup the arguments for the handler() class
59 kwargs = {}
60 kwargs["sensors"] = [u'Folder Name Search with RegEx Match{dirname=Program Files,regex=Microsoft.*}',
61 u'Computer Name']
62 kwargs["qtype"] = u'manual'
63
64 print "...CALLING: handler.ask with args: {}".format(kwargs)
65 response = handler.ask(**kwargs)
66
67 print "...OUTPUT: Type of response: ", type(response)
68
69 print "...OUTPUT: Pretty print of response:"
70 pprint.pformat(response)
71
72 print "...OUTPUT: Equivalent Question if it were to be asked in the Tanium Console: "
73 print response['question_object'].query_text
74
75 # call the export_obj() method to convert response to CSV and store it in out
76 export_kwargs = {}
77 export_kwargs['obj'] = response['question_results']
78 export_kwargs['export_format'] = 'csv'
79
80 print "...CALLING: handler.export_obj() with args {}".format(export_kwargs)
81 out = handler.export_obj(**export_kwargs)
82
83 # trim the output if it is more than 15 lines long
84 if len(out.splitlines()) > 15:
85     out = out.splitlines()[0:15]
86     out.append('..trimmed for brevity..')
87     out = '\n'.join(out)
88
```

```

89 print "...OUTPUT: CSV Results of response: "
90 print out

```

### 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.

- STDOUT from Example Python Code
- STDERR from Example Python Code
- Example Python Code

```

1  # import the basic python packages we need
2  import os
3  import sys
4  import tempfile
5  import pprint
6  import traceback
7
8  # disable python from generating a .pyc file
9  sys.dont_write_bytecode = True
10
11 # change me to the path of pytan if this script is not running from EXAMPLES/PYTAN_API
12 pytan_loc = "~/gh/pytan"
13 pytan_static_path = os.path.join(os.path.expanduser(pytan_loc), 'lib')
14
15 # Determine our script name, script dir
16 my_file = os.path.abspath(sys.argv[0])
17 my_dir = os.path.dirname(my_file)
18
19 # try to automatically determine the pytan lib directory by assuming it is in '../..lib/'
20 parent_dir = os.path.dirname(my_dir)
21 pytan_root_dir = os.path.dirname(parent_dir)
22 lib_dir = os.path.join(pytan_root_dir, 'lib')
23
24 # add pytan_loc and lib_dir to the PYTHONPATH variable
25 path_adds = [lib_dir, pytan_static_path]
26 [sys.path.append(aa) for aa in path_adds if aa not in sys.path]
27
28 # import pytan
29 import pytan
30
31 # create a dictionary of arguments for the pytan handler
32 handler_args = {}
33
34 # establish our connection info for the Tanium Server
35 handler_args['username'] = "Administrator"
36 handler_args['password'] = "Tanium2015!"
37 handler_args['host'] = "10.0.1.240"
38 handler_args['port'] = "443" # optional
39
40 # optional, level 0 is no output except warnings/errors
41 # level 1 through 12 are more and more verbose
42 handler_args['loglevel'] = 1

```

```
43 # optional, use a debug format for the logging output (uses two lines per log entry)
44 handler_args['debugformat'] = False
45
46 # optional, this saves all response objects to handler.session.ALL_REQUESTS_RESPONSES
47 # very useful for capturing the full exchange of XML requests and responses
48 handler_args['record_all_requests'] = True
49
50
51 # instantiate a handler using all of the arguments in the handler_args dictionary
52 print "...CALLING: pytan.handler() with args: {}".format(handler_args)
53 handler = pytan.Handler(**handler_args)
54
55 # print out the handler string
56 print "...OUTPUT: handler string: {}".format(handler)
57
58 # setup the arguments for the handler() class
59 kwargs = {}
60 kwargs["sensors"] = u'Computer Name{fake=Dweedle}'
61 kwargs["qtype"] = u'manual'
62
63 print "...CALLING: handler.ask with args: {}".format(kwargs)
64 response = handler.ask(**kwargs)
65
66 print "...OUTPUT: Type of response: ", type(response)
67
68 print "...OUTPUT: Pretty print of response:"
69 pprint.pformat(response)
70
71 print "...OUTPUT: Equivalent Question if it were to be asked in the Tanium Console: "
72 print response['question_object'].query_text
73
74 # call the export_obj() method to convert response to CSV and store it in out
75 export_kwargs = {}
76 export_kwargs['obj'] = response['question_results']
77 export_kwargs['export_format'] = 'csv'
78
79 print "...CALLING: handler.export_obj() with args {}".format(export_kwargs)
80 out = handler.export_obj(**export_kwargs)
81
82 # trim the output if it is more than 15 lines long
83 if len(out.splitlines()) > 15:
84     out = out.splitlines()[0:15]
85     out.append('..trimmed for brevity..')
86     out = '\n'.join(out)
87
88 print "...OUTPUT: CSV Results of response: "
89 print out
```

### 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.

- STDOUT from Example Python Code

- STDERR from Example Python Code
- Example Python Code

```

1  # import the basic python packages we need
2  import os
3  import sys
4  import tempfile
5  import pprint
6  import traceback
7
8  # disable python from generating a .pyc file
9  sys.dont_write_bytecode = True
10
11 # change me to the path of pytan if this script is not running from EXAMPLES/PYTAN_API
12 pytan_loc = "~/gh/pytan"
13 pytan_static_path = os.path.join(os.path.expanduser(pytan_loc), 'lib')
14
15 # Determine our script name, script dir
16 my_file = os.path.abspath(sys.argv[0])
17 my_dir = os.path.dirname(my_file)
18
19 # try to automatically determine the pytan lib directory by assuming it is in '../..lib/'
20 parent_dir = os.path.dirname(my_dir)
21 pytan_root_dir = os.path.dirname(parent_dir)
22 lib_dir = os.path.join(pytan_root_dir, 'lib')
23
24 # add pytan_loc and lib_dir to the PYTHONPATH variable
25 path_adds = [lib_dir, pytan_static_path]
26 [sys.path.append(aa) for aa in path_adds if aa not in sys.path]
27
28 # import pytan
29 import pytan
30
31 # create a dictionary of arguments for the pytan handler
32 handler_args = {}
33
34 # establish our connection info for the Tanium Server
35 handler_args['username'] = "Administrator"
36 handler_args['password'] = "Tanium2015!"
37 handler_args['host'] = "10.0.1.240"
38 handler_args['port'] = "443" # optional
39
40 # optional, level 0 is no output except warnings/errors
41 # level 1 through 12 are more and more verbose
42 handler_args['loglevel'] = 1
43
44 # optional, use a debug format for the logging output (uses two lines per log entry)
45 handler_args['debugformat'] = False
46
47 # optional, this saves all response objects to handler.session.ALL_REQUESTS_RESPONSES
48 # very useful for capturing the full exchange of XML requests and responses
49 handler_args['record_all_requests'] = True
50
51 # instantiate a handler using all of the arguments in the handler_args dictionary
52 print "...CALLING: pytan.handler() with args: {}".format(handler_args)
53 handler = pytan.Handler(**handler_args)
54
55 # print out the handler string

```

```
56 print "...OUTPUT: handler string: {}".format(handler)
57
58 # setup the arguments for the handler() class
59 kwargs = {}
60 kwargs["sensors"] = u'Folder Name Search with RegEx Match'
61 kwargs["qtype"] = u'manual'
62
63 print "...CALLING: handler.ask with args: {}".format(kwargs)
64 response = handler.ask(**kwargs)
65
66 print "...OUTPUT: Type of response: ", type(response)
67
68 print "...OUTPUT: Pretty print of response:"
69 pprint.pformat(response)
70
71 print "...OUTPUT: Equivalent Question if it were to be asked in the Tanium Console: "
72 print response['question_object'].query_text
73
74 # call the export_obj() method to convert response to CSV and store it in out
75 export_kwargs = {}
76 export_kwargs['obj'] = response['question_results']
77 export_kwargs['export_format'] = 'csv'
78
79 print "...CALLING: handler.export_obj() with args {}".format(export_kwargs)
80 out = handler.export_obj(**export_kwargs)
81
82 # trim the output if it is more than 15 lines long
83 if len(out.splitlines()) > 15:
84     out = out.splitlines()[0:15]
85     out.append('..trimmed for brevity..')
86     out = '\n'.join(out)
87
88 print "...OUTPUT: CSV Results of response: "
89 print out
```

### 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.

- STDOUT from Example Python Code
- STDERR from Example Python Code
- Example Python Code

```
1 # import the basic python packages we need
2 import os
3 import sys
4 import tempfile
5 import pprint
6 import traceback
7
8 # disable python from generating a .pyc file
```



```

9 sys.dont_write_bytecode = True
10
11 # change me to the path of pytan if this script is not running from EXAMPLES/PYTAN_API
12 pytan_loc = "~/gh/pytan"
13 pytan_static_path = os.path.join(os.path.expanduser(pytan_loc), 'lib')
14
15 # Determine our script name, script dir
16 my_file = os.path.abspath(sys.argv[0])
17 my_dir = os.path.dirname(my_file)
18
19 # try to automatically determine the pytan lib directory by assuming it is in '../..lib/'
20 parent_dir = os.path.dirname(my_dir)
21 pytan_root_dir = os.path.dirname(parent_dir)
22 lib_dir = os.path.join(pytan_root_dir, 'lib')
23
24 # add pytan_loc and lib_dir to the PYTHONPATH variable
25 path_adds = [lib_dir, pytan_static_path]
26 [sys.path.append(aa) for aa in path_adds if aa not in sys.path]
27
28 # import pytan
29 import pytan
30
31 # create a dictionary of arguments for the pytan handler
32 handler_args = {}
33
34 # establish our connection info for the Tanium Server
35 handler_args['username'] = "Administrator"
36 handler_args['password'] = "Tanium2015!"
37 handler_args['host'] = "10.0.1.240"
38 handler_args['port'] = "443" # optional
39
40 # optional, level 0 is no output except warnings/errors
41 # level 1 through 12 are more and more verbose
42 handler_args['loglevel'] = 1
43
44 # optional, use a debug format for the logging output (uses two lines per log entry)
45 handler_args['debugformat'] = False
46
47 # optional, this saves all response objects to handler.session.ALL_REQUESTS_RESPONSES
48 # very useful for capturing the full exchange of XML requests and responses
49 handler_args['record_all_requests'] = True
50
51 # instantiate a handler using all of the arguments in the handler_args dictionary
52 print "...CALLING: pytan.handler() with args: {}".format(handler_args)
53 handler = pytan.Handler(**handler_args)
54
55 # print out the handler string
56 print "...OUTPUT: handler string: {}".format(handler)
57
58 # setup the arguments for the handler() class
59 kwargs = {}
60 kwargs["sensors"] = u'Folder Name Search with RegEx Match{dirname=Program Files,regex=Microsoft.*}, t
61 kwargs["qtype"] = u'manual'
62
63 print "...CALLING: handler.ask with args: {}".format(kwargs)
64 response = handler.ask(**kwargs)
65
66 print "...OUTPUT: Type of response: ", type(response)

```

```
67
68 print "...OUTPUT: Pretty print of response:"
69 print pprint.pformat(response)
70
71 print "...OUTPUT: Equivalent Question if it were to be asked in the Tanium Console: "
72 print response['question_object'].query_text
73
74 # call the export_obj() method to convert response to CSV and store it in out
75 export_kwargs = {}
76 export_kwargs['obj'] = response['question_results']
77 export_kwargs['export_format'] = 'csv'
78
79 print "...CALLING: handler.export_obj() with args {}".format(export_kwargs)
80 out = handler.export_obj(**export_kwargs)
81
82 # trim the output if it is more than 15 lines long
83 if len(out.splitlines()) > 15:
84     out = out.splitlines()[0:15]
85     out.append('..trimmed for brevity..')
86     out = '\n'.join(out)
87
88 print "...OUTPUT: CSV Results of response: "
89 print out
```

### 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.

- STDOUT from Example Python Code
- STDERR from Example Python Code
- Example Python Code

```
1 # import the basic python packages we need
2 import os
3 import sys
4 import tempfile
5 import pprint
6 import traceback
7
8 # disable python from generating a .pyc file
9 sys.dont_write_bytecode = True
10
11 # change me to the path of pytan if this script is not running from EXAMPLES/PYTAN_API
12 pytan_loc = "~/gh/pytan"
13 pytan_static_path = os.path.join(os.path.expanduser(pytan_loc), 'lib')
14
15 # Determine our script name, script dir
16 my_file = os.path.abspath(sys.argv[0])
17 my_dir = os.path.dirname(my_file)
```

```

18
19 # try to automatically determine the pytan lib directory by assuming it is in '../..lib/'
20 parent_dir = os.path.dirname(my_dir)
21 pytan_root_dir = os.path.dirname(parent_dir)
22 lib_dir = os.path.join(pytan_root_dir, 'lib')
23
24 # add pytan_loc and lib_dir to the PYTHONPATH variable
25 path_adds = [lib_dir, pytan_static_path]
26 [sys.path.append(aa) for aa in path_adds if aa not in sys.path]
27
28 # import pytan
29 import pytan
30
31 # create a dictionary of arguments for the pytan handler
32 handler_args = {}
33
34 # establish our connection info for the Tanium Server
35 handler_args['username'] = "Administrator"
36 handler_args['password'] = "Tanium2015!"
37 handler_args['host'] = "10.0.1.240"
38 handler_args['port'] = "443" # optional
39
40 # optional, level 0 is no output except warnings/errors
41 # level 1 through 12 are more and more verbose
42 handler_args['loglevel'] = 1
43
44 # optional, use a debug format for the logging output (uses two lines per log entry)
45 handler_args['debugformat'] = False
46
47 # optional, this saves all response objects to handler.session.ALL_REQUESTS_RESPONSES
48 # very useful for capturing the full exchange of XML requests and responses
49 handler_args['record_all_requests'] = True
50
51 # instantiate a handler using all of the arguments in the handler_args dictionary
52 print "...CALLING: pytan.handler() with args: {}".format(handler_args)
53 handler = pytan.Handler(**handler_args)
54
55 # print out the handler string
56 print "...OUTPUT: handler string: {}".format(handler)
57
58 # setup the arguments for the handler() class
59 kwargs = {}
60 kwargs["sensors"] = u'Operating System, that contains:Windows, opt:max_data_age:3600, opt:value_type:'
61 kwargs["qtype"] = u'manual'
62
63 print "...CALLING: handler.ask with args: {}".format(kwargs)
64 response = handler.ask(**kwargs)
65
66 print "...OUTPUT: Type of response: ", type(response)
67
68 print "...OUTPUT: Pretty print of response:"
69 print pprint.pformat(response)
70
71 print "...OUTPUT: Equivalent Question if it were to be asked in the Tanium Console: "
72 print response['question_object'].query_text
73
74 # call the export_obj() method to convert response to CSV and store it in out
75 export_kwargs = {}

```

```
76 export_kwargs['obj'] = response['question_results']
77 export_kwargs['export_format'] = 'csv'
78
79 print "...CALLING: handler.export_obj() with args {}".format(export_kwargs)
80 out = handler.export_obj(**export_kwargs)
81
82 # trim the output if it is more than 15 lines long
83 if len(out.splitlines()) > 15:
84     out = out.splitlines()[0:15]
85     out.append('..trimmed for brevity..')
86     out = '\n'.join(out)
87
88 print "...OUTPUT: CSV Results of response: "
89 print out
```

### 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.

- STDOUT from Example Python Code
- STDERR from Example Python Code
- Example Python Code

```
1  # import the basic python packages we need
2  import os
3  import sys
4  import tempfile
5  import pprint
6  import traceback
7
8  # disable python from generating a .pyc file
9  sys.dont_write_bytecode = True
10
11 # change me to the path of pytan if this script is not running from EXAMPLES/PYTAN_API
12 pytan_loc = "~/gh/pytan"
13 pytan_static_path = os.path.join(os.path.expanduser(pytan_loc), 'lib')
14
15 # Determine our script name, script dir
16 my_file = os.path.abspath(sys.argv[0])
17 my_dir = os.path.dirname(my_file)
18
19 # try to automatically determine the pytan lib directory by assuming it is in '../..lib/'
20 parent_dir = os.path.dirname(my_dir)
21 pytan_root_dir = os.path.dirname(parent_dir)
22 lib_dir = os.path.join(pytan_root_dir, 'lib')
23
24 # add pytan_loc and lib_dir to the PYTHONPATH variable
25 path_adds = [lib_dir, pytan_static_path]
26 [sys.path.append(aa) for aa in path_adds if aa not in sys.path]
27
28 # import pytan
```

```

29 import pytan
30
31 # create a dictionary of arguments for the pytan handler
32 handler_args = {}
33
34 # establish our connection info for the Tanium Server
35 handler_args['username'] = "Administrator"
36 handler_args['password'] = "Tanium2015!"
37 handler_args['host'] = "10.0.1.240"
38 handler_args['port'] = "443" # optional
39
40 # optional, level 0 is no output except warnings/errors
41 # level 1 through 12 are more and more verbose
42 handler_args['loglevel'] = 1
43
44 # optional, use a debug format for the logging output (uses two lines per log entry)
45 handler_args['debugformat'] = False
46
47 # optional, this saves all response objects to handler.session.ALL_REQUESTS_RESPONSES
48 # very useful for capturing the full exchange of XML requests and responses
49 handler_args['record_all_requests'] = True
50
51 # instantiate a handler using all of the arguments in the handler_args dictionary
52 print "...CALLING: pytan.handler() with args: {}".format(handler_args)
53 handler = pytan.Handler(**handler_args)
54
55 # print out the handler string
56 print "...OUTPUT: handler string: {}".format(handler)
57
58 # setup the arguments for the handler() class
59 kwargs = {}
60 kwargs["sensors"] = u'Operating System, that contains:Windows'
61 kwargs["qtype"] = u'manual'
62
63 print "...CALLING: handler.ask with args: {}".format(kwargs)
64 response = handler.ask(**kwargs)
65
66 print "...OUTPUT: Type of response: ", type(response)
67
68 print "...OUTPUT: Pretty print of response:"
69 print pprint.pformat(response)
70
71 print "...OUTPUT: Equivalent Question if it were to be asked in the Tanium Console: "
72 print response['question_object'].query_text
73
74 # call the export_obj() method to convert response to CSV and store it in out
75 export_kwargs = {}
76 export_kwargs['obj'] = response['question_results']
77 export_kwargs['export_format'] = 'csv'
78
79 print "...CALLING: handler.export_obj() with args {}".format(export_kwargs)
80 out = handler.export_obj(**export_kwargs)
81
82 # trim the output if it is more than 15 lines long
83 if len(out.splitlines()) > 15:
84     out = out.splitlines()[0:15]
85     out.append('..trimmed for brevity..')
86     out = '\n'.join(out)

```

```
87
88 print "...OUTPUT: CSV Results of response: "
89 print out
```

### 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.

- STDOUT from Example Python Code
- STDERR from Example Python Code
- Example Python Code

```
1  # import the basic python packages we need
2  import os
3  import sys
4  import tempfile
5  import pprint
6  import traceback
7
8  # disable python from generating a .pyc file
9  sys.dont_write_bytecode = True
10
11  # change me to the path of pytan if this script is not running from EXAMPLES/PYTAN_API
12  pytan_loc = "~/gh/pytan"
13  pytan_static_path = os.path.join(os.path.expanduser(pytan_loc), 'lib')
14
15  # Determine our script name, script dir
16  my_file = os.path.abspath(sys.argv[0])
17  my_dir = os.path.dirname(my_file)
18
19  # try to automatically determine the pytan lib directory by assuming it is in '../..lib/'
20  parent_dir = os.path.dirname(my_dir)
21  pytan_root_dir = os.path.dirname(parent_dir)
22  lib_dir = os.path.join(pytan_root_dir, 'lib')
23
24  # add pytan_loc and lib_dir to the PYTHONPATH variable
25  path_adds = [lib_dir, pytan_static_path]
26  [sys.path.append(aa) for aa in path_adds if aa not in sys.path]
27
28  # import pytan
29  import pytan
30
31  # create a dictionary of arguments for the pytan handler
32  handler_args = {}
33
34  # establish our connection info for the Tanium Server
35  handler_args['username'] = "Administrator"
36  handler_args['password'] = "Tanium2015!"
37  handler_args['host'] = "10.0.1.240"
38  handler_args['port'] = "443" # optional
39
```

```

40 # optional, level 0 is no output except warnings/errors
41 # level 1 through 12 are more and more verbose
42 handler_args['loglevel'] = 1
43
44 # optional, use a debug format for the logging output (uses two lines per log entry)
45 handler_args['debugformat'] = False
46
47 # optional, this saves all response objects to handler.session.ALL_REQUESTS_RESPONSES
48 # very useful for capturing the full exchange of XML requests and responses
49 handler_args['record_all_requests'] = True
50
51 # instantiate a handler using all of the arguments in the handler_args dictionary
52 print "...CALLING: pytan.handler() with args: {}".format(handler_args)
53 handler = pytan.Handler(**handler_args)
54
55 # print out the handler string
56 print "...OUTPUT: handler string: {}".format(handler)
57
58 # setup the arguments for the handler() class
59 kwargs = {}
60 kwargs["sensors"] = u'Folder Name Search with RegEx Match{dirname=Program Files,regex=Microsoft.*}, t
61 kwargs["qtype"] = u'manual'
62
63 print "...CALLING: handler.ask with args: {}".format(kwargs)
64 response = handler.ask(**kwargs)
65
66 print "...OUTPUT: Type of response: ", type(response)
67
68 print "...OUTPUT: Pretty print of response:"
69 print pprint.pformat(response)
70
71 print "...OUTPUT: Equivalent Question if it were to be asked in the Tanium Console: "
72 print response['question_object'].query_text
73
74 # call the export_obj() method to convert response to CSV and store it in out
75 export_kwargs = {}
76 export_kwargs['obj'] = response['question_results']
77 export_kwargs['export_format'] = 'csv'
78
79 print "...CALLING: handler.export_obj() with args {}".format(export_kwargs)
80 out = handler.export_obj(**export_kwargs)
81
82 # trim the output if it is more than 15 lines long
83 if len(out.splitlines()) > 15:
84     out = out.splitlines()[0:15]
85     out.append('..trimmed for brevity..')
86     out = '\n'.join(out)
87
88 print "...OUTPUT: CSV Results of response: "
89 print out

```

### 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 parameters, question filters, or question options supplied.

- STDOUT from Example Python Code
- STDERR from Example Python Code
- Example Python Code

```
1  # import the basic python packages we need
2  import os
3  import sys
4  import tempfile
5  import pprint
6  import traceback
7
8  # disable python from generating a .pyc file
9  sys.dont_write_bytecode = True
10
11 # change me to the path of pytan if this script is not running from EXAMPLES/PYTAN_API
12 pytan_loc = "~/gh/pytan"
13 pytan_static_path = os.path.join(os.path.expanduser(pytan_loc), 'lib')
14
15 # Determine our script name, script dir
16 my_file = os.path.abspath(sys.argv[0])
17 my_dir = os.path.dirname(my_file)
18
19 # try to automatically determine the pytan lib directory by assuming it is in '../..lib/'
20 parent_dir = os.path.dirname(my_dir)
21 pytan_root_dir = os.path.dirname(parent_dir)
22 lib_dir = os.path.join(pytan_root_dir, 'lib')
23
24 # add pytan_loc and lib_dir to the PYTHONPATH variable
25 path_adds = [lib_dir, pytan_static_path]
26 [sys.path.append(aa) for aa in path_adds if aa not in sys.path]
27
28 # import pytan
29 import pytan
30
31 # create a dictionary of arguments for the pytan handler
32 handler_args = {}
33
34 # establish our connection info for the Tanium Server
35 handler_args['username'] = "Administrator"
36 handler_args['password'] = "Tanium2015!"
37 handler_args['host'] = "10.0.1.240"
38 handler_args['port'] = "443" # optional
39
40 # optional, level 0 is no output except warnings/errors
41 # level 1 through 12 are more and more verbose
42 handler_args['loglevel'] = 1
43
44 # optional, use a debug format for the logging output (uses two lines per log entry)
45 handler_args['debugformat'] = False
46
47 # optional, this saves all response objects to handler.session.ALL_REQUESTS_RESPONSES
48 # very useful for capturing the full exchange of XML requests and responses
49 handler_args['record_all_requests'] = True
```



```

50
51 # instantiate a handler using all of the arguments in the handler_args dictionary
52 print "...CALLING: pytan.handler() with args: {}".format(handler_args)
53 handler = pytan.Handler(**handler_args)
54
55 # print out the handler string
56 print "...OUTPUT: handler string: {}".format(handler)
57
58 # setup the arguments for the handler() class
59 kwargs = {}
60 kwargs["sensors"] = u'Operating System, that contains:Windows, opt:match_all_values, opt:ignore_case,'
61 kwargs["qtype"] = u'manual'
62
63 print "...CALLING: handler.ask with args: {}".format(kwargs)
64 response = handler.ask(**kwargs)
65
66 print "...OUTPUT: Type of response: ", type(response)
67
68 print "...OUTPUT: Pretty print of response:"
69 print pprint.pformat(response)
70
71 print "...OUTPUT: Equivalent Question if it were to be asked in the Tanium Console: "
72 print response['question_object'].query_text
73
74 # call the export_obj() method to convert response to CSV and store it in out
75 export_kwargs = {}
76 export_kwargs['obj'] = response['question_results']
77 export_kwargs['export_format'] = 'csv'
78
79 print "...CALLING: handler.export_obj() with args {}".format(export_kwargs)
80 out = handler.export_obj(**export_kwargs)
81
82 # trim the output if it is more than 15 lines long
83 if len(out.splitlines()) > 15:
84     out = out.splitlines()[0:15]
85     out.append('..trimmed for brevity..')
86     out = '\n'.join(out)
87
88 print "...OUTPUT: CSV Results of response: "
89 print out

```

### Ask Manual Question Complex Query<sup>1</sup>

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.

- STDOUT from Example Python Code
- STDERR from Example Python Code

- Example Python Code

```
1  # import the basic python packages we need
2  import os
3  import sys
4  import tempfile
5  import pprint
6  import traceback
7
8  # disable python from generating a .pyc file
9  sys.dont_write_bytecode = True
10
11  # change me to the path of pytan if this script is not running from EXAMPLES/PYTAN_API
12  pytan_loc = "~/gh/pytan"
13  pytan_static_path = os.path.join(os.path.expanduser(pytan_loc), 'lib')
14
15  # Determine our script name, script dir
16  my_file = os.path.abspath(sys.argv[0])
17  my_dir = os.path.dirname(my_file)
18
19  # try to automatically determine the pytan lib directory by assuming it is in '../..lib/'
20  parent_dir = os.path.dirname(my_dir)
21  pytan_root_dir = os.path.dirname(parent_dir)
22  lib_dir = os.path.join(pytan_root_dir, 'lib')
23
24  # add pytan_loc and lib_dir to the PYTHONPATH variable
25  path_adds = [lib_dir, pytan_static_path]
26  [sys.path.append(aa) for aa in path_adds if aa not in sys.path]
27
28  # import pytan
29  import pytan
30
31  # create a dictionary of arguments for the pytan handler
32  handler_args = {}
33
34  # establish our connection info for the Tanium Server
35  handler_args['username'] = "Administrator"
36  handler_args['password'] = "Tanium2015!"
37  handler_args['host'] = "10.0.1.240"
38  handler_args['port'] = "443" # optional
39
40  # optional, level 0 is no output except warnings/errors
41  # level 1 through 12 are more and more verbose
42  handler_args['loglevel'] = 1
43
44  # optional, use a debug format for the logging output (uses two lines per log entry)
45  handler_args['debugformat'] = False
46
47  # optional, this saves all response objects to handler.session.ALL_REQUESTS_RESPONSES
48  # very useful for capturing the full exchange of XML requests and responses
49  handler_args['record_all_requests'] = True
50
51  # instantiate a handler using all of the arguments in the handler_args dictionary
52  print "...CALLING: pytan.handler() with args: {}".format(handler_args)
53  handler = pytan.Handler(**handler_args)
54
55  # print out the handler string
56  print "...OUTPUT: handler string: {}".format(handler)
57
```

```

58 # setup the arguments for the handler() class
59 kwargs = {}
60 kwargs["question_filters"] = [u'Operating System, that contains:Windows',
61     u'Operating System, that does not contain:Windows']
62 kwargs["sensors"] = [u'Computer Name',
63     u'Folder Name Search with RegEx Match{dirname=Program Files,regex=Microsoft.*, invalidparam=test}',
64     kwargs["question_options"] = [u'ignore_case', u'or']
65     kwargs["qtype"] = u'manual'
66
67     print "...CALLING: handler.ask with args: {}".format(kwargs)
68     response = handler.ask(**kwargs)
69
70     print "...OUTPUT: Type of response: ", type(response)
71
72     print "...OUTPUT: Pretty print of response:"
73     print pprint.pformat(response)
74
75     print "...OUTPUT: Equivalent Question if it were to be asked in the Tanium Console: "
76     print response['question_object'].query_text
77
78     # call the export_obj() method to convert response to CSV and store it in out
79     export_kwargs = {}
80     export_kwargs['obj'] = response['question_results']
81     export_kwargs['export_format'] = 'csv'
82
83     print "...CALLING: handler.export_obj() with args {}".format(export_kwargs)
84     out = handler.export_obj(**export_kwargs)
85
86     # trim the output if it is more than 15 lines long
87     if len(out.splitlines()) > 15:
88         out = out.splitlines()[0:15]
89         out.append('..trimmed for brevity..')
90         out = '\n'.join(out)
91
92     print "...OUTPUT: CSV Results of response: "
93     print out

```

### 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

- STDOUT from Example Python Code
- STDERR from Example Python Code
- Example Python Code

```

1 # import the basic python packages we need
2 import os
3 import sys
4 import tempfile
5 import pprint
6 import traceback
7
8 # disable python from generating a .pyc file
9 sys.dont_write_bytecode = True

```

```
10
11 # change me to the path of pytan if this script is not running from EXAMPLES/PYTAN_API
12 pytan_loc = "~/gh/pytan"
13 pytan_static_path = os.path.join(os.path.expanduser(pytan_loc), 'lib')
14
15 # Determine our script name, script dir
16 my_file = os.path.abspath(sys.argv[0])
17 my_dir = os.path.dirname(my_file)
18
19 # try to automatically determine the pytan lib directory by assuming it is in '../..lib/'
20 parent_dir = os.path.dirname(my_dir)
21 pytan_root_dir = os.path.dirname(parent_dir)
22 lib_dir = os.path.join(pytan_root_dir, 'lib')
23
24 # add pytan_loc and lib_dir to the PYTHONPATH variable
25 path_adds = [lib_dir, pytan_static_path]
26 [sys.path.append(aa) for aa in path_adds if aa not in sys.path]
27
28 # import pytan
29 import pytan
30
31 # create a dictionary of arguments for the pytan handler
32 handler_args = {}
33
34 # establish our connection info for the Tanium Server
35 handler_args['username'] = "Administrator"
36 handler_args['password'] = "Tanium2015!"
37 handler_args['host'] = "10.0.1.240"
38 handler_args['port'] = "443" # optional
39
40 # optional, level 0 is no output except warnings/errors
41 # level 1 through 12 are more and more verbose
42 handler_args['loglevel'] = 1
43
44 # optional, use a debug format for the logging output (uses two lines per log entry)
45 handler_args['debugformat'] = False
46
47 # optional, this saves all response objects to handler.session.ALL_REQUESTS_RESPONSES
48 # very useful for capturing the full exchange of XML requests and responses
49 handler_args['record_all_requests'] = True
50
51 # instantiate a handler using all of the arguments in the handler_args dictionary
52 print "...CALLING: pytan.handler() with args: {}".format(handler_args)
53 handler = pytan.Handler(**handler_args)
54
55 # print out the handler string
56 print "...OUTPUT: handler string: {}".format(handler)
57
58 # setup the arguments for the handler() class
59 kwargs = {}
60 kwargs["question_filters"] = [u'Installed Applications, that regex match:.*Google (Search|Chrome).*']
61 kwargs["sensors"] = [u'Computer Name',
62     u'Last Logged In User',
63     u'Installed Applications, that regex match:.*Google (Search|Chrome).*']
64 kwargs["question_options"] = [u'ignore_case', u'or']
65 kwargs["qtype"] = u'manual'
66
67 print "...CALLING: handler.ask with args: {}".format(kwargs)
```

```

68 response = handler.ask(**kwargs)
69
70 print "...OUTPUT: Type of response: ", type(response)
71
72 print "...OUTPUT: Pretty print of response:"
73 print pprint.pformat(response)
74
75 print "...OUTPUT: Equivalent Question if it were to be asked in the Tanium Console: "
76 print response['question_object'].query_text
77
78 # call the export_obj() method to convert response to CSV and store it in out
79 export_kwargs = {}
80 export_kwargs['obj'] = response['question_results']
81 export_kwargs['export_format'] = 'csv'
82
83 print "...CALLING: handler.export_obj() with args {}".format(export_kwargs)
84 out = handler.export_obj(**export_kwargs)
85
86 # trim the output if it is more than 15 lines long
87 if len(out.splitlines()) > 15:
88     out = out.splitlines()[0:15]
89     out.append('..trimmed for brevity..')
90     out = '\n'.join(out)
91
92 print "...OUTPUT: CSV Results of response: "
93 print out

```

### 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.

- STDOUT from Example Python Code
- STDERR from Example Python Code
- Example Python Code

```

1 # import the basic python packages we need
2 import os
3 import sys
4 import tempfile
5 import pprint
6 import traceback
7
8 # disable python from generating a .pyc file
9 sys.dont_write_bytecode = True
10
11 # change me to the path of pytan if this script is not running from EXAMPLES/PYTAN_API

```

```
12 pytan_loc = "~/gh/pytan"
13 pytan_static_path = os.path.join(os.path.expanduser(pytan_loc), 'lib')
14
15 # Determine our script name, script dir
16 my_file = os.path.abspath(sys.argv[0])
17 my_dir = os.path.dirname(my_file)
18
19 # try to automatically determine the pytan lib directory by assuming it is in '../..lib/'
20 parent_dir = os.path.dirname(my_dir)
21 pytan_root_dir = os.path.dirname(parent_dir)
22 lib_dir = os.path.join(pytan_root_dir, 'lib')
23
24 # add pytan_loc and lib_dir to the PYTHONPATH variable
25 path_adds = [lib_dir, pytan_static_path]
26 [sys.path.append(aa) for aa in path_adds if aa not in sys.path]
27
28 # import pytan
29 import pytan
30
31 # create a dictionary of arguments for the pytan handler
32 handler_args = {}
33
34 # establish our connection info for the Tanium Server
35 handler_args['username'] = "Administrator"
36 handler_args['password'] = "Tanium2015!"
37 handler_args['host'] = "10.0.1.240"
38 handler_args['port'] = "443" # optional
39
40 # optional, level 0 is no output except warnings/errors
41 # level 1 through 12 are more and more verbose
42 handler_args['loglevel'] = 1
43
44 # optional, use a debug format for the logging output (uses two lines per log entry)
45 handler_args['debugformat'] = False
46
47 # optional, this saves all response objects to handler.session.ALL_REQUESTS_RESPONSES
48 # very useful for capturing the full exchange of XML requests and responses
49 handler_args['record_all_requests'] = True
50
51 # instantiate a handler using all of the arguments in the handler_args dictionary
52 print "...CALLING: pytan.handler() with args: {}".format(handler_args)
53 handler = pytan.Handler(**handler_args)
54
55 # print out the handler string
56 print "...OUTPUT: handler string: {}".format(handler)
57
58 # setup the arguments for the handler() class
59 kwargs = {}
60 kwargs["question_filter_defs"] = [{u'filter': {u'not_flag': 0,
61                                     u'operator': u'RegexMatch',
62                                     u'value': u'.*Windows.*'},
63                                   u'name': u'Operating System'}]
64 kwargs["sensor_defs"] = [u'Computer Name',
65                           {u'filter': {u'not_flag': 0,
66                                         u'operator': u'RegexMatch',
67                                         u'value': u'.*Shared.*'},
68                             u'name': u'Folder Name Search with RegEx Match',
69                             u'options': {u'ignore_case_flag': 0,
```

```

70         u'max_age_seconds': 3600,
71         u'value_type': u'string'},
72     u'params': {u'dirname': u'Program Files'}}]
73 kwargs["question_option_defs"] = {u'and_flag': 0, u'ignore_case_flag': 0, u'max_age_seconds': 3600}
74 kwargs["qtype"] = u'_manual'
75
76 print "...CALLING: handler.ask with args: {}".format(kwargs)
77 response = handler.ask(**kwargs)
78
79 print "...OUTPUT: Type of response: ", type(response)
80
81 print "...OUTPUT: Pretty print of response:"
82 print pprint.pformat(response)
83
84 print "...OUTPUT: Equivalent Question if it were to be asked in the Tanium Console: "
85 print response['question_object'].query_text
86
87 # call the export_obj() method to convert response to CSV and store it in out
88 export_kwargs = {}
89 export_kwargs['obj'] = response['question_results']
90 export_kwargs['export_format'] = 'csv'
91
92 print "...CALLING: handler.export_obj() with args {}".format(export_kwargs)
93 out = handler.export_obj(**export_kwargs)
94
95 # trim the output if it is more than 15 lines long
96 if len(out.splitlines()) > 15:
97     out = out.splitlines()[0:15]
98     out.append('..trimmed for brevity..')
99     out = '\n'.join(out)
100
101 print "...OUTPUT: CSV Results of response: "
102 print out

```

## PyTan API Valid Saved Questions Examples

All of the PyTan API examples for Valid Saved Questions

### Ask Saved Question Refresh Data

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

- STDOUT from Example Python Code
- STDERR from Example Python Code
- Example Python Code

```

1 # import the basic python packages we need
2 import os
3 import sys
4 import tempfile
5 import pprint
6 import traceback
7
8 # disable python from generating a .pyc file
9 sys.dont_write_bytecode = True

```

```
10
11 # change me to the path of pytan if this script is not running from EXAMPLES/PYTAN_API
12 pytan_loc = "~/gh/pytan"
13 pytan_static_path = os.path.join(os.path.expanduser(pytan_loc), 'lib')
14
15 # Determine our script name, script dir
16 my_file = os.path.abspath(sys.argv[0])
17 my_dir = os.path.dirname(my_file)
18
19 # try to automatically determine the pytan lib directory by assuming it is in '../..lib/'
20 parent_dir = os.path.dirname(my_dir)
21 pytan_root_dir = os.path.dirname(parent_dir)
22 lib_dir = os.path.join(pytan_root_dir, 'lib')
23
24 # add pytan_loc and lib_dir to the PYTHONPATH variable
25 path_adds = [lib_dir, pytan_static_path]
26 [sys.path.append(aa) for aa in path_adds if aa not in sys.path]
27
28 # import pytan
29 import pytan
30
31 # create a dictionary of arguments for the pytan handler
32 handler_args = {}
33
34 # establish our connection info for the Tanium Server
35 handler_args['username'] = "Administrator"
36 handler_args['password'] = "Tanium2015!"
37 handler_args['host'] = "10.0.1.240"
38 handler_args['port'] = "443" # optional
39
40 # optional, level 0 is no output except warnings/errors
41 # level 1 through 12 are more and more verbose
42 handler_args['loglevel'] = 1
43
44 # optional, use a debug format for the logging output (uses two lines per log entry)
45 handler_args['debugformat'] = False
46
47 # optional, this saves all response objects to handler.session.ALL_REQUESTS_RESPONSES
48 # very useful for capturing the full exchange of XML requests and responses
49 handler_args['record_all_requests'] = True
50
51 # instantiate a handler using all of the arguments in the handler_args dictionary
52 print "...CALLING: pytan.handler() with args: {}".format(handler_args)
53 handler = pytan.Handler(**handler_args)
54
55 # print out the handler string
56 print "...OUTPUT: handler string: {}".format(handler)
57
58 # setup the arguments for the handler() class
59 kwargs = {}
60 kwargs["refresh_data"] = True
61 kwargs["qtype"] = u'saved'
62 kwargs["name"] = u'Installed Applications'
63
64 print "...CALLING: handler.ask with args: {}".format(kwargs)
65 response = handler.ask(**kwargs)
66
67 print "...OUTPUT: Type of response: ", type(response)
```



```

68
69 print "...OUTPUT: Pretty print of response:"
70 pprint.pformat(response)
71
72 print "...OUTPUT: Equivalent Question if it were to be asked in the Tanium Console: "
73 print response['question_object'].query_text
74
75 # call the export_obj() method to convert response to CSV and store it in out
76 export_kwargs = {}
77 export_kwargs['obj'] = response['question_results']
78 export_kwargs['export_format'] = 'csv'
79
80 print "...CALLING: handler.export_obj() with args {}".format(export_kwargs)
81 out = handler.export_obj(**export_kwargs)
82
83 # trim the output if it is more than 15 lines long
84 if len(out.splitlines()) > 15:
85     out = out.splitlines()[0:15]
86     out.append('..trimmed for brevity..')
87     out = '\n'.join(out)
88
89 print "...OUTPUT: CSV Results of response: "
90 print out

```

### Ask Saved Question By Name

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

- STDOUT from Example Python Code
- STDERR from Example Python Code
- Example Python Code

```

1  # import the basic python packages we need
2  import os
3  import sys
4  import tempfile
5  import pprint
6  import traceback
7
8  # disable python from generating a .pyc file
9  sys.dont_write_bytecode = True
10
11 # change me to the path of pytan if this script is not running from EXAMPLES/PYTAN_API
12 pytan_loc = "~/gh/pytan"
13 pytan_static_path = os.path.join(os.path.expanduser(pytan_loc), 'lib')
14
15 # Determine our script name, script dir
16 my_file = os.path.abspath(sys.argv[0])
17 my_dir = os.path.dirname(my_file)
18
19 # try to automatically determine the pytan lib directory by assuming it is in '../..lib/'
20 parent_dir = os.path.dirname(my_dir)
21 pytan_root_dir = os.path.dirname(parent_dir)
22 lib_dir = os.path.join(pytan_root_dir, 'lib')
23
24 # add pytan_loc and lib_dir to the PYTHONPATH variable

```

```
25 path_adds = [lib_dir, pytan_static_path]
26 [sys.path.append(aa) for aa in path_adds if aa not in sys.path]
27
28 # import pytan
29 import pytan
30
31 # create a dictionary of arguments for the pytan handler
32 handler_args = {}
33
34 # establish our connection info for the Tanium Server
35 handler_args['username'] = "Administrator"
36 handler_args['password'] = "Tanium2015!"
37 handler_args['host'] = "10.0.1.240"
38 handler_args['port'] = "443" # optional
39
40 # optional, level 0 is no output except warnings/errors
41 # level 1 through 12 are more and more verbose
42 handler_args['loglevel'] = 1
43
44 # optional, use a debug format for the logging output (uses two lines per log entry)
45 handler_args['debugformat'] = False
46
47 # optional, this saves all response objects to handler.session.ALL_REQUESTS_RESPONSES
48 # very useful for capturing the full exchange of XML requests and responses
49 handler_args['record_all_requests'] = True
50
51 # instantiate a handler using all of the arguments in the handler_args dictionary
52 print "...CALLING: pytan.handler() with args: {}".format(handler_args)
53 handler = pytan.Handler(**handler_args)
54
55 # print out the handler string
56 print "...OUTPUT: handler string: {}".format(handler)
57
58 # setup the arguments for the handler() class
59 kwargs = {}
60 kwargs["qtype"] = u'saved'
61 kwargs["name"] = u'Installed Applications'
62
63 print "...CALLING: handler.ask with args: {}".format(kwargs)
64 response = handler.ask(**kwargs)
65
66 print "...OUTPUT: Type of response: ", type(response)
67
68 print "...OUTPUT: Pretty print of response:"
69 print pprint.pformat(response)
70
71 print "...OUTPUT: Equivalent Question if it were to be asked in the Tanium Console: "
72 print response['question_object'].query_text
73
74 # call the export_obj() method to convert response to CSV and store it in out
75 export_kwargs = {}
76 export_kwargs['obj'] = response['question_results']
77 export_kwargs['export_format'] = 'csv'
78
79 print "...CALLING: handler.export_obj() with args {}".format(export_kwargs)
80 out = handler.export_obj(**export_kwargs)
81
82 # trim the output if it is more than 15 lines long
```

```

83 if len(out.splitlines()) > 15:
84     out = out.splitlines()[0:15]
85     out.append('...trimmed for brevity..')
86     out = '\n'.join(out)
87
88 print "...OUTPUT: CSV Results of response: "
89 print out

```

### Ask Saved Question By Name In List

Ask a saved question by referencing the name of a saved question in a list of strings.

- STDOUT from Example Python Code
- STDERR from Example Python Code
- Example Python Code

```

1  # import the basic python packages we need
2  import os
3  import sys
4  import tempfile
5  import pprint
6  import traceback
7
8  # disable python from generating a .pyc file
9  sys.dont_write_bytecode = True
10
11 # change me to the path of pytan if this script is not running from EXAMPLES/PYTAN_API
12 pytan_loc = "~/gh/pytan"
13 pytan_static_path = os.path.join(os.path.expanduser(pytan_loc), 'lib')
14
15 # Determine our script name, script dir
16 my_file = os.path.abspath(sys.argv[0])
17 my_dir = os.path.dirname(my_file)
18
19 # try to automatically determine the pytan lib directory by assuming it is in '../..lib/'
20 parent_dir = os.path.dirname(my_dir)
21 pytan_root_dir = os.path.dirname(parent_dir)
22 lib_dir = os.path.join(pytan_root_dir, 'lib')
23
24 # add pytan_loc and lib_dir to the PYTHONPATH variable
25 path_adds = [lib_dir, pytan_static_path]
26 [sys.path.append(aa) for aa in path_adds if aa not in sys.path]
27
28 # import pytan
29 import pytan
30
31 # create a dictionary of arguments for the pytan handler
32 handler_args = {}
33
34 # establish our connection info for the Tanium Server
35 handler_args['username'] = "Administrator"
36 handler_args['password'] = "Tanium2015!"
37 handler_args['host'] = "10.0.1.240"
38 handler_args['port'] = "443" # optional
39
40 # optional, level 0 is no output except warnings/errors

```

```
41 # level 1 through 12 are more and more verbose
42 handler_args['loglevel'] = 1
43
44 # optional, use a debug format for the logging output (uses two lines per log entry)
45 handler_args['debugformat'] = False
46
47 # optional, this saves all response objects to handler.session.ALL_REQUESTS_RESPONSES
48 # very useful for capturing the full exchange of XML requests and responses
49 handler_args['record_all_requests'] = True
50
51 # instantiate a handler using all of the arguments in the handler_args dictionary
52 print "...CALLING: pytan.handler() with args: {}".format(handler_args)
53 handler = pytan.Handler(**handler_args)
54
55 # print out the handler string
56 print "...OUTPUT: handler string: {}".format(handler)
57
58 # setup the arguments for the handler() class
59 kwargs = {}
60 kwargs["qtype"] = u'saved'
61 kwargs["name"] = [u'Installed Applications']
62
63 print "...CALLING: handler.ask with args: {}".format(kwargs)
64 response = handler.ask(**kwargs)
65
66 print "...OUTPUT: Type of response: ", type(response)
67
68 print "...OUTPUT: Pretty print of response:"
69 print pprint.pformat(response)
70
71 print "...OUTPUT: Equivalent Question if it were to be asked in the Tanium Console: "
72 print response['question_object'].query_text
73
74 # call the export_obj() method to convert response to CSV and store it in out
75 export_kwargs = {}
76 export_kwargs['obj'] = response['question_results']
77 export_kwargs['export_format'] = 'csv'
78
79 print "...CALLING: handler.export_obj() with args {}".format(export_kwargs)
80 out = handler.export_obj(**export_kwargs)
81
82 # trim the output if it is more than 15 lines long
83 if len(out.splitlines()) > 15:
84     out = out.splitlines()[0:15]
85     out.append('..trimmed for brevity..')
86     out = '\n'.join(out)
87
88 print "...OUTPUT: CSV Results of response: "
89 print out
```

## PyTan API Invalid Create Object Examples

All of the PyTan API examples for Invalid Create Object

## Invalid Create Sensor

Create a sensor (Unsupported!)

- STDOUT from Example Python Code
- STDERR from Example Python Code
- Example Python Code

```

1  # import the basic python packages we need
2  import os
3  import sys
4  import tempfile
5  import pprint
6  import traceback
7
8  # disable python from generating a .pyc file
9  sys.dont_write_bytecode = True
10
11  # change me to the path of pytan if this script is not running from EXAMPLES/PYTAN_API
12  pytan_loc = "~/gh/pytan"
13  pytan_static_path = os.path.join(os.path.expanduser(pytan_loc), 'lib')
14
15  # Determine our script name, script dir
16  my_file = os.path.abspath(sys.argv[0])
17  my_dir = os.path.dirname(my_file)
18
19  # try to automatically determine the pytan lib directory by assuming it is in '../..lib/'
20  parent_dir = os.path.dirname(my_dir)
21  pytan_root_dir = os.path.dirname(parent_dir)
22  lib_dir = os.path.join(pytan_root_dir, 'lib')
23
24  # add pytan_loc and lib_dir to the PYTHONPATH variable
25  path_adds = [lib_dir, pytan_static_path]
26  [sys.path.append(aa) for aa in path_adds if aa not in sys.path]
27
28  # import pytan
29  import pytan
30
31  # create a dictionary of arguments for the pytan handler
32  handler_args = {}
33
34  # establish our connection info for the Tanium Server
35  handler_args['username'] = "Administrator"
36  handler_args['password'] = "Tanium2015!"
37  handler_args['host'] = "10.0.1.240"
38  handler_args['port'] = "443" # optional
39
40  # optional, level 0 is no output except warnings/errors
41  # level 1 through 12 are more and more verbose
42  handler_args['loglevel'] = 1
43
44  # optional, use a debug format for the logging output (uses two lines per log entry)
45  handler_args['debugformat'] = False
46
47  # optional, this saves all response objects to handler.session.ALL_REQUESTS_RESPONSES
48  # very useful for capturing the full exchange of XML requests and responses
49  handler_args['record_all_requests'] = True

```

```
50
51 # instantiate a handler using all of the arguments in the handler_args dictionary
52 print "...CALLING: pytan.handler() with args: {}".format(handler_args)
53 handler = pytan.Handler(**handler_args)
54
55 # print out the handler string
56 print "...OUTPUT: handler string: {}".format(handler)
57
58 # setup the arguments for the handler() class
59 kwargs = {}
60 kwargs["unsupported"] = True
61
62 print "...CALLING: handler.create_sensor() with args: {}".format(kwargs)
63 try:
64     handler.create_sensor(**kwargs)
65 except Exception as e:
66     print "...EXCEPTION: {}".format(e)
67     # this should throw an exception of type: pytan.exceptions.HandlerError
68     # uncomment to see full exception
69     # traceback.print_exc(file=sys.stdout)
```

## PyTan API Invalid Create Object From JSON Examples

All of the PyTan API examples for Invalid Create Object From JSON

### Invalid Create Saved Action From JSON

Create a saved action from json (not supported!)

- STDOUT from Example Python Code
- STDERR from Example Python Code
- Example Python Code

```
1 # import the basic python packages we need
2 import os
3 import sys
4 import tempfile
5 import pprint
6 import traceback
7
8 # disable python from generating a .pyc file
9 sys.dont_write_bytecode = True
10
11 # change me to the path of pytan if this script is not running from EXAMPLES/PYTAN_API
12 pytan_loc = "~/gh/pytan"
13 pytan_static_path = os.path.join(os.path.expanduser(pytan_loc), 'lib')
14
15 # Determine our script name, script dir
16 my_file = os.path.abspath(sys.argv[0])
17 my_dir = os.path.dirname(my_file)
18
19 # try to automatically determine the pytan lib directory by assuming it is in '../..lib/'
20 parent_dir = os.path.dirname(my_dir)
21 pytan_root_dir = os.path.dirname(parent_dir)
22 lib_dir = os.path.join(pytan_root_dir, 'lib')
```

```

23
24 # add pytan_loc and lib_dir to the PYTHONPATH variable
25 path_adds = [lib_dir, pytan_static_path]
26 [sys.path.append(aa) for aa in path_adds if aa not in sys.path]
27
28 # import pytan
29 import pytan
30
31 # create a dictionary of arguments for the pytan handler
32 handler_args = {}
33
34 # establish our connection info for the Tanium Server
35 handler_args['username'] = "Administrator"
36 handler_args['password'] = "Tanium2015!"
37 handler_args['host'] = "10.0.1.240"
38 handler_args['port'] = "443" # optional
39
40 # optional, level 0 is no output except warnings/errors
41 # level 1 through 12 are more and more verbose
42 handler_args['loglevel'] = 1
43
44 # optional, use a debug format for the logging output (uses two lines per log entry)
45 handler_args['debugformat'] = False
46
47 # optional, this saves all response objects to handler.session.ALL_REQUESTS_RESPONSES
48 # very useful for capturing the full exchange of XML requests and responses
49 handler_args['record_all_requests'] = True
50
51 # instantiate a handler using all of the arguments in the handler_args dictionary
52 print "...CALLING: pytan.handler() with args: {}".format(handler_args)
53 handler = pytan.Handler(**handler_args)
54
55 # print out the handler string
56 print "...OUTPUT: handler string: {}".format(handler)
57
58 # setup the arguments for the handler.get() method
59 get_kwargs = {}
60 get_kwargs["objtype"] = u'saved_action'
61 get_kwargs["name"] = u'Distribute Tanium Standard Utilities'
62
63 # get objects to use as an export to JSON file
64 print "...CALLING: handler.get() with args: {}".format(get_kwargs)
65 orig_objs = handler.get(**get_kwargs)
66
67 # export orig_objs to a json file
68 export_kwargs = {}
69 export_kwargs['obj'] = orig_objs
70 export_kwargs['export_format'] = 'json'
71 export_kwargs['report_dir'] = tempfile.gettempdir()
72
73 print "...CALLING: handler.export_to_report_file() with args: {}".format(export_kwargs)
74 json_file, results = handler.export_to_report_file(**export_kwargs)
75
76 # create the object from the exported JSON file
77 create_kwargs = {}
78 create_kwargs['objtype'] = u'saved_action'
79 create_kwargs['json_file'] = json_file
80

```

```
81 # call the handler with the create_from_json method, passing in kwargs for arguments
82 print "...CALLING: handler.create_from_json() with args {}".format(create_kwargs)
83 try:
84     response = handler.create_from_json(**create_kwargs)
85 except Exception as e:
86     print "...EXCEPTION: {}".format(e)
87     # this should throw an exception of type: pytan.exceptions.HandlerError
88     # uncomment to see full exception
89     # traceback.print_exc(file=sys.stdout)
```

## Invalid Create Client From JSON

Create a client from json (not supported!)

- STDOUT from Example Python Code
- STDERR from Example Python Code
- Example Python Code

```
1  # import the basic python packages we need
2  import os
3  import sys
4  import tempfile
5  import pprint
6  import traceback
7
8  # disable python from generating a .pyc file
9  sys.dont_write_bytecode = True
10
11 # change me to the path of pytan if this script is not running from EXAMPLES/PYTAN_API
12 pytan_loc = "~/gh/pytan"
13 pytan_static_path = os.path.join(os.path.expanduser(pytan_loc), 'lib')
14
15 # Determine our script name, script dir
16 my_file = os.path.abspath(sys.argv[0])
17 my_dir = os.path.dirname(my_file)
18
19 # try to automatically determine the pytan lib directory by assuming it is in '../..lib/'
20 parent_dir = os.path.dirname(my_dir)
21 pytan_root_dir = os.path.dirname(parent_dir)
22 lib_dir = os.path.join(pytan_root_dir, 'lib')
23
24 # add pytan_loc and lib_dir to the PYTHONPATH variable
25 path_adds = [lib_dir, pytan_static_path]
26 [sys.path.append(aa) for aa in path_adds if aa not in sys.path]
27
28 # import pytan
29 import pytan
30
31 # create a dictionary of arguments for the pytan handler
32 handler_args = {}
33
34 # establish our connection info for the Tanium Server
35 handler_args['username'] = "Administrator"
36 handler_args['password'] = "Tanium2015!"
37 handler_args['host'] = "10.0.1.240"
38 handler_args['port'] = "443" # optional
```



```

39
40 # optional, level 0 is no output except warnings/errors
41 # level 1 through 12 are more and more verbose
42 handler_args['loglevel'] = 1
43
44 # optional, use a debug format for the logging output (uses two lines per log entry)
45 handler_args['debugformat'] = False
46
47 # optional, this saves all response objects to handler.session.ALL_REQUESTS_RESPONSES
48 # very useful for capturing the full exchange of XML requests and responses
49 handler_args['record_all_requests'] = True
50
51 # instantiate a handler using all of the arguments in the handler_args dictionary
52 print "...CALLING: pytan.handler() with args: {}".format(handler_args)
53 handler = pytan.Handler(**handler_args)
54
55 # print out the handler string
56 print "...OUTPUT: handler string: {}".format(handler)
57
58 # setup the arguments for the handler.get() method
59 get_kwargs = {}
60 get_kwargs["objtype"] = u'client'
61 get_kwargs["status"] = u'Leader'
62
63 # get objects to use as an export to JSON file
64 print "...CALLING: handler.get() with args: {}".format(get_kwargs)
65 orig_objs = handler.get(**get_kwargs)
66
67 # export orig_objs to a json file
68 export_kwargs = {}
69 export_kwargs['obj'] = orig_objs
70 export_kwargs['export_format'] = 'json'
71 export_kwargs['report_dir'] = tempfile.gettempdir()
72
73 print "...CALLING: handler.export_to_report_file() with args: {}".format(export_kwargs)
74 json_file, results = handler.export_to_report_file(**export_kwargs)
75
76 # create the object from the exported JSON file
77 create_kwargs = {}
78 create_kwargs['objtype'] = u'client'
79 create_kwargs['json_file'] = json_file
80
81 # call the handler with the create_from_json method, passing in kwargs for arguments
82 print "...CALLING: handler.create_from_json() with args {}".format(create_kwargs)
83 try:
84     response = handler.create_from_json(**create_kwargs)
85 except Exception as e:
86     print "...EXCEPTION: {}".format(e)
87     # this should throw an exception of type: pytan.exceptions.HandlerError
88     # uncomment to see full exception
89     # traceback.print_exc(file=sys.stdout)

```

### Invalid Create Userrole From JSON

Create a user role from json (not supported!)

- STDOUT from Example Python Code

- STDERR from Example Python Code
- Example Python Code

```
1  # import the basic python packages we need
2  import os
3  import sys
4  import tempfile
5  import pprint
6  import traceback
7
8  # disable python from generating a .pyc file
9  sys.dont_write_bytecode = True
10
11  # change me to the path of pytan if this script is not running from EXAMPLES/PYTAN_API
12  pytan_loc = "~/gh/pytan"
13  pytan_static_path = os.path.join(os.path.expanduser(pytan_loc), 'lib')
14
15  # Determine our script name, script dir
16  my_file = os.path.abspath(sys.argv[0])
17  my_dir = os.path.dirname(my_file)
18
19  # try to automatically determine the pytan lib directory by assuming it is in '../..lib/'
20  parent_dir = os.path.dirname(my_dir)
21  pytan_root_dir = os.path.dirname(parent_dir)
22  lib_dir = os.path.join(pytan_root_dir, 'lib')
23
24  # add pytan_loc and lib_dir to the PYTHONPATH variable
25  path_adds = [lib_dir, pytan_static_path]
26  [sys.path.append(aa) for aa in path_adds if aa not in sys.path]
27
28  # import pytan
29  import pytan
30
31  # create a dictionary of arguments for the pytan handler
32  handler_args = {}
33
34  # establish our connection info for the Tanium Server
35  handler_args['username'] = "Administrator"
36  handler_args['password'] = "Tanium2015!"
37  handler_args['host'] = "10.0.1.240"
38  handler_args['port'] = "443" # optional
39
40  # optional, level 0 is no output except warnings/errors
41  # level 1 through 12 are more and more verbose
42  handler_args['loglevel'] = 1
43
44  # optional, use a debug format for the logging output (uses two lines per log entry)
45  handler_args['debugformat'] = False
46
47  # optional, this saves all response objects to handler.session.ALL_REQUESTS_RESPONSES
48  # very useful for capturing the full exchange of XML requests and responses
49  handler_args['record_all_requests'] = True
50
51  # instantiate a handler using all of the arguments in the handler_args dictionary
52  print "...CALLING: pytan.handler() with args: {}".format(handler_args)
53  handler = pytan.Handler(**handler_args)
54
55  # print out the handler string
```

```

56 print "...OUTPUT: handler string: {}".format(handler)
57
58 # setup the arguments for the handler.get() method
59 get_kwargs = {}
60 get_kwargs["objtype"] = u'userrole'
61 get_kwargs["name"] = u'Administrator'
62
63 # get objects to use as an export to JSON file
64 print "...CALLING: handler.get() with args: {}".format(get_kwargs)
65 orig_objs = handler.get(**get_kwargs)
66
67 # export orig_objs to a json file
68 export_kwargs = {}
69 export_kwargs['obj'] = orig_objs
70 export_kwargs['export_format'] = 'json'
71 export_kwargs['report_dir'] = tempfile.gettempdir()
72
73 print "...CALLING: handler.export_to_report_file() with args: {}".format(export_kwargs)
74 json_file, results = handler.export_to_report_file(**export_kwargs)
75
76 # create the object from the exported JSON file
77 create_kwargs = {}
78 create_kwargs['objtype'] = u'userrole'
79 create_kwargs['json_file'] = json_file
80
81 # call the handler with the create_from_json method, passing in kwargs for arguments
82 print "...CALLING: handler.create_from_json() with args {}".format(create_kwargs)
83 try:
84     response = handler.create_from_json(**create_kwargs)
85 except Exception as e:
86     print "...EXCEPTION: {}".format(e)
87     # this should throw an exception of type: pytan.exceptions.HandlerError
88     # uncomment to see full exception
89     # traceback.print_exc(file=sys.stdout)

```

### Invalid Create Setting From JSON

Create a setting from json (not supported!)

- STDOUT from Example Python Code
- STDERR from Example Python Code
- Example Python Code

```

1 # import the basic python packages we need
2 import os
3 import sys
4 import tempfile
5 import pprint
6 import traceback
7
8 # disable python from generating a .pyc file
9 sys.dont_write_bytecode = True
10
11 # change me to the path of pytan if this script is not running from EXAMPLES/PYTAN_API
12 pytan_loc = "~/gh/pytan"
13 pytan_static_path = os.path.join(os.path.expanduser(pytan_loc), 'lib')

```

```
14
15 # Determine our script name, script dir
16 my_file = os.path.abspath(sys.argv[0])
17 my_dir = os.path.dirname(my_file)
18
19 # try to automatically determine the pytan lib directory by assuming it is in '../..lib/'
20 parent_dir = os.path.dirname(my_dir)
21 pytan_root_dir = os.path.dirname(parent_dir)
22 lib_dir = os.path.join(pytan_root_dir, 'lib')
23
24 # add pytan_loc and lib_dir to the PYTHONPATH variable
25 path_adds = [lib_dir, pytan_static_path]
26 [sys.path.append(aa) for aa in path_adds if aa not in sys.path]
27
28 # import pytan
29 import pytan
30
31 # create a dictionary of arguments for the pytan handler
32 handler_args = {}
33
34 # establish our connection info for the Tanium Server
35 handler_args['username'] = "Administrator"
36 handler_args['password'] = "Tanium2015!"
37 handler_args['host'] = "10.0.1.240"
38 handler_args['port'] = "443" # optional
39
40 # optional, level 0 is no output except warnings/errors
41 # level 1 through 12 are more and more verbose
42 handler_args['loglevel'] = 1
43
44 # optional, use a debug format for the logging output (uses two lines per log entry)
45 handler_args['debugformat'] = False
46
47 # optional, this saves all response objects to handler.session.ALL_REQUESTS_RESPONSES
48 # very useful for capturing the full exchange of XML requests and responses
49 handler_args['record_all_requests'] = True
50
51 # instantiate a handler using all of the arguments in the handler_args dictionary
52 print "...CALLING: pytan.handler() with args: {}".format(handler_args)
53 handler = pytan.Handler(**handler_args)
54
55 # print out the handler string
56 print "...OUTPUT: handler string: {}".format(handler)
57
58 # setup the arguments for the handler.get() method
59 get_kwargs = {}
60 get_kwargs["objtype"] = u'setting'
61 get_kwargs["id"] = 1
62
63 # get objects to use as an export to JSON file
64 print "...CALLING: handler.get() with args: {}".format(get_kwargs)
65 orig_objs = handler.get(**get_kwargs)
66
67 # export orig_objs to a json file
68 export_kwargs = {}
69 export_kwargs['obj'] = orig_objs
70 export_kwargs['export_format'] = 'json'
71 export_kwargs['report_dir'] = tempfile.gettempdir()
```

```

72
73 print "...CALLING: handler.export_to_report_file() with args: {}".format(export_kwargs)
74 json_file, results = handler.export_to_report_file(**export_kwargs)
75
76 # create the object from the exported JSON file
77 create_kwargs = {}
78 create_kwargs['objtype'] = u'setting'
79 create_kwargs['json_file'] = json_file
80
81 # call the handler with the create_from_json method, passing in kwargs for arguments
82 print "...CALLING: handler.create_from_json() with args {}".format(create_kwargs)
83 try:
84     response = handler.create_from_json(**create_kwargs)
85 except Exception as e:
86     print "...EXCEPTION: {}".format(e)
87     # this should throw an exception of type: pytan.exceptions.HandlerError
88     # uncomment to see full exception
89     # traceback.print_exc(file=sys.stdout)

```

## PyTan API Invalid Deploy Action Examples

All of the PyTan API examples for Invalid Deploy Action

### 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

- STDOUT from Example Python Code
- STDERR from Example Python Code
- Example Python Code

```

1  # import the basic python packages we need
2  import os
3  import sys
4  import tempfile
5  import pprint
6  import traceback
7
8  # disable python from generating a .pyc file
9  sys.dont_write_bytecode = True
10
11 # change me to the path of pytan if this script is not running from EXAMPLES/PYTAN_API
12 pytan_loc = "~/gh/pytan"
13 pytan_static_path = os.path.join(os.path.expanduser(pytan_loc), 'lib')
14
15 # Determine our script name, script dir
16 my_file = os.path.abspath(sys.argv[0])
17 my_dir = os.path.dirname(my_file)
18
19 # try to automatically determine the pytan lib directory by assuming it is in '../..lib/'
20 parent_dir = os.path.dirname(my_dir)
21 pytan_root_dir = os.path.dirname(parent_dir)
22 lib_dir = os.path.join(pytan_root_dir, 'lib')
23

```

```
24 # add pytan_loc and lib_dir to the PYTHONPATH variable
25 path_adds = [lib_dir, pytan_static_path]
26 [sys.path.append(aa) for aa in path_adds if aa not in sys.path]
27
28 # import pytan
29 import pytan
30
31 # create a dictionary of arguments for the pytan handler
32 handler_args = {}
33
34 # establish our connection info for the Tanium Server
35 handler_args['username'] = "Administrator"
36 handler_args['password'] = "Tanium2015!"
37 handler_args['host'] = "10.0.1.240"
38 handler_args['port'] = "443" # optional
39
40 # optional, level 0 is no output except warnings/errors
41 # level 1 through 12 are more and more verbose
42 handler_args['loglevel'] = 1
43
44 # optional, use a debug format for the logging output (uses two lines per log entry)
45 handler_args['debugformat'] = False
46
47 # optional, this saves all response objects to handler.session.ALL_REQUESTS_RESPONSES
48 # very useful for capturing the full exchange of XML requests and responses
49 handler_args['record_all_requests'] = True
50
51 # instantiate a handler using all of the arguments in the handler_args dictionary
52 print "...CALLING: pytan.handler() with args: {}".format(handler_args)
53 handler = pytan.Handler(**handler_args)
54
55 # print out the handler string
56 print "...OUTPUT: handler string: {}".format(handler)
57
58 # setup the arguments for the handler() class
59 kwargs = {}
60 kwargs["report_dir"] = u'/tmp'
61 kwargs["package"] = u'Distribute Tanium Standard Utilities'
62
63 print "...CALLING: handler.deploy_action() with args: {}".format(kwargs)
64 try:
65     handler.deploy_action(**kwargs)
66 except Exception as e:
67     print "...EXCEPTION: {}".format(e)
68     # this should throw an exception of type: pytan.exceptions.RunFalse
69     # uncomment to see full exception
70     # traceback.print_exc(file=sys.stdout)
```

## Invalid Deploy Action Package Help

Have `deploy_action()` return the help for package

- STDOUT from Example Python Code
- STDERR from Example Python Code
- Example Python Code

```

1  # import the basic python packages we need
2  import os
3  import sys
4  import tempfile
5  import pprint
6  import traceback
7
8  # disable python from generating a .pyc file
9  sys.dont_write_bytecode = True
10
11  # change me to the path of pytan if this script is not running from EXAMPLES/PYTAN_API
12  pytan_loc = "~/gh/pytan"
13  pytan_static_path = os.path.join(os.path.expanduser(pytan_loc), 'lib')
14
15  # Determine our script name, script dir
16  my_file = os.path.abspath(sys.argv[0])
17  my_dir = os.path.dirname(my_file)
18
19  # try to automatically determine the pytan lib directory by assuming it is in '../..lib/'
20  parent_dir = os.path.dirname(my_dir)
21  pytan_root_dir = os.path.dirname(parent_dir)
22  lib_dir = os.path.join(pytan_root_dir, 'lib')
23
24  # add pytan_loc and lib_dir to the PYTHONPATH variable
25  path_adds = [lib_dir, pytan_static_path]
26  [sys.path.append(aa) for aa in path_adds if aa not in sys.path]
27
28  # import pytan
29  import pytan
30
31  # create a dictionary of arguments for the pytan handler
32  handler_args = {}
33
34  # establish our connection info for the Tanium Server
35  handler_args['username'] = "Administrator"
36  handler_args['password'] = "Tanium2015!"
37  handler_args['host'] = "10.0.1.240"
38  handler_args['port'] = "443" # optional
39
40  # optional, level 0 is no output except warnings/errors
41  # level 1 through 12 are more and more verbose
42  handler_args['loglevel'] = 1
43
44  # optional, use a debug format for the logging output (uses two lines per log entry)
45  handler_args['debugformat'] = False
46
47  # optional, this saves all response objects to handler.session.ALL_REQUESTS_RESPONSES
48  # very useful for capturing the full exchange of XML requests and responses
49  handler_args['record_all_requests'] = True
50
51  # instantiate a handler using all of the arguments in the handler_args dictionary
52  print "...CALLING: pytan.handler() with args: {}".format(handler_args)
53  handler = pytan.Handler(**handler_args)
54
55  # print out the handler string
56  print "...OUTPUT: handler string: {}".format(handler)
57
58  # setup the arguments for the handler() class

```

```
59 kwargs = {}
60 kwargs["package_help"] = True
61
62 print "...CALLING: handler.deploy_action() with args: {}".format(kwargs)
63 try:
64     handler.deploy_action(**kwargs)
65 except Exception as e:
66     print "...EXCEPTION: {}".format(e)
67     # this should throw an exception of type: pytan.exceptions.PytanHelp
68     # uncomment to see full exception
69     # traceback.print_exc(file=sys.stdout)
```

## Invalid Deploy Action Package

Deploy an action using a non-existing package.

- STDOUT from Example Python Code
- STDERR from Example Python Code
- Example Python Code

```
1  # import the basic python packages we need
2  import os
3  import sys
4  import tempfile
5  import pprint
6  import traceback
7
8  # disable python from generating a .pyc file
9  sys.dont_write_bytecode = True
10
11  # change me to the path of pytan if this script is not running from EXAMPLES/PYTAN_API
12  pytan_loc = "~/gh/pytan"
13  pytan_static_path = os.path.join(os.path.expanduser(pytan_loc), 'lib')
14
15  # Determine our script name, script dir
16  my_file = os.path.abspath(sys.argv[0])
17  my_dir = os.path.dirname(my_file)
18
19  # try to automatically determine the pytan lib directory by assuming it is in '../..lib/'
20  parent_dir = os.path.dirname(my_dir)
21  pytan_root_dir = os.path.dirname(parent_dir)
22  lib_dir = os.path.join(pytan_root_dir, 'lib')
23
24  # add pytan_loc and lib_dir to the PYTHONPATH variable
25  path_adds = [lib_dir, pytan_static_path]
26  [sys.path.append(aa) for aa in path_adds if aa not in sys.path]
27
28  # import pytan
29  import pytan
30
31  # create a dictionary of arguments for the pytan handler
32  handler_args = {}
33
34  # establish our connection info for the Tanium Server
35  handler_args['username'] = "Administrator"
36  handler_args['password'] = "Tanium2015!"
```



```

37 handler_args['host'] = "10.0.1.240"
38 handler_args['port'] = "443" # optional
39
40 # optional, level 0 is no output except warnings/errors
41 # level 1 through 12 are more and more verbose
42 handler_args['loglevel'] = 1
43
44 # optional, use a debug format for the logging output (uses two lines per log entry)
45 handler_args['debugformat'] = False
46
47 # optional, this saves all response objects to handler.session.ALL_REQUESTS_RESPONSES
48 # very useful for capturing the full exchange of XML requests and responses
49 handler_args['record_all_requests'] = True
50
51 # instantiate a handler using all of the arguments in the handler_args dictionary
52 print "...CALLING: pytan.handler() with args: {}".format(handler_args)
53 handler = pytan.Handler(**handler_args)
54
55 # print out the handler string
56 print "...OUTPUT: handler string: {}".format(handler)
57
58 # setup the arguments for the handler() class
59 kwargs = {}
60 kwargs["run"] = True
61 kwargs["package"] = u'Invalid Package'
62
63 print "...CALLING: handler.deploy_action() with args: {}".format(kwargs)
64 try:
65     handler.deploy_action(**kwargs)
66 except Exception as e:
67     print "...EXCEPTION: {}".format(e)
68     # this should throw an exception of type: pytan.exceptions.HandlerError
69     # uncomment to see full exception
70     # traceback.print_exc(file=sys.stdout)

```

### Invalid Deploy Action Options Help

Have `deploy_action()` return the help for options

- STDOUT from Example Python Code
- STDERR from Example Python Code
- Example Python Code

```

1 # import the basic python packages we need
2 import os
3 import sys
4 import tempfile
5 import pprint
6 import traceback
7
8 # disable python from generating a .pyc file
9 sys.dont_write_bytecode = True
10
11 # change me to the path of pytan if this script is not running from EXAMPLES/PYTAN_API
12 pytan_loc = "~/gh/pytan"
13 pytan_static_path = os.path.join(os.path.expanduser(pytan_loc), 'lib')

```

```
14
15 # Determine our script name, script dir
16 my_file = os.path.abspath(sys.argv[0])
17 my_dir = os.path.dirname(my_file)
18
19 # try to automatically determine the pytan lib directory by assuming it is in '../..lib/'
20 parent_dir = os.path.dirname(my_dir)
21 pytan_root_dir = os.path.dirname(parent_dir)
22 lib_dir = os.path.join(pytan_root_dir, 'lib')
23
24 # add pytan_loc and lib_dir to the PYTHONPATH variable
25 path_adds = [lib_dir, pytan_static_path]
26 [sys.path.append(aa) for aa in path_adds if aa not in sys.path]
27
28 # import pytan
29 import pytan
30
31 # create a dictionary of arguments for the pytan handler
32 handler_args = {}
33
34 # establish our connection info for the Tanium Server
35 handler_args['username'] = "Administrator"
36 handler_args['password'] = "Tanium2015!"
37 handler_args['host'] = "10.0.1.240"
38 handler_args['port'] = "443" # optional
39
40 # optional, level 0 is no output except warnings/errors
41 # level 1 through 12 are more and more verbose
42 handler_args['loglevel'] = 1
43
44 # optional, use a debug format for the logging output (uses two lines per log entry)
45 handler_args['debugformat'] = False
46
47 # optional, this saves all response objects to handler.session.ALL_REQUESTS_RESPONSES
48 # very useful for capturing the full exchange of XML requests and responses
49 handler_args['record_all_requests'] = True
50
51 # instantiate a handler using all of the arguments in the handler_args dictionary
52 print "...CALLING: pytan.handler() with args: {}".format(handler_args)
53 handler = pytan.Handler(**handler_args)
54
55 # print out the handler string
56 print "...OUTPUT: handler string: {}".format(handler)
57
58 # setup the arguments for the handler() class
59 kwargs = {}
60 kwargs["options_help"] = True
61
62 print "...CALLING: handler.deploy_action() with args: {}".format(kwargs)
63 try:
64     handler.deploy_action(**kwargs)
65 except Exception as e:
66     print "...EXCEPTION: {}".format(e)
67     # this should throw an exception of type: pytan.exceptions.PytanHelp
68     # uncomment to see full exception
69     # traceback.print_exc(file=sys.stdout)
```

## Invalid Deploy Action Empty Package

Deploy an action using an empty package string.

- STDOUT from Example Python Code
- STDERR from Example Python Code
- Example Python Code

```

1  # import the basic python packages we need
2  import os
3  import sys
4  import tempfile
5  import pprint
6  import traceback
7
8  # disable python from generating a .pyc file
9  sys.dont_write_bytecode = True
10
11 # change me to the path of pytan if this script is not running from EXAMPLES/PYTAN_API
12 pytan_loc = "~/gh/pytan"
13 pytan_static_path = os.path.join(os.path.expanduser(pytan_loc), 'lib')
14
15 # Determine our script name, script dir
16 my_file = os.path.abspath(sys.argv[0])
17 my_dir = os.path.dirname(my_file)
18
19 # try to automatically determine the pytan lib directory by assuming it is in '../..lib/'
20 parent_dir = os.path.dirname(my_dir)
21 pytan_root_dir = os.path.dirname(parent_dir)
22 lib_dir = os.path.join(pytan_root_dir, 'lib')
23
24 # add pytan_loc and lib_dir to the PYTHONPATH variable
25 path_adds = [lib_dir, pytan_static_path]
26 [sys.path.append(aa) for aa in path_adds if aa not in sys.path]
27
28 # import pytan
29 import pytan
30
31 # create a dictionary of arguments for the pytan handler
32 handler_args = {}
33
34 # establish our connection info for the Tanium Server
35 handler_args['username'] = "Administrator"
36 handler_args['password'] = "Tanium2015!"
37 handler_args['host'] = "10.0.1.240"
38 handler_args['port'] = "443" # optional
39
40 # optional, level 0 is no output except warnings/errors
41 # level 1 through 12 are more and more verbose
42 handler_args['loglevel'] = 1
43
44 # optional, use a debug format for the logging output (uses two lines per log entry)
45 handler_args['debugformat'] = False
46
47 # optional, this saves all response objects to handler.session.ALL_REQUESTS_RESPONSES
48 # very useful for capturing the full exchange of XML requests and responses
49 handler_args['record_all_requests'] = True

```

```
50
51 # instantiate a handler using all of the arguments in the handler_args dictionary
52 print "...CALLING: pytan.handler() with args: {}".format(handler_args)
53 handler = pytan.Handler(**handler_args)
54
55 # print out the handler string
56 print "...OUTPUT: handler string: {}".format(handler)
57
58 # setup the arguments for the handler() class
59 kwargs = {}
60 kwargs["run"] = True
61 kwargs["package"] = u''
62
63 print "...CALLING: handler.deploy_action() with args: {}".format(kwargs)
64 try:
65     handler.deploy_action(**kwargs)
66 except Exception as e:
67     print "...EXCEPTION: {}".format(e)
68     # this should throw an exception of type: pytan.exceptions.HumanParserError
69     # uncomment to see full exception
70     # traceback.print_exc(file=sys.stdout)
```

### Invalid Deploy Action Filters Help

Have `deploy_action()` return the help for filters

- STDOUT from Example Python Code
- STDERR from Example Python Code
- Example Python Code

```
1 # import the basic python packages we need
2 import os
3 import sys
4 import tempfile
5 import pprint
6 import traceback
7
8 # disable python from generating a .pyc file
9 sys.dont_write_bytecode = True
10
11 # change me to the path of pytan if this script is not running from EXAMPLES/PYTAN_API
12 pytan_loc = "~/gh/pytan"
13 pytan_static_path = os.path.join(os.path.expanduser(pytan_loc), 'lib')
14
15 # Determine our script name, script dir
16 my_file = os.path.abspath(sys.argv[0])
17 my_dir = os.path.dirname(my_file)
18
19 # try to automatically determine the pytan lib directory by assuming it is in '../..lib/'
20 parent_dir = os.path.dirname(my_dir)
21 pytan_root_dir = os.path.dirname(parent_dir)
22 lib_dir = os.path.join(pytan_root_dir, 'lib')
23
24 # add pytan_loc and lib_dir to the PYTHONPATH variable
25 path_adds = [lib_dir, pytan_static_path]
26 [sys.path.append(aa) for aa in path_adds if aa not in sys.path]
```

```

27
28 # import pytan
29 import pytan
30
31 # create a dictionary of arguments for the pytan handler
32 handler_args = {}
33
34 # establish our connection info for the Tanium Server
35 handler_args['username'] = "Administrator"
36 handler_args['password'] = "Tanium2015!"
37 handler_args['host'] = "10.0.1.240"
38 handler_args['port'] = "443" # optional
39
40 # optional, level 0 is no output except warnings/errors
41 # level 1 through 12 are more and more verbose
42 handler_args['loglevel'] = 1
43
44 # optional, use a debug format for the logging output (uses two lines per log entry)
45 handler_args['debugformat'] = False
46
47 # optional, this saves all response objects to handler.session.ALL_REQUESTS_RESPONSES
48 # very useful for capturing the full exchange of XML requests and responses
49 handler_args['record_all_requests'] = True
50
51 # instantiate a handler using all of the arguments in the handler_args dictionary
52 print "...CALLING: pytan.handler() with args: {}".format(handler_args)
53 handler = pytan.Handler(**handler_args)
54
55 # print out the handler string
56 print "...OUTPUT: handler string: {}".format(handler)
57
58 # setup the arguments for the handler() class
59 kwargs = {}
60 kwargs["filters_help"] = True
61
62 print "...CALLING: handler.deploy_action() with args: {}".format(kwargs)
63 try:
64     handler.deploy_action(**kwargs)
65 except Exception as e:
66     print "...EXCEPTION: {}".format(e)
67     # this should throw an exception of type: pytan.exceptions.PytanHelp
68     # uncomment to see full exception
69     # traceback.print_exc(file=sys.stdout)

```

### Invalid Deploy Action Missing Parameters

Deploy an action using a package that requires parameters but do not supply any parameters.

- STDOUT from Example Python Code
- STDERR from Example Python Code
- Example Python Code

```

1 # import the basic python packages we need
2 import os
3 import sys
4 import tempfile

```

```
5 import pprint
6 import traceback
7
8 # disable python from generating a .pyc file
9 sys.dont_write_bytecode = True
10
11 # change me to the path of pytan if this script is not running from EXAMPLES/PYTAN_API
12 pytan_loc = "~/gh/pytan"
13 pytan_static_path = os.path.join(os.path.expanduser(pytan_loc), 'lib')
14
15 # Determine our script name, script dir
16 my_file = os.path.abspath(sys.argv[0])
17 my_dir = os.path.dirname(my_file)
18
19 # try to automatically determine the pytan lib directory by assuming it is in '../..lib/'
20 parent_dir = os.path.dirname(my_dir)
21 pytan_root_dir = os.path.dirname(parent_dir)
22 lib_dir = os.path.join(pytan_root_dir, 'lib')
23
24 # add pytan_loc and lib_dir to the PYTHONPATH variable
25 path_adds = [lib_dir, pytan_static_path]
26 [sys.path.append(aa) for aa in path_adds if aa not in sys.path]
27
28 # import pytan
29 import pytan
30
31 # create a dictionary of arguments for the pytan handler
32 handler_args = {}
33
34 # establish our connection info for the Tanium Server
35 handler_args['username'] = "Administrator"
36 handler_args['password'] = "Tanium2015!"
37 handler_args['host'] = "10.0.1.240"
38 handler_args['port'] = "443" # optional
39
40 # optional, level 0 is no output except warnings/errors
41 # level 1 through 12 are more and more verbose
42 handler_args['loglevel'] = 1
43
44 # optional, use a debug format for the logging output (uses two lines per log entry)
45 handler_args['debugformat'] = False
46
47 # optional, this saves all response objects to handler.session.ALL_REQUESTS_RESPONSES
48 # very useful for capturing the full exchange of XML requests and responses
49 handler_args['record_all_requests'] = True
50
51 # instantiate a handler using all of the arguments in the handler_args dictionary
52 print "...CALLING: pytan.handler() with args: {}".format(handler_args)
53 handler = pytan.Handler(**handler_args)
54
55 # print out the handler string
56 print "...OUTPUT: handler string: {}".format(handler)
57
58 # setup the arguments for the handler() class
59 kwargs = {}
60 kwargs["run"] = True
61 kwargs["package"] = u'Custom Tagging - Add Tags'
```

```

63 print "...CALLING: handler.deploy_action() with args: {}".format(kwargs)
64 try:
65     handler.deploy_action(**kwargs)
66 except Exception as e:
67     print "...EXCEPTION: {}".format(e)
68     # this should throw an exception of type: pytan.exceptions.HandlerError
69     # uncomment to see full exception
70     # traceback.print_exc(file=sys.stdout)

```

## PyTan API Invalid Export Basetype Examples

All of the PyTan API examples for Invalid Export Basetype

### Invalid Export Basetype CSV Bad Explode Type

Export a BaseType from getting objects using a bad explode\_json\_string\_values

- STDOUT from Example Python Code
- STDERR from Example Python Code
- Example Python Code

```

1  # import the basic python packages we need
2  import os
3  import sys
4  import tempfile
5  import pprint
6  import traceback
7
8  # disable python from generating a .pyc file
9  sys.dont_write_bytecode = True
10
11 # change me to the path of pytan if this script is not running from EXAMPLES/PYTAN_API
12 pytan_loc = "~/gh/pytan"
13 pytan_static_path = os.path.join(os.path.expanduser(pytan_loc), 'lib')
14
15 # Determine our script name, script dir
16 my_file = os.path.abspath(sys.argv[0])
17 my_dir = os.path.dirname(my_file)
18
19 # try to automatically determine the pytan lib directory by assuming it is in '../..lib/'
20 parent_dir = os.path.dirname(my_dir)
21 pytan_root_dir = os.path.dirname(parent_dir)
22 lib_dir = os.path.join(pytan_root_dir, 'lib')
23
24 # add pytan_loc and lib_dir to the PYTHONPATH variable
25 path_adds = [lib_dir, pytan_static_path]
26 [sys.path.append(aa) for aa in path_adds if aa not in sys.path]
27
28 # import pytan
29 import pytan
30
31 # create a dictionary of arguments for the pytan handler
32 handler_args = {}
33
34 # establish our connection info for the Tanium Server

```

```
35 handler_args['username'] = "Administrator"
36 handler_args['password'] = "Tanium2015!"
37 handler_args['host'] = "10.0.1.240"
38 handler_args['port'] = "443" # optional
39
40 # optional, level 0 is no output except warnings/errors
41 # level 1 through 12 are more and more verbose
42 handler_args['loglevel'] = 1
43
44 # optional, use a debug format for the logging output (uses two lines per log entry)
45 handler_args['debugformat'] = False
46
47 # optional, this saves all response objects to handler.session.ALL_REQUESTS_RESPONSES
48 # very useful for capturing the full exchange of XML requests and responses
49 handler_args['record_all_requests'] = True
50
51 # instantiate a handler using all of the arguments in the handler_args dictionary
52 print "...CALLING: pytan.handler() with args: {}".format(handler_args)
53 handler = pytan.Handler(**handler_args)
54
55 # print out the handler string
56 print "...OUTPUT: handler string: {}".format(handler)
57
58 # setup the arguments for the handler() class
59 kwargs = {}
60 kwargs["export_format"] = u'csv'
61 kwargs["explode_json_string_values"] = u'bad'
62
63 # setup the arguments for handler.get()
64 get_kwargs = {
65     'name': [
66         "Computer Name", "IP Route Details", "IP Address",
67         'Folder Name Search with RegEx Match',
68     ],
69     'objtype': 'sensor',
70 }
71
72 # get the objects that will provide the basetype that we want to use
73 print "...CALLING: handler.get() with args: {}".format(get_kwargs)
74 response = handler.get(**get_kwargs)
75
76 # store the basetype object as the obj we want to export
77 kwargs['obj'] = response
78
79 # export the object to a string
80 print "...CALLING: handler.export_obj() with args {}".format(kwargs)
81 try:
82     handler.export_obj(**kwargs)
83 except Exception as e:
84     print "...EXCEPTION: {}".format(e)
85     # this should throw an exception of type: pytan.exceptions.HandlerError
86     # uncomment to see full exception
87     # traceback.print_exc(file=sys.stdout)
```

## Invalid Export Basetype CSV Bad Sort Sub Type

Export a BaseType from getting objects using a bad header\_sort



- STDOUT from Example Python Code
- STDERR from Example Python Code
- Example Python Code

```

1  # import the basic python packages we need
2  import os
3  import sys
4  import tempfile
5  import pprint
6  import traceback
7
8  # disable python from generating a .pyc file
9  sys.dont_write_bytecode = True
10
11 # change me to the path of pytan if this script is not running from EXAMPLES/PYTAN_API
12 pytan_loc = "~/gh/pytan"
13 pytan_static_path = os.path.join(os.path.expanduser(pytan_loc), 'lib')
14
15 # Determine our script name, script dir
16 my_file = os.path.abspath(sys.argv[0])
17 my_dir = os.path.dirname(my_file)
18
19 # try to automatically determine the pytan lib directory by assuming it is in '../..lib/'
20 parent_dir = os.path.dirname(my_dir)
21 pytan_root_dir = os.path.dirname(parent_dir)
22 lib_dir = os.path.join(pytan_root_dir, 'lib')
23
24 # add pytan_loc and lib_dir to the PYTHONPATH variable
25 path_adds = [lib_dir, pytan_static_path]
26 [sys.path.append(aa) for aa in path_adds if aa not in sys.path]
27
28 # import pytan
29 import pytan
30
31 # create a dictionary of arguments for the pytan handler
32 handler_args = {}
33
34 # establish our connection info for the Tanium Server
35 handler_args['username'] = "Administrator"
36 handler_args['password'] = "Tanium2015!"
37 handler_args['host'] = "10.0.1.240"
38 handler_args['port'] = "443" # optional
39
40 # optional, level 0 is no output except warnings/errors
41 # level 1 through 12 are more and more verbose
42 handler_args['loglevel'] = 1
43
44 # optional, use a debug format for the logging output (uses two lines per log entry)
45 handler_args['debugformat'] = False
46
47 # optional, this saves all response objects to handler.session.ALL_REQUESTS_RESPONSES
48 # very useful for capturing the full exchange of XML requests and responses
49 handler_args['record_all_requests'] = True
50
51 # instantiate a handler using all of the arguments in the handler_args dictionary
52 print "...CALLING: pytan.handler() with args: {}".format(handler_args)
53 handler = pytan.Handler(**handler_args)
54

```

```
55 # print out the handler string
56 print "...OUTPUT: handler string: {}".format(handler)
57
58 # setup the arguments for the handler() class
59 kwargs = {}
60 kwargs["export_format"] = u'csv'
61 kwargs["header_sort"] = [[]]
62
63 # setup the arguments for handler.get()
64 get_kwargs = {
65     'name': [
66         "Computer Name", "IP Route Details", "IP Address",
67         'Folder Name Search with RegEx Match',
68     ],
69     'objtype': 'sensor',
70 }
71
72 # get the objects that will provide the basetype that we want to use
73 print "...CALLING: handler.get() with args: {}".format(get_kwargs)
74 response = handler.get(**get_kwargs)
75
76 # store the basetype object as the obj we want to export
77 kwargs['obj'] = response
78
79 # export the object to a string
80 print "...CALLING: handler.export_obj() with args {}".format(kwargs)
81 try:
82     handler.export_obj(**kwargs)
83 except Exception as e:
84     print "...EXCEPTION: {}".format(e)
85     # this should throw an exception of type: pytan.exceptions.HandlerError
86     # uncomment to see full exception
87     # traceback.print_exc(file=sys.stdout)
```

### Invalid Export Basetype CSV Bad Sort Type

Export a BaseType from getting objects using a bad header\_sort

- STDOUT from Example Python Code
- STDERR from Example Python Code
- Example Python Code

```
1 # import the basic python packages we need
2 import os
3 import sys
4 import tempfile
5 import pprint
6 import traceback
7
8 # disable python from generating a .pyc file
9 sys.dont_write_bytecode = True
10
11 # change me to the path of pytan if this script is not running from EXAMPLES/PYTAN_API
12 pytan_loc = "~/gh/pytan"
13 pytan_static_path = os.path.join(os.path.expanduser(pytan_loc), 'lib')
14
```

```

15 # Determine our script name, script dir
16 my_file = os.path.abspath(sys.argv[0])
17 my_dir = os.path.dirname(my_file)
18
19 # try to automatically determine the pytan lib directory by assuming it is in '../../lib/'
20 parent_dir = os.path.dirname(my_dir)
21 pytan_root_dir = os.path.dirname(parent_dir)
22 lib_dir = os.path.join(pytan_root_dir, 'lib')
23
24 # add pytan_loc and lib_dir to the PYTHONPATH variable
25 path_adds = [lib_dir, pytan_static_path]
26 [sys.path.append(aa) for aa in path_adds if aa not in sys.path]
27
28 # import pytan
29 import pytan
30
31 # create a dictionary of arguments for the pytan handler
32 handler_args = {}
33
34 # establish our connection info for the Tanium Server
35 handler_args['username'] = "Administrator"
36 handler_args['password'] = "Tanium2015!"
37 handler_args['host'] = "10.0.1.240"
38 handler_args['port'] = "443" # optional
39
40 # optional, level 0 is no output except warnings/errors
41 # level 1 through 12 are more and more verbose
42 handler_args['loglevel'] = 1
43
44 # optional, use a debug format for the logging output (uses two lines per log entry)
45 handler_args['debugformat'] = False
46
47 # optional, this saves all response objects to handler.session.ALL_REQUESTS_RESPONSES
48 # very useful for capturing the full exchange of XML requests and responses
49 handler_args['record_all_requests'] = True
50
51 # instantiate a handler using all of the arguments in the handler_args dictionary
52 print "...CALLING: pytan.handler() with args: {}".format(handler_args)
53 handler = pytan.Handler(**handler_args)
54
55 # print out the handler string
56 print "...OUTPUT: handler string: {}".format(handler)
57
58 # setup the arguments for the handler() class
59 kwargs = {}
60 kwargs["export_format"] = u'csv'
61 kwargs["header_sort"] = u'bad'
62
63 # setup the arguments for handler.get()
64 get_kwargs = {
65     'name': [
66         "Computer Name", "IP Route Details", "IP Address",
67         'Folder Name Search with RegEx Match',
68     ],
69     'objtype': 'sensor',
70 }
71
72 # get the objects that will provide the basetype that we want to use

```

```
73 print "...CALLING: handler.get() with args: {}".format(get_kwargs)
74 response = handler.get(**get_kwargs)
75
76 # store the basetype object as the obj we want to export
77 kwargs['obj'] = response
78
79 # export the object to a string
80 print "...CALLING: handler.export_obj() with args {}".format(kwargs)
81 try:
82     handler.export_obj(**kwargs)
83 except Exception as e:
84     print "...EXCEPTION: {}".format(e)
85     # this should throw an exception of type: pytan.exceptions.HandlerError
86     # uncomment to see full exception
87     # traceback.print_exc(file=sys.stdout)
```

### Invalid Export Basetype XML Bad Minimal Type

Export a BaseType from getting objects using a bad minimal

- STDOUT from Example Python Code
- STDERR from Example Python Code
- Example Python Code

```
1  # import the basic python packages we need
2  import os
3  import sys
4  import tempfile
5  import pprint
6  import traceback
7
8  # disable python from generating a .pyc file
9  sys.dont_write_bytecode = True
10
11 # change me to the path of pytan if this script is not running from EXAMPLES/PYTAN_API
12 pytan_loc = "~/gh/pytan"
13 pytan_static_path = os.path.join(os.path.expanduser(pytan_loc), 'lib')
14
15 # Determine our script name, script dir
16 my_file = os.path.abspath(sys.argv[0])
17 my_dir = os.path.dirname(my_file)
18
19 # try to automatically determine the pytan lib directory by assuming it is in '../..lib/'
20 parent_dir = os.path.dirname(my_dir)
21 pytan_root_dir = os.path.dirname(parent_dir)
22 lib_dir = os.path.join(pytan_root_dir, 'lib')
23
24 # add pytan_loc and lib_dir to the PYTHONPATH variable
25 path_adds = [lib_dir, pytan_static_path]
26 [sys.path.append(aa) for aa in path_adds if aa not in sys.path]
27
28 # import pytan
29 import pytan
30
31 # create a dictionary of arguments for the pytan handler
32 handler_args = {}
```

```

33
34 # establish our connection info for the Tanium Server
35 handler_args['username'] = "Administrator"
36 handler_args['password'] = "Tanium2015!"
37 handler_args['host'] = "10.0.1.240"
38 handler_args['port'] = "443" # optional
39
40 # optional, level 0 is no output except warnings/errors
41 # level 1 through 12 are more and more verbose
42 handler_args['loglevel'] = 1
43
44 # optional, use a debug format for the logging output (uses two lines per log entry)
45 handler_args['debugformat'] = False
46
47 # optional, this saves all response objects to handler.session.ALL_REQUESTS_RESPONSES
48 # very useful for capturing the full exchange of XML requests and responses
49 handler_args['record_all_requests'] = True
50
51 # instantiate a handler using all of the arguments in the handler_args dictionary
52 print "...CALLING: pytan.handler() with args: {}".format(handler_args)
53 handler = pytan.Handler(**handler_args)
54
55 # print out the handler string
56 print "...OUTPUT: handler string: {}".format(handler)
57
58 # setup the arguments for the handler() class
59 kwargs = {}
60 kwargs["export_format"] = u'xml'
61 kwargs["minimal"] = u'bad'
62
63 # setup the arguments for handler.get()
64 get_kwargs = {
65     'name': [
66         "Computer Name", "IP Route Details", "IP Address",
67         'Folder Name Search with RegEx Match',
68     ],
69     'objtype': 'sensor',
70 }
71
72 # get the objects that will provide the basetype that we want to use
73 print "...CALLING: handler.get() with args: {}".format(get_kwargs)
74 response = handler.get(**get_kwargs)
75
76 # store the basetype object as the obj we want to export
77 kwargs['obj'] = response
78
79 # export the object to a string
80 print "...CALLING: handler.export_obj() with args {}".format(kwargs)
81 try:
82     handler.export_obj(**kwargs)
83 except Exception as e:
84     print "...EXCEPTION: {}".format(e)
85     # this should throw an exception of type: pytan.exceptions.HandlerError
86     # uncomment to see full exception
87     # traceback.print_exc(file=sys.stdout)

```

## Invalid Export Basetype JSON Bad Include Type

Export a BaseType from getting objects using a bad include\_type

- STDOUT from Example Python Code
- STDERR from Example Python Code
- Example Python Code

```
1  # import the basic python packages we need
2  import os
3  import sys
4  import tempfile
5  import pprint
6  import traceback
7
8  # disable python from generating a .pyc file
9  sys.dont_write_bytecode = True
10
11  # change me to the path of pytan if this script is not running from EXAMPLES/PYTAN_API
12  pytan_loc = "~/gh/pytan"
13  pytan_static_path = os.path.join(os.path.expanduser(pytan_loc), 'lib')
14
15  # Determine our script name, script dir
16  my_file = os.path.abspath(sys.argv[0])
17  my_dir = os.path.dirname(my_file)
18
19  # try to automatically determine the pytan lib directory by assuming it is in '../..lib/'
20  parent_dir = os.path.dirname(my_dir)
21  pytan_root_dir = os.path.dirname(parent_dir)
22  lib_dir = os.path.join(pytan_root_dir, 'lib')
23
24  # add pytan_loc and lib_dir to the PYTHONPATH variable
25  path_adds = [lib_dir, pytan_static_path]
26  [sys.path.append(aa) for aa in path_adds if aa not in sys.path]
27
28  # import pytan
29  import pytan
30
31  # create a dictionary of arguments for the pytan handler
32  handler_args = {}
33
34  # establish our connection info for the Tanium Server
35  handler_args['username'] = "Administrator"
36  handler_args['password'] = "Tanium2015!"
37  handler_args['host'] = "10.0.1.240"
38  handler_args['port'] = "443" # optional
39
40  # optional, level 0 is no output except warnings/errors
41  # level 1 through 12 are more and more verbose
42  handler_args['loglevel'] = 1
43
44  # optional, use a debug format for the logging output (uses two lines per log entry)
45  handler_args['debugformat'] = False
46
47  # optional, this saves all response objects to handler.session.ALL_REQUESTS_RESPONSES
48  # very useful for capturing the full exchange of XML requests and responses
49  handler_args['record_all_requests'] = True
```

```

50
51 # instantiate a handler using all of the arguments in the handler_args dictionary
52 print "...CALLING: pytan.handler() with args: {}".format(handler_args)
53 handler = pytan.Handler(**handler_args)
54
55 # print out the handler string
56 print "...OUTPUT: handler string: {}".format(handler)
57
58 # setup the arguments for the handler() class
59 kwargs = {}
60 kwargs["export_format"] = u'json'
61 kwargs["include_type"] = u'bad'
62
63 # setup the arguments for handler.get()
64 get_kwargs = {
65     'name': [
66         "Computer Name", "IP Route Details", "IP Address",
67         'Folder Name Search with RegEx Match',
68     ],
69     'objtype': 'sensor',
70 }
71
72 # get the objects that will provide the basetype that we want to use
73 print "...CALLING: handler.get() with args: {}".format(get_kwargs)
74 response = handler.get(**get_kwargs)
75
76 # store the basetype object as the obj we want to export
77 kwargs['obj'] = response
78
79 # export the object to a string
80 print "...CALLING: handler.export_obj() with args {}".format(kwargs)
81 try:
82     handler.export_obj(**kwargs)
83 except Exception as e:
84     print "...EXCEPTION: {}".format(e)
85     # this should throw an exception of type: pytan.exceptions.HandlerError
86     # uncomment to see full exception
87     # traceback.print_exc(file=sys.stdout)

```

### Invalid Export Basetype JSON Bad Explode Type

Export a BaseType from getting objects using a bad explode\_json\_string\_values

- STDOUT from Example Python Code
- STDERR from Example Python Code
- Example Python Code

```

1 # import the basic python packages we need
2 import os
3 import sys
4 import tempfile
5 import pprint
6 import traceback
7
8 # disable python from generating a .pyc file
9 sys.dont_write_bytecode = True

```

```

10
11 # change me to the path of pytan if this script is not running from EXAMPLES/PYTAN_API
12 pytan_loc = "~/gh/pytan"
13 pytan_static_path = os.path.join(os.path.expanduser(pytan_loc), 'lib')
14
15 # Determine our script name, script dir
16 my_file = os.path.abspath(sys.argv[0])
17 my_dir = os.path.dirname(my_file)
18
19 # try to automatically determine the pytan lib directory by assuming it is in '../..lib/'
20 parent_dir = os.path.dirname(my_dir)
21 pytan_root_dir = os.path.dirname(parent_dir)
22 lib_dir = os.path.join(pytan_root_dir, 'lib')
23
24 # add pytan_loc and lib_dir to the PYTHONPATH variable
25 path_adds = [lib_dir, pytan_static_path]
26 [sys.path.append(aa) for aa in path_adds if aa not in sys.path]
27
28 # import pytan
29 import pytan
30
31 # create a dictionary of arguments for the pytan handler
32 handler_args = {}
33
34 # establish our connection info for the Tanium Server
35 handler_args['username'] = "Administrator"
36 handler_args['password'] = "Tanium2015!"
37 handler_args['host'] = "10.0.1.240"
38 handler_args['port'] = "443" # optional
39
40 # optional, level 0 is no output except warnings/errors
41 # level 1 through 12 are more and more verbose
42 handler_args['loglevel'] = 1
43
44 # optional, use a debug format for the logging output (uses two lines per log entry)
45 handler_args['debugformat'] = False
46
47 # optional, this saves all response objects to handler.session.ALL_REQUESTS_RESPONSES
48 # very useful for capturing the full exchange of XML requests and responses
49 handler_args['record_all_requests'] = True
50
51 # instantiate a handler using all of the arguments in the handler_args dictionary
52 print "...CALLING: pytan.handler() with args: {}".format(handler_args)
53 handler = pytan.Handler(**handler_args)
54
55 # print out the handler string
56 print "...OUTPUT: handler string: {}".format(handler)
57
58 # setup the arguments for the handler() class
59 kwargs = {}
60 kwargs["export_format"] = u'json'
61 kwargs["explode_json_string_values"] = u'bad'
62
63 # setup the arguments for handler.get()
64 get_kwargs = {
65     'name': [
66         "Computer Name", "IP Route Details", "IP Address",
67         'Folder Name Search with RegEx Match',

```



```

68     ],
69     'objtype': 'sensor',
70 }
71
72 # get the objects that will provide the basetype that we want to use
73 print "...CALLING: handler.get() with args: {}".format(get_kwargs)
74 response = handler.get(**get_kwargs)
75
76 # store the basetype object as the obj we want to export
77 kwargs['obj'] = response
78
79 # export the object to a string
80 print "...CALLING: handler.export_obj() with args {}".format(kwargs)
81 try:
82     handler.export_obj(**kwargs)
83 except Exception as e:
84     print "...EXCEPTION: {}".format(e)
85     # this should throw an exception of type: pytan.exceptions.HandlerError
86     # uncomment to see full exception
87     # traceback.print_exc(file=sys.stdout)

```

### Invalid Export Basetype Bad Format

Export a BaseType from getting objects using a bad export\_format

- STDOUT from Example Python Code
- STDERR from Example Python Code
- Example Python Code

```

1  # import the basic python packages we need
2  import os
3  import sys
4  import tempfile
5  import pprint
6  import traceback
7
8  # disable python from generating a .pyc file
9  sys.dont_write_bytecode = True
10
11 # change me to the path of pytan if this script is not running from EXAMPLES/PYTAN_API
12 pytan_loc = "~/gh/pytan"
13 pytan_static_path = os.path.join(os.path.expanduser(pytan_loc), 'lib')
14
15 # Determine our script name, script dir
16 my_file = os.path.abspath(sys.argv[0])
17 my_dir = os.path.dirname(my_file)
18
19 # try to automatically determine the pytan lib directory by assuming it is in '.././lib/'
20 parent_dir = os.path.dirname(my_dir)
21 pytan_root_dir = os.path.dirname(parent_dir)
22 lib_dir = os.path.join(pytan_root_dir, 'lib')
23
24 # add pytan_loc and lib_dir to the PYTHONPATH variable
25 path_adds = [lib_dir, pytan_static_path]
26 [sys.path.append(aa) for aa in path_adds if aa not in sys.path]
27

```

```
28 # import pytan
29 import pytan
30
31 # create a dictionary of arguments for the pytan handler
32 handler_args = {}
33
34 # establish our connection info for the Tanium Server
35 handler_args['username'] = "Administrator"
36 handler_args['password'] = "Tanium2015!"
37 handler_args['host'] = "10.0.1.240"
38 handler_args['port'] = "443" # optional
39
40 # optional, level 0 is no output except warnings/errors
41 # level 1 through 12 are more and more verbose
42 handler_args['loglevel'] = 1
43
44 # optional, use a debug format for the logging output (uses two lines per log entry)
45 handler_args['debugformat'] = False
46
47 # optional, this saves all response objects to handler.session.ALL_REQUESTS_RESPONSES
48 # very useful for capturing the full exchange of XML requests and responses
49 handler_args['record_all_requests'] = True
50
51 # instantiate a handler using all of the arguments in the handler_args dictionary
52 print "...CALLING: pytan.handler() with args: {}".format(handler_args)
53 handler = pytan.Handler(**handler_args)
54
55 # print out the handler string
56 print "...OUTPUT: handler string: {}".format(handler)
57
58 # setup the arguments for the handler() class
59 kwargs = {}
60 kwargs["export_format"] = u'bad'
61
62 # setup the arguments for handler.get()
63 get_kwargs = {
64     'name': [
65         "Computer Name", "IP Route Details", "IP Address",
66         'Folder Name Search with RegEx Match',
67     ],
68     'objtype': 'sensor',
69 }
70
71 # get the objects that will provide the basetype that we want to use
72 print "...CALLING: handler.get() with args: {}".format(get_kwargs)
73 response = handler.get(**get_kwargs)
74
75 # store the basetype object as the obj we want to export
76 kwargs['obj'] = response
77
78 # export the object to a string
79 print "...CALLING: handler.export_obj() with args {}".format(kwargs)
80 try:
81     handler.export_obj(**kwargs)
82 except Exception as e:
83     print "...EXCEPTION: {}".format(e)
84     # this should throw an exception of type: pytan.exceptions.HandlerError
85     # uncomment to see full exception
```

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

## PyTan API Invalid Export ResultSet Examples

All of the PyTan API examples for Invalid Export ResultSet

### Invalid Export ResultSet CSV Bad Sort Sub Type

Export a ResultSet from asking a question using a bad header\_sort

- STDOUT from Example Python Code
- STDERR from Example Python Code
- Example Python Code

```
1  # import the basic python packages we need
2  import os
3  import sys
4  import tempfile
5  import pprint
6  import traceback
7
8  # disable python from generating a .pyc file
9  sys.dont_write_bytecode = True
10
11  # change me to the path of pytan if this script is not running from EXAMPLES/PYTAN_API
12  pytan_loc = "~/gh/pytan"
13  pytan_static_path = os.path.join(os.path.expanduser(pytan_loc), 'lib')
14
15  # Determine our script name, script dir
16  my_file = os.path.abspath(sys.argv[0])
17  my_dir = os.path.dirname(my_file)
18
19  # try to automatically determine the pytan lib directory by assuming it is in '../..lib/'
20  parent_dir = os.path.dirname(my_dir)
21  pytan_root_dir = os.path.dirname(parent_dir)
22  lib_dir = os.path.join(pytan_root_dir, 'lib')
23
24  # add pytan_loc and lib_dir to the PYTHONPATH variable
25  path_adds = [lib_dir, pytan_static_path]
26  [sys.path.append(aa) for aa in path_adds if aa not in sys.path]
27
28  # import pytan
29  import pytan
30
31  # create a dictionary of arguments for the pytan handler
32  handler_args = {}
33
34  # establish our connection info for the Tanium Server
35  handler_args['username'] = "Administrator"
36  handler_args['password'] = "Tanium2015!"
37  handler_args['host'] = "10.0.1.240"
38  handler_args['port'] = "443" # optional
39
40  # optional, level 0 is no output except warnings/errors
41  # level 1 through 12 are more and more verbose
```

```
42 handler_args['loglevel'] = 1
43
44 # optional, use a debug format for the logging output (uses two lines per log entry)
45 handler_args['debugformat'] = False
46
47 # optional, this saves all response objects to handler.session.ALL_REQUESTS_RESPONSES
48 # very useful for capturing the full exchange of XML requests and responses
49 handler_args['record_all_requests'] = True
50
51 # instantiate a handler using all of the arguments in the handler_args dictionary
52 print "...CALLING: pytan.handler() with args: {}".format(handler_args)
53 handler = pytan.Handler(**handler_args)
54
55 # print out the handler string
56 print "...OUTPUT: handler string: {}".format(handler)
57
58 # setup the arguments for the handler() class
59 kwargs = {}
60 kwargs["export_format"] = u'csv'
61 kwargs["header_sort"] = [[]]
62
63 # setup the arguments for handler.ask()
64 ask_kwargs = {
65     'qtype': 'manual',
66     'sensors': [
67         "Computer Name"
68     ],
69 }
70
71 # ask the question that will provide the resultset that we want to use
72 print "...CALLING: handler.ask() with args {}".format(ask_kwargs)
73 response = handler.ask(**ask_kwargs)
74
75 # store the resultset object as the obj we want to export
76 kwargs['obj'] = response['question_results']
77
78 # export the object to a string
79 print "...CALLING: handler.export_obj() with args {}".format(kwargs)
80 try:
81     handler.export_obj(**kwargs)
82 except Exception as e:
83     print "...EXCEPTION: {}".format(e)
84     # this should throw an exception of type: pytan.exceptions.HandlerError
85     # uncomment to see full exception
86     # traceback.print_exc(file=sys.stdout)
```

### Invalid Export ResultSet CSV Bad Sort Type

Export a ResultSet from asking a question using a bad header\_sort

- STDOUT from Example Python Code
- STDERR from Example Python Code
- Example Python Code

```
1 # import the basic python packages we need
2 import os
```

```

3 import sys
4 import tempfile
5 import pprint
6 import traceback
7
8 # disable python from generating a .pyc file
9 sys.dont_write_bytecode = True
10
11 # change me to the path of pytan if this script is not running from EXAMPLES/PYTAN_API
12 pytan_loc = "~/gh/pytan"
13 pytan_static_path = os.path.join(os.path.expanduser(pytan_loc), 'lib')
14
15 # Determine our script name, script dir
16 my_file = os.path.abspath(sys.argv[0])
17 my_dir = os.path.dirname(my_file)
18
19 # try to automatically determine the pytan lib directory by assuming it is in '../..lib/'
20 parent_dir = os.path.dirname(my_dir)
21 pytan_root_dir = os.path.dirname(parent_dir)
22 lib_dir = os.path.join(pytan_root_dir, 'lib')
23
24 # add pytan_loc and lib_dir to the PYTHONPATH variable
25 path_adds = [lib_dir, pytan_static_path]
26 [sys.path.append(aa) for aa in path_adds if aa not in sys.path]
27
28 # import pytan
29 import pytan
30
31 # create a dictionary of arguments for the pytan handler
32 handler_args = {}
33
34 # establish our connection info for the Tanium Server
35 handler_args['username'] = "Administrator"
36 handler_args['password'] = "Tanium2015!"
37 handler_args['host'] = "10.0.1.240"
38 handler_args['port'] = "443" # optional
39
40 # optional, level 0 is no output except warnings/errors
41 # level 1 through 12 are more and more verbose
42 handler_args['loglevel'] = 1
43
44 # optional, use a debug format for the logging output (uses two lines per log entry)
45 handler_args['debugformat'] = False
46
47 # optional, this saves all response objects to handler.session.ALL_REQUESTS_RESPONSES
48 # very useful for capturing the full exchange of XML requests and responses
49 handler_args['record_all_requests'] = True
50
51 # instantiate a handler using all of the arguments in the handler_args dictionary
52 print "...CALLING: pytan.handler() with args: {}".format(handler_args)
53 handler = pytan.Handler(**handler_args)
54
55 # print out the handler string
56 print "...OUTPUT: handler string: {}".format(handler)
57
58 # setup the arguments for the handler() class
59 kwargs = {}
60 kwargs["export_format"] = u'csv'

```

```
61 kwargs["header_sort"] = u'bad'
62
63 # setup the arguments for handler.ask()
64 ask_kwargs = {
65     'qtype': 'manual',
66     'sensors': [
67         "Computer Name"
68     ],
69 }
70
71 # ask the question that will provide the resultset that we want to use
72 print "...CALLING: handler.ask() with args {}".format(ask_kwargs)
73 response = handler.ask(**ask_kwargs)
74
75 # store the resultset object as the obj we want to export
76 kwargs['obj'] = response['question_results']
77
78 # export the object to a string
79 print "...CALLING: handler.export_obj() with args {}".format(kwargs)
80 try:
81     handler.export_obj(**kwargs)
82 except Exception as e:
83     print "...EXCEPTION: {}".format(e)
84     # this should throw an exception of type: pytan.exceptions.HandlerError
85     # uncomment to see full exception
86     # traceback.print_exc(file=sys.stdout)
```

### Invalid Export ResultSet CSV Bad Expand Type

Export a ResultSet from asking a question using a bad `expand_grouped_columns`

- STDOUT from Example Python Code
- STDERR from Example Python Code
- Example Python Code

```
1 # import the basic python packages we need
2 import os
3 import sys
4 import tempfile
5 import pprint
6 import traceback
7
8 # disable python from generating a .pyc file
9 sys.dont_write_bytecode = True
10
11 # change me to the path of pytan if this script is not running from EXAMPLES/PYTAN_API
12 pytan_loc = "~/gh/pytan"
13 pytan_static_path = os.path.join(os.path.expanduser(pytan_loc), 'lib')
14
15 # Determine our script name, script dir
16 my_file = os.path.abspath(sys.argv[0])
17 my_dir = os.path.dirname(my_file)
18
19 # try to automatically determine the pytan lib directory by assuming it is in '../..lib/'
20 parent_dir = os.path.dirname(my_dir)
21 pytan_root_dir = os.path.dirname(parent_dir)
```

```

22 lib_dir = os.path.join(pytan_root_dir, 'lib')
23
24 # add pytan_loc and lib_dir to the PYTHONPATH variable
25 path_adds = [lib_dir, pytan_static_path]
26 [sys.path.append(aa) for aa in path_adds if aa not in sys.path]
27
28 # import pytan
29 import pytan
30
31 # create a dictionary of arguments for the pytan handler
32 handler_args = {}
33
34 # establish our connection info for the Tanium Server
35 handler_args['username'] = "Administrator"
36 handler_args['password'] = "Tanium2015!"
37 handler_args['host'] = "10.0.1.240"
38 handler_args['port'] = "443" # optional
39
40 # optional, level 0 is no output except warnings/errors
41 # level 1 through 12 are more and more verbose
42 handler_args['loglevel'] = 1
43
44 # optional, use a debug format for the logging output (uses two lines per log entry)
45 handler_args['debugformat'] = False
46
47 # optional, this saves all response objects to handler.session.ALL_REQUESTS_RESPONSES
48 # very useful for capturing the full exchange of XML requests and responses
49 handler_args['record_all_requests'] = True
50
51 # instantiate a handler using all of the arguments in the handler_args dictionary
52 print "...CALLING: pytan.handler() with args: {}".format(handler_args)
53 handler = pytan.Handler(**handler_args)
54
55 # print out the handler string
56 print "...OUTPUT: handler string: {}".format(handler)
57
58 # setup the arguments for the handler() class
59 kwargs = {}
60 kwargs["export_format"] = u'csv'
61 kwargs["expand_grouped_columns"] = u'bad'
62
63 # setup the arguments for handler.ask()
64 ask_kwargs = {
65     'qtype': 'manual',
66     'sensors': [
67         "Computer Name"
68     ],
69 }
70
71 # ask the question that will provide the resultset that we want to use
72 print "...CALLING: handler.ask() with args {}".format(ask_kwargs)
73 response = handler.ask(**ask_kwargs)
74
75 # store the resultset object as the obj we want to export
76 kwargs['obj'] = response['question_results']
77
78 # export the object to a string
79 print "...CALLING: handler.export_obj() with args {}".format(kwargs)

```

```
80 try:
81     handler.export_obj(**kwargs)
82 except Exception as e:
83     print "...EXCEPTION: {}".format(e)
84     # this should throw an exception of type: pytan.exceptions.HandlerError
85     # uncomment to see full exception
86     # traceback.print_exc(file=sys.stdout)
```

### Invalid Export ResultSet CSV Bad Sensors Sub Type

Export a ResultSet from asking a question using a bad sensors

- STDOUT from Example Python Code
- STDERR from Example Python Code
- Example Python Code

```
1  # import the basic python packages we need
2  import os
3  import sys
4  import tempfile
5  import pprint
6  import traceback
7
8  # disable python from generating a .pyc file
9  sys.dont_write_bytecode = True
10
11 # change me to the path of pytan if this script is not running from EXAMPLES/PYTAN_API
12 pytan_loc = "~/gh/pytan"
13 pytan_static_path = os.path.join(os.path.expanduser(pytan_loc), 'lib')
14
15 # Determine our script name, script dir
16 my_file = os.path.abspath(sys.argv[0])
17 my_dir = os.path.dirname(my_file)
18
19 # try to automatically determine the pytan lib directory by assuming it is in '../..lib/'
20 parent_dir = os.path.dirname(my_dir)
21 pytan_root_dir = os.path.dirname(parent_dir)
22 lib_dir = os.path.join(pytan_root_dir, 'lib')
23
24 # add pytan_loc and lib_dir to the PYTHONPATH variable
25 path_adds = [lib_dir, pytan_static_path]
26 [sys.path.append(aa) for aa in path_adds if aa not in sys.path]
27
28 # import pytan
29 import pytan
30
31 # create a dictionary of arguments for the pytan handler
32 handler_args = {}
33
34 # establish our connection info for the Tanium Server
35 handler_args['username'] = "Administrator"
36 handler_args['password'] = "Tanium2015!"
37 handler_args['host'] = "10.0.1.240"
38 handler_args['port'] = "443" # optional
39
40 # optional, level 0 is no output except warnings/errors
```



```

41 # level 1 through 12 are more and more verbose
42 handler_args['loglevel'] = 1
43
44 # optional, use a debug format for the logging output (uses two lines per log entry)
45 handler_args['debugformat'] = False
46
47 # optional, this saves all response objects to handler.session.ALL_REQUESTS_RESPONSES
48 # very useful for capturing the full exchange of XML requests and responses
49 handler_args['record_all_requests'] = True
50
51 # instantiate a handler using all of the arguments in the handler_args dictionary
52 print "...CALLING: pytan.handler() with args: {}".format(handler_args)
53 handler = pytan.Handler(**handler_args)
54
55 # print out the handler string
56 print "...OUTPUT: handler string: {}".format(handler)
57
58 # setup the arguments for the handler() class
59 kwargs = {}
60 kwargs["export_format"] = u'csv'
61 kwargs["sensors"] = [[]]
62 kwargs["header_add_sensor"] = True
63
64 # setup the arguments for handler.ask()
65 ask_kwargs = {
66     'qtype': 'manual',
67     'sensors': [
68         "Computer Name"
69     ],
70 }
71
72 # ask the question that will provide the resultset that we want to use
73 print "...CALLING: handler.ask() with args {}".format(ask_kwargs)
74 response = handler.ask(**ask_kwargs)
75
76 # store the resultset object as the obj we want to export
77 kwargs['obj'] = response['question_results']
78
79 # export the object to a string
80 print "...CALLING: handler.export_obj() with args {}".format(kwargs)
81 try:
82     handler.export_obj(**kwargs)
83 except Exception as e:
84     print "...EXCEPTION: {}".format(e)
85     # this should throw an exception of type: pytan.exceptions.HandlerError
86     # uncomment to see full exception
87     # traceback.print_exc(file=sys.stdout)

```

### Invalid Export ResultSet Bad Format

Export a ResultSet from asking a question using a bad export\_format

- STDOUT from Example Python Code
- STDERR from Example Python Code
- Example Python Code

```
1  # import the basic python packages we need
2  import os
3  import sys
4  import tempfile
5  import pprint
6  import traceback
7
8  # disable python from generating a .pyc file
9  sys.dont_write_bytecode = True
10
11  # change me to the path of pytan if this script is not running from EXAMPLES/PYTAN_API
12  pytan_loc = "~/gh/pytan"
13  pytan_static_path = os.path.join(os.path.expanduser(pytan_loc), 'lib')
14
15  # Determine our script name, script dir
16  my_file = os.path.abspath(sys.argv[0])
17  my_dir = os.path.dirname(my_file)
18
19  # try to automatically determine the pytan lib directory by assuming it is in '../..lib/'
20  parent_dir = os.path.dirname(my_dir)
21  pytan_root_dir = os.path.dirname(parent_dir)
22  lib_dir = os.path.join(pytan_root_dir, 'lib')
23
24  # add pytan_loc and lib_dir to the PYTHONPATH variable
25  path_adds = [lib_dir, pytan_static_path]
26  [sys.path.append(aa) for aa in path_adds if aa not in sys.path]
27
28  # import pytan
29  import pytan
30
31  # create a dictionary of arguments for the pytan handler
32  handler_args = {}
33
34  # establish our connection info for the Tanium Server
35  handler_args['username'] = "Administrator"
36  handler_args['password'] = "Tanium2015!"
37  handler_args['host'] = "10.0.1.240"
38  handler_args['port'] = "443" # optional
39
40  # optional, level 0 is no output except warnings/errors
41  # level 1 through 12 are more and more verbose
42  handler_args['loglevel'] = 1
43
44  # optional, use a debug format for the logging output (uses two lines per log entry)
45  handler_args['debugformat'] = False
46
47  # optional, this saves all response objects to handler.session.ALL_REQUESTS_RESPONSES
48  # very useful for capturing the full exchange of XML requests and responses
49  handler_args['record_all_requests'] = True
50
51  # instantiate a handler using all of the arguments in the handler_args dictionary
52  print "...CALLING: pytan.handler() with args: {}".format(handler_args)
53  handler = pytan.Handler(**handler_args)
54
55  # print out the handler string
56  print "...OUTPUT: handler string: {}".format(handler)
57
58  # setup the arguments for the handler() class
```

```

59 kwargs = {}
60 kwargs["export_format"] = u'bad'
61
62 # setup the arguments for handler.ask()
63 ask_kwargs = {
64     'qtype': 'manual',
65     'sensors': [
66         "Computer Name"
67     ],
68 }
69
70 # ask the question that will provide the resultset that we want to use
71 print "...CALLING: handler.ask() with args {}".format(ask_kwargs)
72 response = handler.ask(**ask_kwargs)
73
74 # store the resultset object as the obj we want to export
75 kwargs['obj'] = response['question_results']
76
77 # export the object to a string
78 print "...CALLING: handler.export_obj() with args {}".format(kwargs)
79 try:
80     handler.export_obj(**kwargs)
81 except Exception as e:
82     print "...EXCEPTION: {}".format(e)
83     # this should throw an exception of type: pytan.exceptions.HandlerError
84     # uncomment to see full exception
85     # traceback.print_exc(file=sys.stdout)

```

## PyTan API Invalid Get Object Examples

All of the PyTan API examples for Invalid Get Object

### Invalid Get Action Single By Name

Get an action by name (name is not a supported selector for action)

- STDOUT from Example Python Code
- STDERR from Example Python Code
- Example Python Code

```

1  # import the basic python packages we need
2  import os
3  import sys
4  import tempfile
5  import pprint
6  import traceback
7
8  # disable python from generating a .pyc file
9  sys.dont_write_bytecode = True
10
11 # change me to the path of pytan if this script is not running from EXAMPLES/PYTAN_API
12 pytan_loc = "~/gh/pytan"
13 pytan_static_path = os.path.join(os.path.expanduser(pytan_loc), 'lib')
14
15 # Determine our script name, script dir

```

```
16 my_file = os.path.abspath(sys.argv[0])
17 my_dir = os.path.dirname(my_file)
18
19 # try to automatically determine the pytan lib directory by assuming it is in '../..lib/'
20 parent_dir = os.path.dirname(my_dir)
21 pytan_root_dir = os.path.dirname(parent_dir)
22 lib_dir = os.path.join(pytan_root_dir, 'lib')
23
24 # add pytan_loc and lib_dir to the PYTHONPATH variable
25 path_adds = [lib_dir, pytan_static_path]
26 [sys.path.append(aa) for aa in path_adds if aa not in sys.path]
27
28 # import pytan
29 import pytan
30
31 # create a dictionary of arguments for the pytan handler
32 handler_args = {}
33
34 # establish our connection info for the Tanium Server
35 handler_args['username'] = "Administrator"
36 handler_args['password'] = "Tanium2015!"
37 handler_args['host'] = "10.0.1.240"
38 handler_args['port'] = "443" # optional
39
40 # optional, level 0 is no output except warnings/errors
41 # level 1 through 12 are more and more verbose
42 handler_args['loglevel'] = 1
43
44 # optional, use a debug format for the logging output (uses two lines per log entry)
45 handler_args['debugformat'] = False
46
47 # optional, this saves all response objects to handler.session.ALL_REQUESTS_RESPONSES
48 # very useful for capturing the full exchange of XML requests and responses
49 handler_args['record_all_requests'] = True
50
51 # instantiate a handler using all of the arguments in the handler_args dictionary
52 print "...CALLING: pytan.handler() with args: {}".format(handler_args)
53 handler = pytan.Handler(**handler_args)
54
55 # print out the handler string
56 print "...OUTPUT: handler string: {}".format(handler)
57
58 # setup the arguments for the handler() class
59 kwargs = {}
60 kwargs["objtype"] = u'action'
61 kwargs["name"] = u'Distribute Tanium Standard Utilities'
62
63 print "...CALLING: handler.get() with args: {}".format(kwargs)
64 try:
65     handler.get(**kwargs)
66 except Exception as e:
67     print "...EXCEPTION: {}".format(e)
68     # this should throw an exception of type: pytan.exceptions.HandlerError
69     # uncomment to see full exception
70     # traceback.print_exc(file=sys.stdout)
```

## Invalid Get Question By Name

Get a question by name (name is not a supported selector for question)

- STDOUT from Example Python Code
- STDERR from Example Python Code
- Example Python Code

```

1  # import the basic python packages we need
2  import os
3  import sys
4  import tempfile
5  import pprint
6  import traceback
7
8  # disable python from generating a .pyc file
9  sys.dont_write_bytecode = True
10
11 # change me to the path of pytan if this script is not running from EXAMPLES/PYTAN_API
12 pytan_loc = "~/gh/pytan"
13 pytan_static_path = os.path.join(os.path.expanduser(pytan_loc), 'lib')
14
15 # Determine our script name, script dir
16 my_file = os.path.abspath(sys.argv[0])
17 my_dir = os.path.dirname(my_file)
18
19 # try to automatically determine the pytan lib directory by assuming it is in '../..lib/'
20 parent_dir = os.path.dirname(my_dir)
21 pytan_root_dir = os.path.dirname(parent_dir)
22 lib_dir = os.path.join(pytan_root_dir, 'lib')
23
24 # add pytan_loc and lib_dir to the PYTHONPATH variable
25 path_adds = [lib_dir, pytan_static_path]
26 [sys.path.append(aa) for aa in path_adds if aa not in sys.path]
27
28 # import pytan
29 import pytan
30
31 # create a dictionary of arguments for the pytan handler
32 handler_args = {}
33
34 # establish our connection info for the Tanium Server
35 handler_args['username'] = "Administrator"
36 handler_args['password'] = "Tanium2015!"
37 handler_args['host'] = "10.0.1.240"
38 handler_args['port'] = "443" # optional
39
40 # optional, level 0 is no output except warnings/errors
41 # level 1 through 12 are more and more verbose
42 handler_args['loglevel'] = 1
43
44 # optional, use a debug format for the logging output (uses two lines per log entry)
45 handler_args['debugformat'] = False
46
47 # optional, this saves all response objects to handler.session.ALL_REQUESTS_RESPONSES
48 # very useful for capturing the full exchange of XML requests and responses
49 handler_args['record_all_requests'] = True

```

```
50
51 # instantiate a handler using all of the arguments in the handler_args dictionary
52 print "...CALLING: pytan.handler() with args: {}".format(handler_args)
53 handler = pytan.Handler(**handler_args)
54
55 # print out the handler string
56 print "...OUTPUT: handler string: {}".format(handler)
57
58 # setup the arguments for the handler() class
59 kwargs = {}
60 kwargs["objtype"] = u'question'
61 kwargs["name"] = u'dweedle'
62
63 print "...CALLING: handler.get() with args: {}".format(kwargs)
64 try:
65     handler.get(**kwargs)
66 except Exception as e:
67     print "...EXCEPTION: {}".format(e)
68     # this should throw an exception of type: pytan.exceptions.HandlerError
69     # uncomment to see full exception
70     # traceback.print_exc(file=sys.stdout)
```

## PyTan API Invalid Questions Examples

All of the PyTan API examples for Invalid Questions

### Invalid Ask Manual Question Sensor Help

Have ask\_manual() return the help for sensors

- STDOUT from Example Python Code
- STDERR from Example Python Code
- Example Python Code

```
1 # import the basic python packages we need
2 import os
3 import sys
4 import tempfile
5 import pprint
6 import traceback
7
8 # disable python from generating a .pyc file
9 sys.dont_write_bytecode = True
10
11 # change me to the path of pytan if this script is not running from EXAMPLES/PYTAN_API
12 pytan_loc = "~/gh/pytan"
13 pytan_static_path = os.path.join(os.path.expanduser(pytan_loc), 'lib')
14
15 # Determine our script name, script dir
16 my_file = os.path.abspath(sys.argv[0])
17 my_dir = os.path.dirname(my_file)
18
19 # try to automatically determine the pytan lib directory by assuming it is in '../..lib/'
20 parent_dir = os.path.dirname(my_dir)
21 pytan_root_dir = os.path.dirname(parent_dir)
```

```

22 lib_dir = os.path.join(pytan_root_dir, 'lib')
23
24 # add pytan_loc and lib_dir to the PYTHONPATH variable
25 path_adds = [lib_dir, pytan_static_path]
26 [sys.path.append(aa) for aa in path_adds if aa not in sys.path]
27
28 # import pytan
29 import pytan
30
31 # create a dictionary of arguments for the pytan handler
32 handler_args = {}
33
34 # establish our connection info for the Tanium Server
35 handler_args['username'] = "Administrator"
36 handler_args['password'] = "Tanium2015!"
37 handler_args['host'] = "10.0.1.240"
38 handler_args['port'] = "443" # optional
39
40 # optional, level 0 is no output except warnings/errors
41 # level 1 through 12 are more and more verbose
42 handler_args['loglevel'] = 1
43
44 # optional, use a debug format for the logging output (uses two lines per log entry)
45 handler_args['debugformat'] = False
46
47 # optional, this saves all response objects to handler.session.ALL_REQUESTS_RESPONSES
48 # very useful for capturing the full exchange of XML requests and responses
49 handler_args['record_all_requests'] = True
50
51 # instantiate a handler using all of the arguments in the handler_args dictionary
52 print "...CALLING: pytan.handler() with args: {}".format(handler_args)
53 handler = pytan.Handler(**handler_args)
54
55 # print out the handler string
56 print "...OUTPUT: handler string: {}".format(handler)
57
58 # setup the arguments for the handler() class
59 kwargs = {}
60 kwargs["qtype"] = u'manual'
61 kwargs["sensors_help"] = True
62
63 print "...CALLING: handler.ask() with args: {}".format(kwargs)
64 try:
65     handler.ask(**kwargs)
66 except Exception as e:
67     print "...EXCEPTION: {}".format(e)
68     # this should throw an exception of type: pytan.exceptions.PytanHelp
69     # uncomment to see full exception
70     # traceback.print_exc(file=sys.stdout)

```

### Invalid Ask Manual Question Filter Help

Have ask\_manual() return the help for filters

- STDOUT from Example Python Code
- STDERR from Example Python Code

- Example Python Code

```
1  # import the basic python packages we need
2  import os
3  import sys
4  import tempfile
5  import pprint
6  import traceback
7
8  # disable python from generating a .pyc file
9  sys.dont_write_bytecode = True
10
11  # change me to the path of pytan if this script is not running from EXAMPLES/PYTAN_API
12  pytan_loc = "~/gh/pytan"
13  pytan_static_path = os.path.join(os.path.expanduser(pytan_loc), 'lib')
14
15  # Determine our script name, script dir
16  my_file = os.path.abspath(sys.argv[0])
17  my_dir = os.path.dirname(my_file)
18
19  # try to automatically determine the pytan lib directory by assuming it is in '../..lib/'
20  parent_dir = os.path.dirname(my_dir)
21  pytan_root_dir = os.path.dirname(parent_dir)
22  lib_dir = os.path.join(pytan_root_dir, 'lib')
23
24  # add pytan_loc and lib_dir to the PYTHONPATH variable
25  path_adds = [lib_dir, pytan_static_path]
26  [sys.path.append(aa) for aa in path_adds if aa not in sys.path]
27
28  # import pytan
29  import pytan
30
31  # create a dictionary of arguments for the pytan handler
32  handler_args = {}
33
34  # establish our connection info for the Tanium Server
35  handler_args['username'] = "Administrator"
36  handler_args['password'] = "Tanium2015!"
37  handler_args['host'] = "10.0.1.240"
38  handler_args['port'] = "443" # optional
39
40  # optional, level 0 is no output except warnings/errors
41  # level 1 through 12 are more and more verbose
42  handler_args['loglevel'] = 1
43
44  # optional, use a debug format for the logging output (uses two lines per log entry)
45  handler_args['debugformat'] = False
46
47  # optional, this saves all response objects to handler.session.ALL_REQUESTS_RESPONSES
48  # very useful for capturing the full exchange of XML requests and responses
49  handler_args['record_all_requests'] = True
50
51  # instantiate a handler using all of the arguments in the handler_args dictionary
52  print "...CALLING: pytan.handler() with args: {}".format(handler_args)
53  handler = pytan.Handler(**handler_args)
54
55  # print out the handler string
56  print "...OUTPUT: handler string: {}".format(handler)
57
```



```

58 # setup the arguments for the handler() class
59 kwargs = {}
60 kwargs["filters_help"] = True
61 kwargs["qtype"] = u'manual'
62
63 print "...CALLING: handler.ask() with args: {}".format(kwargs)
64 try:
65     handler.ask(**kwargs)
66 except Exception as e:
67     print "...EXCEPTION: {}".format(e)
68     # this should throw an exception of type: pytan.exceptions.PytanHelp
69     # uncomment to see full exception
70     # traceback.print_exc(file=sys.stdout)

```

### Invalid Ask Manual Question Option Help

Have ask\_manual() return the help for options

- STDOUT from Example Python Code
- STDERR from Example Python Code
- Example Python Code

```

1  # import the basic python packages we need
2  import os
3  import sys
4  import tempfile
5  import pprint
6  import traceback
7
8  # disable python from generating a .pyc file
9  sys.dont_write_bytecode = True
10
11 # change me to the path of pytan if this script is not running from EXAMPLES/PYTAN_API
12 pytan_loc = "~/gh/pytan"
13 pytan_static_path = os.path.join(os.path.expanduser(pytan_loc), 'lib')
14
15 # Determine our script name, script dir
16 my_file = os.path.abspath(sys.argv[0])
17 my_dir = os.path.dirname(my_file)
18
19 # try to automatically determine the pytan lib directory by assuming it is in '../..lib/'
20 parent_dir = os.path.dirname(my_dir)
21 pytan_root_dir = os.path.dirname(parent_dir)
22 lib_dir = os.path.join(pytan_root_dir, 'lib')
23
24 # add pytan_loc and lib_dir to the PYTHONPATH variable
25 path_adds = [lib_dir, pytan_static_path]
26 [sys.path.append(aa) for aa in path_adds if aa not in sys.path]
27
28 # import pytan
29 import pytan
30
31 # create a dictionary of arguments for the pytan handler
32 handler_args = {}
33
34 # establish our connection info for the Tanium Server

```

```
35 handler_args['username'] = "Administrator"
36 handler_args['password'] = "Tanium2015!"
37 handler_args['host'] = "10.0.1.240"
38 handler_args['port'] = "443" # optional
39
40 # optional, level 0 is no output except warnings/errors
41 # level 1 through 12 are more and more verbose
42 handler_args['loglevel'] = 1
43
44 # optional, use a debug format for the logging output (uses two lines per log entry)
45 handler_args['debugformat'] = False
46
47 # optional, this saves all response objects to handler.session.ALL_REQUESTS_RESPONSES
48 # very useful for capturing the full exchange of XML requests and responses
49 handler_args['record_all_requests'] = True
50
51 # instantiate a handler using all of the arguments in the handler_args dictionary
52 print "...CALLING: pytan.handler() with args: {}".format(handler_args)
53 handler = pytan.Handler(**handler_args)
54
55 # print out the handler string
56 print "...OUTPUT: handler string: {}".format(handler)
57
58 # setup the arguments for the handler() class
59 kwargs = {}
60 kwargs["options_help"] = True
61 kwargs["qtype"] = u'manual'
62
63 print "...CALLING: handler.ask() with args: {}".format(kwargs)
64 try:
65     handler.ask(**kwargs)
66 except Exception as e:
67     print "...EXCEPTION: {}".format(e)
68     # this should throw an exception of type: pytan.exceptions.PytanHelp
69     # uncomment to see full exception
70     # traceback.print_exc(file=sys.stdout)
```

### Invalid Ask Manual Question Bad Filter

Ask a question using an invalid filter.

- STDOUT from Example Python Code
- STDERR from Example Python Code
- Example Python Code

```
1 # import the basic python packages we need
2 import os
3 import sys
4 import tempfile
5 import pprint
6 import traceback
7
8 # disable python from generating a .pyc file
9 sys.dont_write_bytecode = True
10
11 # change me to the path of pytan if this script is not running from EXAMPLES/PYTAN_API
```

```

12 pytan_loc = "~/gh/pytan"
13 pytan_static_path = os.path.join(os.path.expanduser(pytan_loc), 'lib')
14
15 # Determine our script name, script dir
16 my_file = os.path.abspath(sys.argv[0])
17 my_dir = os.path.dirname(my_file)
18
19 # try to automatically determine the pytan lib directory by assuming it is in '../..lib/'
20 parent_dir = os.path.dirname(my_dir)
21 pytan_root_dir = os.path.dirname(parent_dir)
22 lib_dir = os.path.join(pytan_root_dir, 'lib')
23
24 # add pytan_loc and lib_dir to the PYTHONPATH variable
25 path_adds = [lib_dir, pytan_static_path]
26 [sys.path.append(aa) for aa in path_adds if aa not in sys.path]
27
28 # import pytan
29 import pytan
30
31 # create a dictionary of arguments for the pytan handler
32 handler_args = {}
33
34 # establish our connection info for the Tanium Server
35 handler_args['username'] = "Administrator"
36 handler_args['password'] = "Tanium2015!"
37 handler_args['host'] = "10.0.1.240"
38 handler_args['port'] = "443" # optional
39
40 # optional, level 0 is no output except warnings/errors
41 # level 1 through 12 are more and more verbose
42 handler_args['loglevel'] = 1
43
44 # optional, use a debug format for the logging output (uses two lines per log entry)
45 handler_args['debugformat'] = False
46
47 # optional, this saves all response objects to handler.session.ALL_REQUESTS_RESPONSES
48 # very useful for capturing the full exchange of XML requests and responses
49 handler_args['record_all_requests'] = True
50
51 # instantiate a handler using all of the arguments in the handler_args dictionary
52 print "...CALLING: pytan.handler() with args: {}".format(handler_args)
53 handler = pytan.Handler(**handler_args)
54
55 # print out the handler string
56 print "...OUTPUT: handler string: {}".format(handler)
57
58 # setup the arguments for the handler() class
59 kwargs = {}
60 kwargs["sensors"] = u'Computer name, that does not meet:little'
61 kwargs["qtype"] = u'manual'
62
63 print "...CALLING: handler.ask() with args: {}".format(kwargs)
64 try:
65     handler.ask(**kwargs)
66 except Exception as e:
67     print "...EXCEPTION: {}".format(e)
68     # this should throw an exception of type: pytan.exceptions.HumanParserError
69     # uncomment to see full exception

```

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

### Invalid Ask Manual Question Bad Sensorname

Ask a question using a sensor that does not exist

- STDOUT from Example Python Code
- STDERR from Example Python Code
- Example Python Code

```
1  # import the basic python packages we need
2  import os
3  import sys
4  import tempfile
5  import pprint
6  import traceback
7
8  # disable python from generating a .pyc file
9  sys.dont_write_bytecode = True
10
11 # change me to the path of pytan if this script is not running from EXAMPLES/PYTAN_API
12 pytan_loc = "~/gh/pytan"
13 pytan_static_path = os.path.join(os.path.expanduser(pytan_loc), 'lib')
14
15 # Determine our script name, script dir
16 my_file = os.path.abspath(sys.argv[0])
17 my_dir = os.path.dirname(my_file)
18
19 # try to automatically determine the pytan lib directory by assuming it is in '../..lib/'
20 parent_dir = os.path.dirname(my_dir)
21 pytan_root_dir = os.path.dirname(parent_dir)
22 lib_dir = os.path.join(pytan_root_dir, 'lib')
23
24 # add pytan_loc and lib_dir to the PYTHONPATH variable
25 path_adds = [lib_dir, pytan_static_path]
26 [sys.path.append(aa) for aa in path_adds if aa not in sys.path]
27
28 # import pytan
29 import pytan
30
31 # create a dictionary of arguments for the pytan handler
32 handler_args = {}
33
34 # establish our connection info for the Tanium Server
35 handler_args['username'] = "Administrator"
36 handler_args['password'] = "Tanium2015!"
37 handler_args['host'] = "10.0.1.240"
38 handler_args['port'] = "443" # optional
39
40 # optional, level 0 is no output except warnings/errors
41 # level 1 through 12 are more and more verbose
42 handler_args['loglevel'] = 1
43
44 # optional, use a debug format for the logging output (uses two lines per log entry)
45 handler_args['debugformat'] = False
46
```

```

47 # optional, this saves all response objects to handler.session.ALL_REQUESTS_RESPONSES
48 # very useful for capturing the full exchange of XML requests and responses
49 handler_args['record_all_requests'] = True
50
51 # instantiate a handler using all of the arguments in the handler_args dictionary
52 print "...CALLING: pytan.handler() with args: {}".format(handler_args)
53 handler = pytan.Handler(**handler_args)
54
55 # print out the handler string
56 print "...OUTPUT: handler string: {}".format(handler)
57
58 # setup the arguments for the handler() class
59 kwargs = {}
60 kwargs["sensors"] = u'Dweedle Dee and Dum'
61 kwargs["qtype"] = u'manual'
62
63 print "...CALLING: handler.ask() with args: {}".format(kwargs)
64 try:
65     handler.ask(**kwargs)
66 except Exception as e:
67     print "...EXCEPTION: {}".format(e)
68     # this should throw an exception of type: pytan.exceptions.HandlerError
69     # uncomment to see full exception
70     # traceback.print_exc(file=sys.stdout)

```

### Invalid Ask Manual Question Too Many Parameter Blocks ({}).

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

- STDOUT from Example Python Code
- STDERR from Example Python Code
- Example Python Code

```

1  # import the basic python packages we need
2  import os
3  import sys
4  import tempfile
5  import pprint
6  import traceback
7
8  # disable python from generating a .pyc file
9  sys.dont_write_bytecode = True
10
11 # change me to the path of pytan if this script is not running from EXAMPLES/PYTAN_API
12 pytan_loc = "~/gh/pytan"
13 pytan_static_path = os.path.join(os.path.expanduser(pytan_loc), 'lib')
14
15 # Determine our script name, script dir
16 my_file = os.path.abspath(sys.argv[0])
17 my_dir = os.path.dirname(my_file)
18
19 # try to automatically determine the pytan lib directory by assuming it is in '../..lib/'
20 parent_dir = os.path.dirname(my_dir)
21 pytan_root_dir = os.path.dirname(parent_dir)
22 lib_dir = os.path.join(pytan_root_dir, 'lib')
23

```

```
24 # add pytan_loc and lib_dir to the PYTHONPATH variable
25 path_adds = [lib_dir, pytan_static_path]
26 [sys.path.append(aa) for aa in path_adds if aa not in sys.path]
27
28 # import pytan
29 import pytan
30
31 # create a dictionary of arguments for the pytan handler
32 handler_args = {}
33
34 # establish our connection info for the Tanium Server
35 handler_args['username'] = "Administrator"
36 handler_args['password'] = "Tanium2015!"
37 handler_args['host'] = "10.0.1.240"
38 handler_args['port'] = "443" # optional
39
40 # optional, level 0 is no output except warnings/errors
41 # level 1 through 12 are more and more verbose
42 handler_args['loglevel'] = 1
43
44 # optional, use a debug format for the logging output (uses two lines per log entry)
45 handler_args['debugformat'] = False
46
47 # optional, this saves all response objects to handler.session.ALL_REQUESTS_RESPONSES
48 # very useful for capturing the full exchange of XML requests and responses
49 handler_args['record_all_requests'] = True
50
51 # instantiate a handler using all of the arguments in the handler_args dictionary
52 print "...CALLING: pytan.handler() with args: {}".format(handler_args)
53 handler = pytan.Handler(**handler_args)
54
55 # print out the handler string
56 print "...OUTPUT: handler string: {}".format(handler)
57
58 # setup the arguments for the handler() class
59 kwargs = {}
60 kwargs["sensors"] = u'Folder Name Search with RegEx Match{dirname=Program Files,regex=.*}{}'
61 kwargs["qtype"] = u'manual'
62
63 print "...CALLING: handler.ask() with args: {}".format(kwargs)
64 try:
65     handler.ask(**kwargs)
66 except Exception as e:
67     print "...EXCEPTION: {}".format(e)
68     # this should throw an exception of type: pytan.exceptions.HumanParserError
69     # uncomment to see full exception
70     # traceback.print_exc(file=sys.stdout)
```

### Invalid Ask Manual Question Bad Option

Ask a question using an invalid option.

- STDOUT from Example Python Code
- STDERR from Example Python Code
- Example Python Code

```

1  # import the basic python packages we need
2  import os
3  import sys
4  import tempfile
5  import pprint
6  import traceback
7
8  # disable python from generating a .pyc file
9  sys.dont_write_bytecode = True
10
11  # change me to the path of pytan if this script is not running from EXAMPLES/PYTAN_API
12  pytan_loc = "~/gh/pytan"
13  pytan_static_path = os.path.join(os.path.expanduser(pytan_loc), 'lib')
14
15  # Determine our script name, script dir
16  my_file = os.path.abspath(sys.argv[0])
17  my_dir = os.path.dirname(my_file)
18
19  # try to automatically determine the pytan lib directory by assuming it is in '../..lib/'
20  parent_dir = os.path.dirname(my_dir)
21  pytan_root_dir = os.path.dirname(parent_dir)
22  lib_dir = os.path.join(pytan_root_dir, 'lib')
23
24  # add pytan_loc and lib_dir to the PYTHONPATH variable
25  path_adds = [lib_dir, pytan_static_path]
26  [sys.path.append(aa) for aa in path_adds if aa not in sys.path]
27
28  # import pytan
29  import pytan
30
31  # create a dictionary of arguments for the pytan handler
32  handler_args = {}
33
34  # establish our connection info for the Tanium Server
35  handler_args['username'] = "Administrator"
36  handler_args['password'] = "Tanium2015!"
37  handler_args['host'] = "10.0.1.240"
38  handler_args['port'] = "443" # optional
39
40  # optional, level 0 is no output except warnings/errors
41  # level 1 through 12 are more and more verbose
42  handler_args['loglevel'] = 1
43
44  # optional, use a debug format for the logging output (uses two lines per log entry)
45  handler_args['debugformat'] = False
46
47  # optional, this saves all response objects to handler.session.ALL_REQUESTS_RESPONSES
48  # very useful for capturing the full exchange of XML requests and responses
49  handler_args['record_all_requests'] = True
50
51  # instantiate a handler using all of the arguments in the handler_args dictionary
52  print "...CALLING: pytan.handler() with args: {}".format(handler_args)
53  handler = pytan.Handler(**handler_args)
54
55  # print out the handler string
56  print "...OUTPUT: handler string: {}".format(handler)
57
58  # setup the arguments for the handler() class

```

```
59 kwargs = {}
60 kwargs["sensors"] = u'Operating system, opt:bad'
61 kwargs["qtype"] = u'manual'
62
63 print "...CALLING: handler.ask() with args: {}".format(kwargs)
64 try:
65     handler.ask(**kwargs)
66 except Exception as e:
67     print "...EXCEPTION: {}".format(e)
68     # this should throw an exception of type: pytan.exceptions.HumanParserError
69     # uncomment to see full exception
70     # traceback.print_exc(file=sys.stdout)
```

### Invalid Ask Manual Question Missing Parameter Split

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

- STDOUT from Example Python Code
- STDERR from Example Python Code
- Example Python Code

```
1  # import the basic python packages we need
2  import os
3  import sys
4  import tempfile
5  import pprint
6  import traceback
7
8  # disable python from generating a .pyc file
9  sys.dont_write_bytecode = True
10
11 # change me to the path of pytan if this script is not running from EXAMPLES/PYTAN_API
12 pytan_loc = "~/gh/pytan"
13 pytan_static_path = os.path.join(os.path.expanduser(pytan_loc), 'lib')
14
15 # Determine our script name, script dir
16 my_file = os.path.abspath(sys.argv[0])
17 my_dir = os.path.dirname(my_file)
18
19 # try to automatically determine the pytan lib directory by assuming it is in '../..lib/'
20 parent_dir = os.path.dirname(my_dir)
21 pytan_root_dir = os.path.dirname(parent_dir)
22 lib_dir = os.path.join(pytan_root_dir, 'lib')
23
24 # add pytan_loc and lib_dir to the PYTHONPATH variable
25 path_adds = [lib_dir, pytan_static_path]
26 [sys.path.append(aa) for aa in path_adds if aa not in sys.path]
27
28 # import pytan
29 import pytan
30
31 # create a dictionary of arguments for the pytan handler
32 handler_args = {}
33
34 # establish our connection info for the Tanium Server
35 handler_args['username'] = "Administrator"
```



```

36 handler_args['password'] = "Tanium2015!"
37 handler_args['host'] = "10.0.1.240"
38 handler_args['port'] = "443" # optional
39
40 # optional, level 0 is no output except warnings/errors
41 # level 1 through 12 are more and more verbose
42 handler_args['loglevel'] = 1
43
44 # optional, use a debug format for the logging output (uses two lines per log entry)
45 handler_args['debugformat'] = False
46
47 # optional, this saves all response objects to handler.session.ALL_REQUESTS_RESPONSES
48 # very useful for capturing the full exchange of XML requests and responses
49 handler_args['record_all_requests'] = True
50
51 # instantiate a handler using all of the arguments in the handler_args dictionary
52 print "...CALLING: pytan.handler() with args: {}".format(handler_args)
53 handler = pytan.Handler(**handler_args)
54
55 # print out the handler string
56 print "...OUTPUT: handler string: {}".format(handler)
57
58 # setup the arguments for the handler() class
59 kwargs = {}
60 kwargs["sensors"] = u'Computer Name{Dweedle}'
61 kwargs["qtype"] = u'manual'
62
63 print "...CALLING: handler.ask() with args: {}".format(kwargs)
64 try:
65     handler.ask(**kwargs)
66 except Exception as e:
67     print "...EXCEPTION: {}".format(e)
68     # this should throw an exception of type: pytan.exceptions.HumanParserError
69     # uncomment to see full exception
70     # traceback.print_exc(file=sys.stdout)

```

## 1.2.12 SOAP API Examples for Platform Version 6.5.314.4301

Each of these sections contains examples that show the HTTP request and response for each step in a given workflow.

### Basic API Authentication

This is an example for how to authenticate against the SOAP API

#### Step 1 - Authenticate to the SOAP API via /auth

- URL: <https://10.0.1.240:443/auth>
- HTTP Method: GET
- Elapsed Time: 0:00:00.005930
- Step 1 Request Body
- Step 1 Response Body

- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip, deflate",
4   "Connection": "keep-alive",
5   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
6   "password": "VGfuaXVtMjAxNSE=",
7   "username": "QWRtaW5pc3RyYXRvcg=="
8 }
```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-length": "135",
4   "content-type": "text/plain; charset=us-ascii"
5 }
```

## Step 2 - Get the server version via /info.json

- URL: <https://10.0.1.240:443/info.json>
- HTTP Method: GET
- Elapsed Time: 0:00:00.013502
- Step 2 Request Body
- Step 2 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip, deflate",
4   "Connection": "keep-alive",
5   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
6   "session": "1-6911-da6e5b707595c7b5e42d4738029d5c97256bb813fc843855cb9c675c54dacb06c8153557fb60ea1"
7 }
```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-length": "86180",
4   "content-type": "application/json"
5 }
```

## Create User

Create a user called API Test User

### Step 1 - Authenticate to the SOAP API via /auth

- URL: <https://10.0.1.240:443/auth>
- HTTP Method: GET

- Elapsed Time: 0:00:00.005858
- Step 1 Request Body
- Step 1 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip, deflate",
4   "Connection": "keep-alive",
5   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
6   "password": "VGfuaXVtMjAxNSE=",
7   "username": "QWRtaW5pc3RyYXRvcg=="
8 }

```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-length": "135",
4   "content-type": "text/plain; charset=us-ascii"
5 }

```

## Step 2 - Get the server version via /info.json

- URL: <https://10.0.1.240:443/info.json>
- HTTP Method: GET
- Elapsed Time: 0:00:00.007489
- Step 2 Request Body
- Step 2 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip, deflate",
4   "Connection": "keep-alive",
5   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
6   "session": "1-6912-68d598b5538e59251cb4a35c2cb69fe3b5967d0713ca5031b766cabd6fd04d6126a9200d6843581"
7 }

```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-length": "86180",
4   "content-type": "application/json"
5 }

```

## Step 3 - Issue a GetObject to find the object to be deleted

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST

- Elapsed Time: 0:00:00.002030
- Step 3 Request Body
- Step 3 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "468",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6912-68d598b5538e59251cb4a35c2cb69fe3b5967d0713ca5031b766cabd6fd04d6126a9200d6843581"
9 }
```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }
```

#### Step 4 - Issue a GetObject to find an object

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.001208
- Step 4 Request Body
- Step 4 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "468",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6912-68d598b5538e59251cb4a35c2cb69fe3b5967d0713ca5031b766cabd6fd04d6126a9200d6843581"
9 }
```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }
```

**Step 5 - Issue an AddObject to add a User object**

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.006900
- Step 5 Request Body
- Step 5 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "1792",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6912-68d598b5538e59251cb4a35c2cb69fe3b5967d0713ca5031b766cabd6fd04d6126a9200d6843581"
9 }

```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }

```

**Step 6 - Issue a GetObject on the recently added object in order to get the full object**

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.001727
- Step 6 Request Body
- Step 6 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "2897",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6912-68d598b5538e59251cb4a35c2cb69fe3b5967d0713ca5031b766cabd6fd04d6126a9200d6843581"
9 }

```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }
```

#### Step 7 - Issue a GetObject to find the object to be deleted

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.002022
- Step 7 Request Body
- Step 7 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "468",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6912-68d598b5538e59251cb4a35c2cb69fe3b5967d0713ca5031b766cabd6fd04d6126a9200d6843581"
9 }
```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }
```

#### Step 8 - Issue a DeleteObject to delete an object

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.005434
- Step 8 Request Body
- Step 8 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "2846",
```

```

6  "Content-Type": "text/xml; charset=utf-8",
7  "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8  "session": "1-6912-68d598b5538e59251cb4a35c2cb69fe3b5967d0713ca5031b766cabd6fd04d6126a9200d6843581"
9  }

```

- Response Headers:

```

1  {
2  "connection": "keep-alive",
3  "content-encoding": "gzip",
4  "content-type": "text/xml; charset=UTF-8",
5  "transfer-encoding": "chunked"
6  }

```

## Create Package

Create a package called package49

### Step 1 - Authenticate to the SOAP API via /auth

- URL: <https://10.0.1.240:443/auth>
- HTTP Method: GET
- Elapsed Time: 0:00:00.006884
- Step 1 Request Body
- Step 1 Response Body
- Request Headers:

```

1  {
2  "Accept": "*/*",
3  "Accept-Encoding": "gzip, deflate",
4  "Connection": "keep-alive",
5  "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
6  "password": "VGfuaXVtMjAxNSE=",
7  "username": "QWRtaW5pc3RyYXRvcg=="
8  }

```

- Response Headers:

```

1  {
2  "connection": "keep-alive",
3  "content-length": "135",
4  "content-type": "text/plain; charset=us-ascii"
5  }

```

### Step 2 - Get the server version via /info.json

- URL: <https://10.0.1.240:443/info.json>
- HTTP Method: GET
- Elapsed Time: 0:00:00.013476
- Step 2 Request Body

- Step 2 Response Body
- Request Headers:

```
1 {  
2   "Accept": "*/*",  
3   "Accept-Encoding": "gzip, deflate",  
4   "Connection": "keep-alive",  
5   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",  
6   "session": "1-6913-78f5cc47430b95414a40a8125bc9afeaa40099d283b2a16b9d506393d17bde8bba85219dfd1e37d",  
7 }
```

- Response Headers:

```
1 {  
2   "connection": "keep-alive",  
3   "content-length": "86180",  
4   "content-type": "application/json"  
5 }
```

### Step 3 - Issue a GetObject to find the object to be deleted

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.002518
- Step 3 Request Body
- Step 3 Response Body
- Request Headers:

```
1 {  
2   "Accept": "*/*",  
3   "Accept-Encoding": "gzip",  
4   "Connection": "keep-alive",  
5   "Content-Length": "510",  
6   "Content-Type": "text/xml; charset=utf-8",  
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",  
8   "session": "1-6913-78f5cc47430b95414a40a8125bc9afeaa40099d283b2a16b9d506393d17bde8bba85219dfd1e37d",  
9 }
```

- Response Headers:

```
1 {  
2   "connection": "keep-alive",  
3   "content-length": "667",  
4   "content-type": "text/xml; charset=UTF-8"  
5 }
```

### Step 4 - Issue a GetObject to get the full object of a sensor for inclusion in a question or action

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.001815



- Step 4 Request Body
- Step 4 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "563",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6913-78f5cc47430b95414a40a8125bc9afeaa40099d283b2a16b9d506393d17bde8bba85219dfd1e37d"
9 }

```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }

```

#### Step 5 - Issue an AddObject to add a Group object for this package

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.004569
- Step 5 Request Body
- Step 5 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "647",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6913-78f5cc47430b95414a40a8125bc9afeaa40099d283b2a16b9d506393d17bde8bba85219dfd1e37d"
9 }

```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-length": "762",
4   "content-type": "text/xml; charset=UTF-8"
5 }

```

#### Step 6 - Issue a GetObject on the recently added object in order to get the full object

- URL: <https://10.0.1.240:443/soap>

- HTTP Method: POST
- Elapsed Time: 0:00:00.014005
- Step 6 Request Body
- Step 6 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "487",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6913-78f5cc47430b95414a40a8125bc9afeaa40099d283b2a16b9d506393d17bde8bba85219dfd1e37d",
9 }
```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }
```

#### Step 7 - Issue an AddObject to add a Group object for this package

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.007014
- Step 7 Request Body
- Step 7 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "5193",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6913-78f5cc47430b95414a40a8125bc9afeaa40099d283b2a16b9d506393d17bde8bba85219dfd1e37d",
9 }
```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }
```

**Step 8 - Issue a GetObject on the recently added object in order to get the full object**

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.001833
- Step 8 Request Body
- Step 8 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "5909",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6913-78f5cc47430b95414a40a8125bc9afeaa40099d283b2a16b9d506393d17bde8bba85219dfd1e37d"
9 }

```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }

```

**Step 9 - Issue a GetObject to find the object to be deleted**

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.001489
- Step 9 Request Body
- Step 9 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "510",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6913-78f5cc47430b95414a40a8125bc9afeaa40099d283b2a16b9d506393d17bde8bba85219dfd1e37d"
9 }

```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }
```

#### Step 10 - Issue a DeleteObject to delete an object

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.013096
- Step 10 Request Body
- Step 10 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "5858",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6913-78f5cc47430b95414a40a8125bc9afeaa40099d283b2a16b9d506393d17bde8bba85219dfd1e37d"
9 }
```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }
```

## Create Group

Create a group called All Windows Computers API Test

#### Step 1 - Authenticate to the SOAP API via /auth

- URL: <https://10.0.1.240:443/auth>
- HTTP Method: GET
- Elapsed Time: 0:00:00.006101
- Step 1 Request Body
- Step 1 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip, deflate",
4   "Connection": "keep-alive",
5   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
6   "password": "VGFuaXVtMjAxNSE=",
7   "username": "QWRtaW5pc3RyYXRvcg=="
8 }

```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-length": "135",
4   "content-type": "text/plain; charset=us-ascii"
5 }

```

## Step 2 - Get the server version via /info.json

- URL: <https://10.0.1.240:443/info.json>
- HTTP Method: GET
- Elapsed Time: 0:00:00.015088
- Step 2 Request Body
- Step 2 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip, deflate",
4   "Connection": "keep-alive",
5   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
6   "session": "1-6914-d31dcf0ab7d38ea2bd8256b9ae9e7d6bf5348b17cce76f3a15e01cf9f00047e409bc9697a8e93d02"
7 }

```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-length": "86180",
4   "content-type": "application/json"
5 }

```

## Step 3 - Issue a GetObject to find the object to be deleted

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.002629
- Step 3 Request Body
- Step 3 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "534",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6914-d31dcf0ab7d38ea2bd8256b9ae9e7d6bf5348b17cce76f3a15e01cf9f00047e409bc9697a8e93d02"
9 }
```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-length": "665",
4   "content-type": "text/xml; charset=UTF-8"
5 }
```

#### Step 4 - Issue a GetObject to get the full object of specified sensors for inclusion in a group

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.002210
- Step 4 Request Body
- Step 4 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "568",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6914-d31dcf0ab7d38ea2bd8256b9ae9e7d6bf5348b17cce76f3a15e01cf9f00047e409bc9697a8e93d02"
9 }
```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }
```

#### Step 5 - Issue an AddObject to add a Group object

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.007362

- Step 5 Request Body
- Step 5 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "692",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6914-d31dcf0ab7d38ea2bd8256b9ae9e7d6bf5348b17cce76f3a15e01cf9f00047e409bc9697a8e93d02"
9 }

```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-length": "762",
4   "content-type": "text/xml; charset=UTF-8"
5 }

```

#### Step 6 - Issue a GetObject on the recently added object in order to get the full object

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.011061
- Step 6 Request Body
- Step 6 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "487",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6914-d31dcf0ab7d38ea2bd8256b9ae9e7d6bf5348b17cce76f3a15e01cf9f00047e409bc9697a8e93d02"
9 }

```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }

```

#### Step 7 - Issue a GetObject to find the object to be deleted

- URL: <https://10.0.1.240:443/soap>

- HTTP Method: POST
- Elapsed Time: 0:00:00.001725
- Step 7 Request Body
- Step 7 Response Body
- Request Headers:

```
1 {  
2   "Accept": "*/*",  
3   "Accept-Encoding": "gzip",  
4   "Connection": "keep-alive",  
5   "Content-Length": "534",  
6   "Content-Type": "text/xml; charset=utf-8",  
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",  
8   "session": "1-6914-d31dcf0ab7d38ea2bd8256b9ae9e7d6bf5348b17cce76f3a15e01cf9f00047e409bc9697a8e93d02",  
9 }
```

- Response Headers:

```
1 {  
2   "connection": "keep-alive",  
3   "content-encoding": "gzip",  
4   "content-type": "text/xml; charset=UTF-8",  
5   "transfer-encoding": "chunked"  
6 }
```

### Step 8 - Issue a DeleteObject to delete an object

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.002416
- Step 8 Request Body
- Step 8 Response Body
- Request Headers:

```
1 {  
2   "Accept": "*/*",  
3   "Accept-Encoding": "gzip",  
4   "Connection": "keep-alive",  
5   "Content-Length": "1128",  
6   "Content-Type": "text/xml; charset=utf-8",  
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",  
8   "session": "1-6914-d31dcf0ab7d38ea2bd8256b9ae9e7d6bf5348b17cce76f3a15e01cf9f00047e409bc9697a8e93d02",  
9 }
```

- Response Headers:

```
1 {  
2   "connection": "keep-alive",  
3   "content-encoding": "gzip",  
4   "content-type": "text/xml; charset=UTF-8",  
5   "transfer-encoding": "chunked"  
6 }
```



## Create Whitelisted Url

Create a whitelisted url

### Step 1 - Authenticate to the SOAP API via /auth

- URL: <https://10.0.1.240:443/auth>
- HTTP Method: GET
- Elapsed Time: 0:00:00.005672
- Step 1 Request Body
- Step 1 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip, deflate",
4   "Connection": "keep-alive",
5   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
6   "password": "VGFuaXVtMjAxNSE=",
7   "username": "QWRtaW5pc3RyYXRvcg=="
8 }
```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-length": "135",
4   "content-type": "text/plain; charset=us-ascii"
5 }
```

### Step 2 - Get the server version via /info.json

- URL: <https://10.0.1.240:443/info.json>
- HTTP Method: GET
- Elapsed Time: 0:00:00.012832
- Step 2 Request Body
- Step 2 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip, deflate",
4   "Connection": "keep-alive",
5   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
6   "session": "1-6915-c6b9d055b32fff084601e262b1959939f54f4ee588a0003b5f75fc2a3e1fc64e2c66262f938b5a3"
7 }
```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-length": "86179",
4   "content-type": "application/json"
5 }
```

### Step 3 - Issue a GetObject to find the object to be deleted

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.277899
- Step 3 Request Body
- Step 3 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "480",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6915-c6b9d055b32fff084601e262b1959939f54f4ee588a0003b5f75fc2a3e1fc64e2c66262f938b5a3"
9 }
```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }
```

### Step 4 - Issue an AddObject to add a WhitelistedURL object

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.019582
- Step 4 Request Body
- Step 4 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "698",
6   "Content-Type": "text/xml; charset=utf-8",
```

```

7  "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8  "session": "1-6915-c6b9d055b32fff084601e262b1959939f54f4ee588a0003b5f75fc2a3e1fc64e2c66262f938b5a39"
9  }

```

- Response Headers:

```

1  {
2  "connection": "keep-alive",
3  "content-length": "1020",
4  "content-type": "text/xml; charset=UTF-8"
5  }

```

#### Step 5 - Issue a GetObject on the recently added object in order to get the full object

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.002186
- Step 5 Request Body
- Step 5 Response Body
- Request Headers:

```

1  {
2  "Accept": "*/*",
3  "Accept-Encoding": "gzip",
4  "Connection": "keep-alive",
5  "Content-Length": "738",
6  "Content-Type": "text/xml; charset=utf-8",
7  "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8  "session": "1-6915-c6b9d055b32fff084601e262b1959939f54f4ee588a0003b5f75fc2a3e1fc64e2c66262f938b5a39"
9  }

```

- Response Headers:

```

1  {
2  "connection": "keep-alive",
3  "content-length": "991",
4  "content-type": "text/xml; charset=UTF-8"
5  }

```

#### Step 6 - Issue a GetObject to find the object to be deleted

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.298260
- Step 6 Request Body
- Step 6 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "480",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6915-c6b9d055b32fff084601e262b1959939f54f4ee588a0003b5f75fc2a3e1fc64e2c66262f938b5a39"
9 }
```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }
```

### Step 7 - Issue a DeleteObject to delete an object

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.004764
- Step 7 Request Body
- Step 7 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "687",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6915-c6b9d055b32fff084601e262b1959939f54f4ee588a0003b5f75fc2a3e1fc64e2c66262f938b5a39"
9 }
```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }
```

### Create Package From JSON

Get a package object, add ‘API TEST’ to the name of the package object, delete any pre-existing package with the new name, then create a new package object with the new name

**Step 1 - Authenticate to the SOAP API via /auth**

- URL: <https://10.0.1.240:443/auth>
- HTTP Method: GET
- Elapsed Time: 0:00:00.008272
- Step 1 Request Body
- Step 1 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip, deflate",
4   "Connection": "keep-alive",
5   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
6   "password": "VGfuaXVtMjAxNSE=",
7   "username": "QWRtaW5pc3RyYXRvcg=="
8 }

```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-length": "135",
4   "content-type": "text/plain; charset=us-ascii"
5 }

```

**Step 2 - Get the server version via /info.json**

- URL: <https://10.0.1.240:443/info.json>
- HTTP Method: GET
- Elapsed Time: 0:00:00.014468
- Step 2 Request Body
- Step 2 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip, deflate",
4   "Connection": "keep-alive",
5   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
6   "session": "1-6916-f129723d3fb3582780ccc4638951752cca124ed745d06b4b45e6e4a925fa6be83376f5ba18a40970"
7 }

```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-length": "86179",
4   "content-type": "application/json"
5 }

```

### Step 3 - Issue a GetObject to find an object

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.002893
- Step 3 Request Body
- Step 3 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "499",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6916-f129723d3fb3582780ccc4638951752cca124ed745d06b4b45e6e4a925fa6be83376f5ba18a4097c"
9 }
```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }
```

### Step 4 - Issue a GetObject to find the object to be deleted

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.001583
- Step 4 Request Body
- Step 4 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "534",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6916-f129723d3fb3582780ccc4638951752cca124ed745d06b4b45e6e4a925fa6be83376f5ba18a4097c"
9 }
```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }

```

#### Step 5 - Issue a DeleteObject to delete an object

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.005890
- Step 5 Request Body
- Step 5 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "2407",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6916-f129723d3fb3582780ccc4638951752cca124ed745d06b4b45e6e4a925fa6be83376f5ba18a40970"
9 }

```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }

```

#### Step 6 - Issue an AddObject to add an object

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.008155
- Step 6 Request Body
- Step 6 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "2446",

```

```
6  "Content-Type": "text/xml; charset=utf-8",
7  "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8  "session": "1-6916-f129723d3fb3582780ccc4638951752cca124ed745d06b4b45e6e4a925fa6be83376f5ba18a40970"
9  }
```

- Response Headers:

```
1  {
2  "connection": "keep-alive",
3  "content-encoding": "gzip",
4  "content-type": "text/xml; charset=UTF-8",
5  "transfer-encoding": "chunked"
6  }
```

#### Step 7 - Issue a GetObject on the recently added object in order to get the full object

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.001707
- Step 7 Request Body
- Step 7 Response Body
- Request Headers:

```
1  {
2  "Accept": "*/*",
3  "Accept-Encoding": "gzip",
4  "Connection": "keep-alive",
5  "Content-Length": "2262",
6  "Content-Type": "text/xml; charset=utf-8",
7  "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8  "session": "1-6916-f129723d3fb3582780ccc4638951752cca124ed745d06b4b45e6e4a925fa6be83376f5ba18a40970"
9  }
```

- Response Headers:

```
1  {
2  "connection": "keep-alive",
3  "content-encoding": "gzip",
4  "content-type": "text/xml; charset=UTF-8",
5  "transfer-encoding": "chunked"
6  }
```

## Create User From JSON

Get a user object, add ‘API TEST’ to the name of the user object, delete any pre-existing user with the new name, then create a new user object with the new name

#### Step 1 - Authenticate to the SOAP API via /auth

- URL: <https://10.0.1.240:443/auth>
- HTTP Method: GET



- Elapsed Time: 0:00:00.005651
- Step 1 Request Body
- Step 1 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip, deflate",
4   "Connection": "keep-alive",
5   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
6   "password": "VGfuaXVtMjAxNSE=",
7   "username": "QWRtaW5pc3RyYXRvcg=="
8 }

```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-length": "135",
4   "content-type": "text/plain; charset=us-ascii"
5 }

```

## Step 2 - Get the server version via /info.json

- URL: <https://10.0.1.240:443/info.json>
- HTTP Method: GET
- Elapsed Time: 0:00:00.013454
- Step 2 Request Body
- Step 2 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip, deflate",
4   "Connection": "keep-alive",
5   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
6   "session": "1-6917-1e673660e061bc1418b184e82563f9713bd4f6e9fc9b9f1572dded518ff76b5c8ef44da01230c83"
7 }

```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-length": "86179",
4   "content-type": "application/json"
5 }

```

## Step 3 - Issue a GetObject to find an object

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST

- Elapsed Time: 0:00:00.001642
- Step 3 Request Body
- Step 3 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "482",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6917-1e673660e061bc1418b184e82563f9713bd4f6e9fc9b9f1572dded518ff76b5c8ef44da01230c831"
9 }
```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }
```

#### Step 4 - Issue a GetObject to find the object to be deleted

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.001725
- Step 4 Request Body
- Step 4 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "468",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6917-1e673660e061bc1418b184e82563f9713bd4f6e9fc9b9f1572dded518ff76b5c8ef44da01230c831"
9 }
```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }
```

**Step 5 - Issue a DeleteObject to delete an object**

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.003683
- Step 5 Request Body
- Step 5 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "2729",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6917-1e673660e061bc1418b184e82563f9713bd4f6e9fc9b9f1572dded518ff76b5c8ef44da01230c83b"
9 }

```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }

```

**Step 6 - Issue an AddObject to add an object**

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.004871
- Step 6 Request Body
- Step 6 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "2768",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6917-1e673660e061bc1418b184e82563f9713bd4f6e9fc9b9f1572dded518ff76b5c8ef44da01230c83b"
9 }

```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }
```

#### Step 7 - Issue a GetObject on the recently added object in order to get the full object

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.002068
- Step 7 Request Body
- Step 7 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "2780",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6917-1e673660e061bc1418b184e82563f9713bd4f6e9fc9b9f1572dded518ff76b5c8ef44da01230c83b"
9 }
```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }
```

#### Create Saved Question From JSON

Get a saved question object, add ‘ API TEST’ to the name of the saved question object, delete any pre-existing saved question with the new name, then create a new saved question object with the new name

#### Step 1 - Authenticate to the SOAP API via /auth

- URL: <https://10.0.1.240:443/auth>
- HTTP Method: GET
- Elapsed Time: 0:00:00.005834
- Step 1 Request Body
- Step 1 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip, deflate",
4   "Connection": "keep-alive",
5   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
6   "password": "VGFuaXVtMjAxNSE=",
7   "username": "QWRtaW5pc3RyYXRvcg=="
8 }

```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-length": "135",
4   "content-type": "text/plain; charset=us-ascii"
5 }

```

## Step 2 - Get the server version via /info.json

- URL: <https://10.0.1.240:443/info.json>
- HTTP Method: GET
- Elapsed Time: 0:00:00.013184
- Step 2 Request Body
- Step 2 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip, deflate",
4   "Connection": "keep-alive",
5   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
6   "session": "1-6918-c608bf977916033bd04dc9bf439ab3a54667c8810d210d950ca4fb630e8f009d9a843c7699f17c3"
7 }

```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-length": "86179",
4   "content-type": "application/json"
5 }

```

## Step 3 - Issue a GetObject to find an object

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.011342
- Step 3 Request Body
- Step 3 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "502",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6918-c608bf977916033bd04dc9bf439ab3a54667c8810d210d950ca4fb630e8f009d9a843c7699f17c33"
9 }
```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }
```

#### Step 4 - Issue a GetObject to find the object to be deleted

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.004730
- Step 4 Request Body
- Step 4 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "543",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6918-c608bf977916033bd04dc9bf439ab3a54667c8810d210d950ca4fb630e8f009d9a843c7699f17c33"
9 }
```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }
```

#### Step 5 - Issue a DeleteObject to delete an object

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.003435

- Step 5 Request Body
- Step 5 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "11006",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6918-c608bf977916033bd04dc9bf439ab3a54667c8810d210d950ca4fb630e8f009d9a843c7699f17c33"
9 }

```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }

```

#### Step 6 - Issue an AddObject to add an object

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.005398
- Step 6 Request Body
- Step 6 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "11047",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6918-c608bf977916033bd04dc9bf439ab3a54667c8810d210d950ca4fb630e8f009d9a843c7699f17c33"
9 }

```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-length": "831",
4   "content-type": "text/xml; charset=UTF-8"
5 }

```

#### Step 7 - Issue a GetObject on the recently added object in order to get the full object

- URL: <https://10.0.1.240:443/soap>

- HTTP Method: POST
- Elapsed Time: 0:00:00.007227
- Step 7 Request Body
- Step 7 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "555",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6918-c608bf977916033bd04dc9bf439ab3a54667c8810d210d950ca4fb630e8f009d9a843c7699f17c3"
9 }
```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }
```

## Create Action From JSON

Get an action object, then create a new object from that (aka re-deploy an action)

### Step 1 - Authenticate to the SOAP API via /auth

- URL: <https://10.0.1.240:443/auth>
- HTTP Method: GET
- Elapsed Time: 0:00:00.005778
- Step 1 Request Body
- Step 1 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip, deflate",
4   "Connection": "keep-alive",
5   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
6   "password": "VGfuaXVtMjAxNSE=",
7   "username": "QWRtaW5pc3RyYXRvcg=="
8 }
```

- Response Headers:



```

1 {
2   "connection": "keep-alive",
3   "content-length": "135",
4   "content-type": "text/plain; charset=us-ascii"
5 }

```

### Step 2 - Get the server version via /info.json

- URL: <https://10.0.1.240:443/info.json>
- HTTP Method: GET
- Elapsed Time: 0:00:00.014138
- Step 2 Request Body
- Step 2 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip, deflate",
4   "Connection": "keep-alive",
5   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
6   "session": "1-6919-207824e5cc87ff1c9c110ee9265a25a61182b7d49b2cabe1874429f40b39acb6836a7359b3ed2ac1"
7 }

```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-length": "86181",
4   "content-type": "application/json"
5 }

```

### Step 3 - Issue a GetObject to find an object

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.002205
- Step 3 Request Body
- Step 3 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "486",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6919-207824e5cc87ff1c9c110ee9265a25a61182b7d49b2cabe1874429f40b39acb6836a7359b3ed2ac1"
9 }

```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }
```

#### Step 4 - Issue an AddObject to add an object

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.004685
- Step 4 Request Body
- Step 4 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "1357",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6919-207824e5cc87ff1c9c110ee9265a25a61182b7d49b2cabe1874429f40b39acb6836a7359b3ed2ack"
9 }
```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }
```

#### Step 5 - Issue a GetObject on the recently added object in order to get the full object

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.002555
- Step 5 Request Body
- Step 5 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
```

```

5  "Content-Length": "1368",
6  "Content-Type": "text/xml; charset=utf-8",
7  "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8  "session": "1-6919-207824e5cc87ff1c9c110ee9265a25a61182b7d49b2cabe1874429f40b39acb6836a7359b3ed2ack
9  }

```

- Response Headers:

```

1  {
2  "connection": "keep-alive",
3  "content-encoding": "gzip",
4  "content-type": "text/xml; charset=UTF-8",
5  "transfer-encoding": "chunked"
6  }

```

## Create Sensor From JSON

Get a sensor object, add 'API TEST' to the name of the sensor object, delete any pre-existing sensor with the new name, then create a new sensor object with the new name

### Step 1 - Authenticate to the SOAP API via /auth

- URL: <https://10.0.1.240:443/auth>
- HTTP Method: GET
- Elapsed Time: 0:00:00.006732
- Step 1 Request Body
- Step 1 Response Body
- Request Headers:

```

1  {
2  "Accept": "*/*",
3  "Accept-Encoding": "gzip, deflate",
4  "Connection": "keep-alive",
5  "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
6  "password": "VGFuaXVtMjAxNSE=",
7  "username": "QWRtaW5pc3RyYXRvcg=="
8  }

```

- Response Headers:

```

1  {
2  "connection": "keep-alive",
3  "content-length": "135",
4  "content-type": "text/plain; charset=us-ascii"
5  }

```

### Step 2 - Get the server version via /info.json

- URL: <https://10.0.1.240:443/info.json>
- HTTP Method: GET

- Elapsed Time: 0:00:00.007148
- Step 2 Request Body
- Step 2 Response Body
- Request Headers:

```
1 {  
2   "Accept": "*/*",  
3   "Accept-Encoding": "gzip, deflate",  
4   "Connection": "keep-alive",  
5   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",  
6   "session": "1-6920-fc22898975af19a91cf1560c9c9cfea6669820e7ed35e7a1672ed77d3adb12add41e4acb3ba2883",  
7 }
```

- Response Headers:

```
1 {  
2   "connection": "keep-alive",  
3   "content-length": "86180",  
4   "content-type": "application/json"  
5 }
```

### Step 3 - Issue a GetObject to find an object

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.001364
- Step 3 Request Body
- Step 3 Response Body
- Request Headers:

```
1 {  
2   "Accept": "*/*",  
3   "Accept-Encoding": "gzip",  
4   "Connection": "keep-alive",  
5   "Content-Length": "507",  
6   "Content-Type": "text/xml; charset=utf-8",  
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",  
8   "session": "1-6920-fc22898975af19a91cf1560c9c9cfea6669820e7ed35e7a1672ed77d3adb12add41e4acb3ba2883",  
9 }
```

- Response Headers:

```
1 {  
2   "connection": "keep-alive",  
3   "content-encoding": "gzip",  
4   "content-type": "text/xml; charset=UTF-8",  
5   "transfer-encoding": "chunked"  
6 }
```

### Step 4 - Issue a GetObject to find the object to be deleted

- URL: <https://10.0.1.240:443/soap>

- HTTP Method: POST
- Elapsed Time: 0:00:00.001953
- Step 4 Request Body
- Step 4 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "523",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6920-fc22898975af19a91cf1560c9c9cfea6669820e7ed35e7a1672ed77d3adb12add41e4acb3ba2883"
9 }

```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }

```

### Step 5 - Issue a DeleteObject to delete an object

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.007358
- Step 5 Request Body
- Step 5 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "1961",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6920-fc22898975af19a91cf1560c9c9cfea6669820e7ed35e7a1672ed77d3adb12add41e4acb3ba2883"
9 }

```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }

```

## Step 6 - Issue an AddObject to add an object

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.005388
- Step 6 Request Body
- Step 6 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "1977",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6920-fc22898975af19a91cf1560c9c9cfea6669820e7ed35e7a1672ed77d3adb12add41e4acb3ba28830"
9 }
```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-length": "763",
4   "content-type": "text/xml; charset=UTF-8"
5 }
```

## Step 7 - Issue a GetObject on the recently added object in order to get the full object

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.025365
- Step 7 Request Body
- Step 7 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "488",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6920-fc22898975af19a91cf1560c9c9cfea6669820e7ed35e7a1672ed77d3adb12add41e4acb3ba28830"
9 }
```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
```

```

4  "content-type": "text/xml; charset=UTF-8",
5  "transfer-encoding": "chunked"
6  }

```

## Create Question From JSON

Get a question object, then create a new object from that (aka re-ask a question)

### Step 1 - Authenticate to the SOAP API via /auth

- URL: <https://10.0.1.240:443/auth>
- HTTP Method: GET
- Elapsed Time: 0:00:00.005651
- Step 1 Request Body
- Step 1 Response Body
- Request Headers:

```

1  {
2  "Accept": "*/*",
3  "Accept-Encoding": "gzip, deflate",
4  "Connection": "keep-alive",
5  "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
6  "password": "VGFuaXVtMjAxNSE=",
7  "username": "QWRtaW5pc3RyYXRvcg=="
8  }

```

- Response Headers:

```

1  {
2  "connection": "keep-alive",
3  "content-length": "135",
4  "content-type": "text/plain; charset=us-ascii"
5  }

```

### Step 2 - Get the server version via /info.json

- URL: <https://10.0.1.240:443/info.json>
- HTTP Method: GET
- Elapsed Time: 0:00:00.013641
- Step 2 Request Body
- Step 2 Response Body
- Request Headers:

```

1  {
2  "Accept": "*/*",
3  "Accept-Encoding": "gzip, deflate",
4  "Connection": "keep-alive",
5  "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",

```

```
6  "session": "1-6921-b6479c197f6840955f0c6a1228688180964bb93eec39fcb7e7689fb0cfac468d4c943eebfd07277c  
7  }
```

- Response Headers:

```
1  {  
2  "connection": "keep-alive",  
3  "content-length": "86180",  
4  "content-type": "application/json"  
5  }
```

### Step 3 - Issue a GetObject to find an object

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.002235
- Step 3 Request Body
- Step 3 Response Body
- Request Headers:

```
1  {  
2  "Accept": "*/*",  
3  "Accept-Encoding": "gzip",  
4  "Connection": "keep-alive",  
5  "Content-Length": "490",  
6  "Content-Type": "text/xml; charset=utf-8",  
7  "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",  
8  "session": "1-6921-b6479c197f6840955f0c6a1228688180964bb93eec39fcb7e7689fb0cfac468d4c943eebfd07277c  
9  }
```

- Response Headers:

```
1  {  
2  "connection": "keep-alive",  
3  "content-encoding": "gzip",  
4  "content-type": "text/xml; charset=UTF-8",  
5  "transfer-encoding": "chunked"  
6  }
```

### Step 4 - Issue an AddObject to add an object

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.005462
- Step 4 Request Body
- Step 4 Response Body
- Request Headers:



```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "2146",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6921-b6479c197f6840955f0c6a1228688180964bb93eec39fcb7e7689fb0cfac468d4c943eebfd072770"
9 }

```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-length": "769",
4   "content-type": "text/xml; charset=UTF-8"
5 }

```

### Step 5 - Issue a GetObject on the recently added object in order to get the full object

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.014726
- Step 5 Request Body
- Step 5 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "494",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6921-b6479c197f6840955f0c6a1228688180964bb93eec39fcb7e7689fb0cfac468d4c943eebfd072770"
9 }

```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }

```

### Create Whitelisted Url From JSON

Get a whitelisted url object, add 'API TEST' to the url\_regex of the whitelisted url object, delete any pre-existing whitelisted url with the new url\_regex, then create a new whitelisted url object with the new url\_regex

### Step 1 - Authenticate to the SOAP API via /auth

- URL: <https://10.0.1.240:443/auth>
- HTTP Method: GET
- Elapsed Time: 0:00:00.007380
- Step 1 Request Body
- Step 1 Response Body
- Request Headers:

```
1 {  
2   "Accept": "*/*",  
3   "Accept-Encoding": "gzip, deflate",  
4   "Connection": "keep-alive",  
5   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",  
6   "password": "VGfuaXVtMjAxNSE=",  
7   "username": "QWRtaW5pc3RyYXRvcg=="  
8 }
```

- Response Headers:

```
1 {  
2   "connection": "keep-alive",  
3   "content-length": "135",  
4   "content-type": "text/plain; charset=us-ascii"  
5 }
```

### Step 2 - Get the server version via /info.json

- URL: <https://10.0.1.240:443/info.json>
- HTTP Method: GET
- Elapsed Time: 0:00:00.006730
- Step 2 Request Body
- Step 2 Response Body
- Request Headers:

```
1 {  
2   "Accept": "*/*",  
3   "Accept-Encoding": "gzip, deflate",  
4   "Connection": "keep-alive",  
5   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",  
6   "session": "1-6922-c6924768f0749f065a818926b06de2a4da74c8f3d64ad752386557a9d4f379fb3163056e554fd94"  
7 }
```

- Response Headers:

```
1 {  
2   "connection": "keep-alive",  
3   "content-length": "86180",  
4   "content-type": "application/json"  
5 }
```

**Step 3 - Issue a GetObject to find an object**

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.287918
- Step 3 Request Body
- Step 3 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "480",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6922-c6924768f0749f065a818926b06de2a4da74c8f3d64ad752386557a9d4f379fb3163056e554fd941"
9 }

```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }

```

**Step 4 - Issue a GetObject to find the object to be deleted**

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.275546
- Step 4 Request Body
- Step 4 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "480",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6922-c6924768f0749f065a818926b06de2a4da74c8f3d64ad752386557a9d4f379fb3163056e554fd941"
9 }

```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }
```

#### Step 5 - Issue a DeleteObject to delete an object

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.009114
- Step 5 Request Body
- Step 5 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "538",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6922-c6924768f0749f065a818926b06de2a4da74c8f3d64ad752386557a9d4f379fb3163056e554fd94"
9 }
```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-length": "957",
4   "content-type": "text/xml; charset=UTF-8"
5 }
```

#### Step 6 - Issue an AddObject to add an object

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.022226
- Step 6 Request Body
- Step 6 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "575",
6   "Content-Type": "text/xml; charset=utf-8",
```

```

7  "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8  "session": "1-6922-c6924768f0749f065a818926b06de2a4da74c8f3d64ad752386557a9d4f379fb3163056e554fd941"
9  }

```

- Response Headers:

```

1  {
2  "connection": "keep-alive",
3  "content-length": "866",
4  "content-type": "text/xml; charset=UTF-8"
5  }

```

### Step 7 - Issue a GetObject on the recently added object in order to get the full object

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.001966
- Step 7 Request Body
- Step 7 Response Body
- Request Headers:

```

1  {
2  "Accept": "*/*",
3  "Accept-Encoding": "gzip",
4  "Connection": "keep-alive",
5  "Content-Length": "589",
6  "Content-Type": "text/xml; charset=utf-8",
7  "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8  "session": "1-6922-c6924768f0749f065a818926b06de2a4da74c8f3d64ad752386557a9d4f379fb3163056e554fd941"
9  }

```

- Response Headers:

```

1  {
2  "connection": "keep-alive",
3  "content-length": "837",
4  "content-type": "text/xml; charset=UTF-8"
5  }

```

## Create Group From JSON

Get a group object, add ' API TEST' to the name of the group object, delete any pre-existing group with the new name, then create a new group object with the new name

### Step 1 - Authenticate to the SOAP API via /auth

- URL: <https://10.0.1.240:443/auth>
- HTTP Method: GET
- Elapsed Time: 0:00:00.008859
- Step 1 Request Body

- Step 1 Response Body
- Request Headers:

```
1 {  
2   "Accept": "*/*",  
3   "Accept-Encoding": "gzip, deflate",  
4   "Connection": "keep-alive",  
5   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",  
6   "password": "VGFuaXVtMjAxNSE=",  
7   "username": "QWRtaW5pc3RyYXRvcg=="  
8 }
```

- Response Headers:

```
1 {  
2   "connection": "keep-alive",  
3   "content-length": "135",  
4   "content-type": "text/plain; charset=us-ascii"  
5 }
```

### Step 2 - Get the server version via /info.json

- URL: <https://10.0.1.240:443/info.json>
- HTTP Method: GET
- Elapsed Time: 0:00:00.014108
- Step 2 Request Body
- Step 2 Response Body
- Request Headers:

```
1 {  
2   "Accept": "*/*",  
3   "Accept-Encoding": "gzip, deflate",  
4   "Connection": "keep-alive",  
5   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",  
6   "session": "1-6923-1b5896b8960f90f883e294e107cd909333673f42fa0e820a1e58cd246a1e64db99db50b01c7aa28"  
7 }
```

- Response Headers:

```
1 {  
2   "connection": "keep-alive",  
3   "content-length": "86180",  
4   "content-type": "application/json"  
5 }
```

### Step 3 - Issue a GetObject to find an object

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.003172
- Step 3 Request Body

- Step 3 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "517",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6923-1b5896b8960f90f883e294e107cd909333673f42fa0e820a1e58cd246a1e64db99db50b01c7aa28"
9 }

```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-length": "941",
4   "content-type": "text/xml; charset=UTF-8"
5 }

```

#### Step 4 - Issue a GetObject to find the object to be deleted

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.001771
- Step 4 Request Body
- Step 4 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "526",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6923-1b5896b8960f90f883e294e107cd909333673f42fa0e820a1e58cd246a1e64db99db50b01c7aa28"
9 }

```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-length": "952",
4   "content-type": "text/xml; charset=UTF-8"
5 }

```

#### Step 5 - Issue a DeleteObject to delete an object

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST

- Elapsed Time: 0:00:00.002377
- Step 5 Request Body
- Step 5 Response Body
- Request Headers:

```
1 {  
2   "Accept": "*/*",  
3   "Accept-Encoding": "gzip",  
4   "Connection": "keep-alive",  
5   "Content-Length": "620",  
6   "Content-Type": "text/xml; charset=utf-8",  
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",  
8   "session": "1-6923-1b5896b8960f90f883e294e107cd909333673f42fa0e820a1e58cd246a1e64db99db50b01c7aa28",  
9 }
```

- Response Headers:

```
1 {  
2   "connection": "keep-alive",  
3   "content-length": "950",  
4   "content-type": "text/xml; charset=UTF-8"  
5 }
```

#### Step 6 - Issue an AddObject to add an object

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.003491
- Step 6 Request Body
- Step 6 Response Body
- Request Headers:

```
1 {  
2   "Accept": "*/*",  
3   "Accept-Encoding": "gzip",  
4   "Connection": "keep-alive",  
5   "Content-Length": "658",  
6   "Content-Type": "text/xml; charset=utf-8",  
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",  
8   "session": "1-6923-1b5896b8960f90f883e294e107cd909333673f42fa0e820a1e58cd246a1e64db99db50b01c7aa28",  
9 }
```

- Response Headers:

```
1 {  
2   "connection": "keep-alive",  
3   "content-length": "762",  
4   "content-type": "text/xml; charset=UTF-8"  
5 }
```



**Step 7 - Issue a GetObject on the recently added object in order to get the full object**

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.014759
- Step 7 Request Body
- Step 7 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "487",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6923-1b5896b8960f90f883e294e107cd909333673f42fa0e820a1e58cd246a1e64db99db50b01c7aa28"
9 }

```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-length": "934",
4   "content-type": "text/xml; charset=UTF-8"
5 }

```

**Deploy Action Simple**

Deploy an action using the package ‘Distribute Tanium Standard Utilities’ to all computers, wait for result data to be complete, and then get result data

**Step 1 - Authenticate to the SOAP API via /auth**

- URL: <https://10.0.1.240:443/auth>
- HTTP Method: GET
- Elapsed Time: 0:00:00.005787
- Step 1 Request Body
- Step 1 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip, deflate",
4   "Connection": "keep-alive",
5   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
6   "password": "VGfuaXVtMjAxNSE=",
7   "username": "QWRtaW5pc3RyYXRvcg=="
8 }

```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-length": "135",
4   "content-type": "text/plain; charset=us-ascii"
5 }
```

## Step 2 - Get the server version via /info.json

- URL: <https://10.0.1.240:443/info.json>
- HTTP Method: GET
- Elapsed Time: 0:00:00.007137
- Step 2 Request Body
- Step 2 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip, deflate",
4   "Connection": "keep-alive",
5   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
6   "session": "1-6924-eb9336c22ad9373c3520002c9e27872eb9f25aa1b7b74bb32d1005293ff1127f6ca5b39c0647aff1"
7 }
```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-length": "86181",
4   "content-type": "application/json"
5 }
```

## Step 3 - Issue a GetObject to get the full object of a package for inclusion in an action

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.003097
- Step 3 Request Body
- Step 3 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "581",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
```

```

8  "session": "1-6924-eb9336c22ad9373c3520002c9e27872eb9f25aalb7b74bb32d1005293ff1127f6ca5b39c0647afff
9  }

```

- Response Headers:

```

1  {
2  "connection": "keep-alive",
3  "content-encoding": "gzip",
4  "content-type": "text/xml; charset=UTF-8",
5  "transfer-encoding": "chunked"
6  }

```

#### Step 4 - Issue an AddObject to add a list of SavedActions (6.5 logic)

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.004732
- Step 4 Request Body
- Step 4 Response Body
- Request Headers:

```

1  {
2  "Accept": "*/*",
3  "Accept-Encoding": "gzip",
4  "Connection": "keep-alive",
5  "Content-Length": "1493",
6  "Content-Type": "text/xml; charset=utf-8",
7  "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8  "session": "1-6924-eb9336c22ad9373c3520002c9e27872eb9f25aalb7b74bb32d1005293ff1127f6ca5b39c0647afff
9  }

```

- Response Headers:

```

1  {
2  "connection": "keep-alive",
3  "content-encoding": "gzip",
4  "content-type": "text/xml; charset=UTF-8",
5  "transfer-encoding": "chunked"
6  }

```

#### Step 5 - Issue a GetObject on the recently added object in order to get the full object

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.003382
- Step 5 Request Body
- Step 5 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "1523",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6924-eb9336c22ad9373c3520002c9e27872eb9f25aa1b7b74bb32d1005293ff1127f6ca5b39c0647aff1"
9 }
```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }
```

#### Step 6 - Issue a GetObject to get the last action created for a SavedAction

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.002423
- Step 6 Request Body
- Step 6 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "557",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6924-eb9336c22ad9373c3520002c9e27872eb9f25aa1b7b74bb32d1005293ff1127f6ca5b39c0647aff1"
9 }
```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }
```

#### Step 7 - Issue a GetObject to get the package for an Action

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.001937

- Step 7 Request Body
- Step 7 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "600",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6924-eb9336c22ad9373c3520002c9e27872eb9f25aa1b7b74bb32d1005293ff1127f6ca5b39c0647afff"
9 }

```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }

```

#### Step 8 - Issue a GetResultInfo on an Action to have the Server create a question that tracks the results for a Deployed Action

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.003328
- Step 8 Request Body
- Step 8 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "552",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6924-eb9336c22ad9373c3520002c9e27872eb9f25aa1b7b74bb32d1005293ff1127f6ca5b39c0647afff"
9 }

```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }

```

**Step 9 - Issue a GetObject on the package for an action to get the full object**

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.001953
- Step 9 Request Body
- Step 9 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "625",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6924-eb9336c22ad9373c3520002c9e27872eb9f25aa1b7b74bb32d1005293ff1127f6ca5b39c0647aff1"
9 }
```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }
```

**Step 10 - ID 522: Issuing an AddObject of a Question object with no Selects and the same Group used by the Action object. The number of systems that should successfully run the Action will be taken from result\_info.passed\_count for the Question asked when all answers for the question have reported in.**

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.002457
- Step 10 Request Body
- Step 10 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "525",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6924-eb9336c22ad9373c3520002c9e27872eb9f25aa1b7b74bb32d1005293ff1127f6ca5b39c0647aff1"
9 }
```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-length": "769",
4   "content-type": "text/xml; charset=UTF-8"
5 }

```

#### Step 11 - Issue a GetObject on the recently added object in order to get the full object

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.002371
- Step 11 Request Body
- Step 11 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "494",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6924-eb9336c22ad9373c3520002c9e27872eb9f25aa1b7b74bb32d1005293ff1127f6ca5b39c0647aff"
9 }

```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }

```

#### Step 12 - Issue a GetResultInfo for a Question to check the current progress of answers

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.001251
- Step 12 Request Body
- Step 12 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "498",
6   "Content-Type": "text/xml; charset=utf-8",

```

```
7 "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8 "session": "1-6924-eb9336c22ad9373c3520002c9e27872eb9f25aalb7b74bb32d1005293ff1127f6ca5b39c0647afff
9 }
```

- Response Headers:

```
1 {
2 "connection": "keep-alive",
3 "content-encoding": "gzip",
4 "content-type": "text/xml; charset=UTF-8",
5 "transfer-encoding": "chunked"
6 }
```

### Step 13 - Issue a GetResultInfo for a Question to check the current progress of answers

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.002060
- Step 13 Request Body
- Step 13 Response Body
- Request Headers:

```
1 {
2 "Accept": "*/*",
3 "Accept-Encoding": "gzip",
4 "Connection": "keep-alive",
5 "Content-Length": "498",
6 "Content-Type": "text/xml; charset=utf-8",
7 "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8 "session": "1-6924-eb9336c22ad9373c3520002c9e27872eb9f25aalb7b74bb32d1005293ff1127f6ca5b39c0647afff
9 }
```

- Response Headers:

```
1 {
2 "connection": "keep-alive",
3 "content-encoding": "gzip",
4 "content-type": "text/xml; charset=UTF-8",
5 "transfer-encoding": "chunked"
6 }
```

### Step 14 - Issue a GetObject for an Action in order to have access to the latest values for stopped\_flag and status

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.002729
- Step 14 Request Body
- Step 14 Response Body
- Request Headers:



```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "1460",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6924-eb9336c22ad9373c3520002c9e27872eb9f25aalb7b74bb32d1005293ff1127f6ca5b39c0647aff1"
9 }

```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }

```

#### Step 15 - Issue a GetResultInfo for an Action to ensure fresh data is available for a GetResultData call

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.002933
- Step 15 Request Body
- Step 15 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "552",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6924-eb9336c22ad9373c3520002c9e27872eb9f25aalb7b74bb32d1005293ff1127f6ca5b39c0647aff1"
9 }

```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }

```

#### Step 16 - Issue a GetResultData with the aggregate option set to True. This will return row counts of machines that have answered instead of all the data

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST

- Elapsed Time: 0:00:00.002875
- Step 16 Request Body
- Step 16 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "626",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6924-eb9336c22ad9373c3520002c9e27872eb9f25aa1b7b74bb32d1005293ff1127f6ca5b39c0647aff1",
9 }
```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }
```

#### Step 17 - Issue a GetObject for an Action in order to have access to the latest values for stopped\_flag and status

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.003173
- Step 17 Request Body
- Step 17 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "1460",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6924-eb9336c22ad9373c3520002c9e27872eb9f25aa1b7b74bb32d1005293ff1127f6ca5b39c0647aff1",
9 }
```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }
```

**Step 18 - Issue a GetResultInfo for an Action to ensure fresh data is available for a GetResultData call**

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.003020
- Step 18 Request Body
- Step 18 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "552",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6924-eb9336c22ad9373c3520002c9e27872eb9f25aa1b7b74bb32d1005293ff1127f6ca5b39c0647afff"
9 }

```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }

```

**Step 19 - Issue a GetResultData with the aggregate option set to True. This will return row counts of machines that have answered instead of all the data**

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.002767
- Step 19 Request Body
- Step 19 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "626",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6924-eb9336c22ad9373c3520002c9e27872eb9f25aa1b7b74bb32d1005293ff1127f6ca5b39c0647afff"
9 }

```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }
```

**Step 20 - Issue a GetObject for an Action in order to have access to the latest values for stopped\_flag and status**

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.003202
- Step 20 Request Body
- Step 20 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "1460",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6924-eb9336c22ad9373c3520002c9e27872eb9f25aa1b7b74bb32d1005293ff1127f6ca5b39c0647aff1"
9 }
```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }
```

**Step 21 - Issue a GetResultInfo for an Action to ensure fresh data is available for a GetResultData call**

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.002996
- Step 21 Request Body
- Step 21 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
```

```

5  "Content-Length": "552",
6  "Content-Type": "text/xml; charset=utf-8",
7  "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8  "session": "1-6924-eb9336c22ad9373c3520002c9e27872eb9f25aa1b7b74bb32d1005293ff1127f6ca5b39c0647afff"
9  }

```

- Response Headers:

```

1  {
2  "connection": "keep-alive",
3  "content-encoding": "gzip",
4  "content-type": "text/xml; charset=UTF-8",
5  "transfer-encoding": "chunked"
6  }

```

**Step 22 - Issue a GetResultData with the aggregate option set to True.** This will return row counts of machines that have answered instead of all the data

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.002978
- Step 22 Request Body
- Step 22 Response Body
- Request Headers:

```

1  {
2  "Accept": "*/*",
3  "Accept-Encoding": "gzip",
4  "Connection": "keep-alive",
5  "Content-Length": "626",
6  "Content-Type": "text/xml; charset=utf-8",
7  "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8  "session": "1-6924-eb9336c22ad9373c3520002c9e27872eb9f25aa1b7b74bb32d1005293ff1127f6ca5b39c0647afff"
9  }

```

- Response Headers:

```

1  {
2  "connection": "keep-alive",
3  "content-encoding": "gzip",
4  "content-type": "text/xml; charset=UTF-8",
5  "transfer-encoding": "chunked"
6  }

```

**Step 23 - Issue a GetObject for an Action in order to have access to the latest values for stopped\_flag and status**

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.003231
- Step 23 Request Body

- Step 23 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "1460",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6924-eb9336c22ad9373c3520002c9e27872eb9f25aa1b7b74bb32d1005293ff1127f6ca5b39c0647aff1"
9 }
```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }
```

#### Step 24 - Issue a `GetResultInfo` for an Action to ensure fresh data is available for a `GetResultData` call

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.003172
- Step 24 Request Body
- Step 24 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "552",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6924-eb9336c22ad9373c3520002c9e27872eb9f25aa1b7b74bb32d1005293ff1127f6ca5b39c0647aff1"
9 }
```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }
```

**Step 25 - Issue a GetResultData with the aggregate option set to True.** This will return row counts of machines that have answered instead of all the data

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.002963
- Step 25 Request Body
- Step 25 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "626",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6924-eb9336c22ad9373c3520002c9e27872eb9f25aa1b7b74bb32d1005293ff1127f6ca5b39c0647aff1"
9 }
```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }
```

**Step 26 - Issue a GetObject for an Action in order to have access to the latest values for stopped\_flag and status**

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.003103
- Step 26 Request Body
- Step 26 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "1460",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6924-eb9336c22ad9373c3520002c9e27872eb9f25aa1b7b74bb32d1005293ff1127f6ca5b39c0647aff1"
9 }
```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }
```

#### Step 27 - Issue a `GetResultInfo` for an Action to ensure fresh data is available for a `GetResultData` call

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.003306
- Step 27 Request Body
- Step 27 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "552",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6924-eb9336c22ad9373c3520002c9e27872eb9f25aa1b7b74bb32d1005293ff1127f6ca5b39c0647aff1"
9 }
```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }
```

#### Step 28 - Issue a `GetResultData` with the aggregate option set to `True`. This will return row counts of machines that have answered instead of all the data

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.003108
- Step 28 Request Body
- Step 28 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
```



```

5  "Content-Length": "626",
6  "Content-Type": "text/xml; charset=utf-8",
7  "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8  "session": "1-6924-eb9336c22ad9373c3520002c9e27872eb9f25aa1b7b74bb32d1005293ff1127f6ca5b39c0647aff1"
9  }

```

- Response Headers:

```

1  {
2  "connection": "keep-alive",
3  "content-encoding": "gzip",
4  "content-type": "text/xml; charset=UTF-8",
5  "transfer-encoding": "chunked"
6  }

```

### Step 29 - Issue a GetObject for an Action in order to have access to the latest values for stopped\_flag and status

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.002511
- Step 29 Request Body
- Step 29 Response Body
- Request Headers:

```

1  {
2  "Accept": "*/*",
3  "Accept-Encoding": "gzip",
4  "Connection": "keep-alive",
5  "Content-Length": "1460",
6  "Content-Type": "text/xml; charset=utf-8",
7  "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8  "session": "1-6924-eb9336c22ad9373c3520002c9e27872eb9f25aa1b7b74bb32d1005293ff1127f6ca5b39c0647aff1"
9  }

```

- Response Headers:

```

1  {
2  "connection": "keep-alive",
3  "content-encoding": "gzip",
4  "content-type": "text/xml; charset=UTF-8",
5  "transfer-encoding": "chunked"
6  }

```

### Step 30 - Issue a GetResultInfo for an Action to ensure fresh data is available for a GetResultData call

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.002556
- Step 30 Request Body

- Step 30 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "552",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6924-eb9336c22ad9373c3520002c9e27872eb9f25aa1b7b74bb32d1005293ff1127f6ca5b39c0647aff1"
9 }
```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }
```

**Step 31 - Issue a GetResultData for an Action with the aggregate option set to False. This will return all of the Action Statuses for each computer that have run this Action**

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.002572
- Step 31 Request Body
- Step 31 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "580",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6924-eb9336c22ad9373c3520002c9e27872eb9f25aa1b7b74bb32d1005293ff1127f6ca5b39c0647aff1"
9 }
```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }
```

## Deploy Action Simple Without Results

Deploy an action using the package ‘Distribute Tanium Standard Utilities’ to all computers and do not wait for result data to be complete and do not get result data

### Step 1 - Authenticate to the SOAP API via /auth

- URL: <https://10.0.1.240:443/auth>
- HTTP Method: GET
- Elapsed Time: 0:00:00.008210
- Step 1 Request Body
- Step 1 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip, deflate",
4   "Connection": "keep-alive",
5   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
6   "password": "VGfuaXVtMjAxNSE=",
7   "username": "QWRtaW5pc3RyYXRvcg=="
8 }
```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-length": "135",
4   "content-type": "text/plain; charset=us-ascii"
5 }
```

### Step 2 - Get the server version via /info.json

- URL: <https://10.0.1.240:443/info.json>
- HTTP Method: GET
- Elapsed Time: 0:00:00.014382
- Step 2 Request Body
- Step 2 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip, deflate",
4   "Connection": "keep-alive",
5   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
6   "session": "1-6925-6c9d7f096a4dc99b8e34bb9d09d699963f8235f2534720facac0830700c5f8799b79fa5d2d9e828"
7 }
```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-length": "86180",
4   "content-type": "application/json"
5 }
```

### Step 3 - Issue a `GetObject` to get the full object of a package for inclusion in an action

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.003270
- Step 3 Request Body
- Step 3 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "581",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6925-6c9d7f096a4dc99b8e34bb9d09d699963f8235f2534720facac0830700c5f8799b79fa5d2d9e828"
9 }
```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }
```

### Step 4 - Issue an `AddObject` to add a list of `SavedActions` (6.5 logic)

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.004854
- Step 4 Request Body
- Step 4 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "1493",
6   "Content-Type": "text/xml; charset=utf-8",
```

```

7  "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8  "session": "1-6925-6c9d7f096a4dc99b8e34bb9d09d699963f8235f2534720facac0830700c5f8799b79fa5d2d9e828"
9  }

```

- Response Headers:

```

1  {
2  "connection": "keep-alive",
3  "content-encoding": "gzip",
4  "content-type": "text/xml; charset=UTF-8",
5  "transfer-encoding": "chunked"
6  }

```

#### Step 5 - Issue a GetObject on the recently added object in order to get the full object

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.003599
- Step 5 Request Body
- Step 5 Response Body
- Request Headers:

```

1  {
2  "Accept": "*/*",
3  "Accept-Encoding": "gzip",
4  "Connection": "keep-alive",
5  "Content-Length": "1523",
6  "Content-Type": "text/xml; charset=utf-8",
7  "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8  "session": "1-6925-6c9d7f096a4dc99b8e34bb9d09d699963f8235f2534720facac0830700c5f8799b79fa5d2d9e828"
9  }

```

- Response Headers:

```

1  {
2  "connection": "keep-alive",
3  "content-encoding": "gzip",
4  "content-type": "text/xml; charset=UTF-8",
5  "transfer-encoding": "chunked"
6  }

```

#### Step 6 - Issue a GetObject to get the last action created for a SavedAction

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.002473
- Step 6 Request Body
- Step 6 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "557",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6925-6c9d7f096a4dc99b8e34bb9d09d699963f8235f2534720facac0830700c5f8799b79fa5d2d9e828"
9 }
```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }
```

### Step 7 - Issue a GetObject to get the package for an Action

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.001965
- Step 7 Request Body
- Step 7 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "600",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6925-6c9d7f096a4dc99b8e34bb9d09d699963f8235f2534720facac0830700c5f8799b79fa5d2d9e828"
9 }
```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }
```

### Step 8 - Issue a GetResultInfo on an Action to have the Server create a question that tracks the results for a Deployed Action

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST

- Elapsed Time: 0:00:00.003232
- Step 8 Request Body
- Step 8 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "552",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6925-6c9d7f096a4dc99b8e34bb9d09d699963f8235f2534720facac0830700c5f8799b79fa5d2d9e828c"
9 }

```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }

```

### Step 9 - Issue a GetObject on the package for an action to get the full object

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.002084
- Step 9 Request Body
- Step 9 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "625",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6925-6c9d7f096a4dc99b8e34bb9d09d699963f8235f2534720facac0830700c5f8799b79fa5d2d9e828c"
9 }

```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }

```

## Deploy Action Simple Against Windows Computers

Deploy an action using the package ‘Distribute Tanium Standard Utilities’ to all computers that pass the filter Operating System, that contains Windows, wait for result data to be complete, and then get result data

### Step 1 - Authenticate to the SOAP API via /auth

- URL: <https://10.0.1.240:443/auth>
- HTTP Method: GET
- Elapsed Time: 0:00:00.005802
- Step 1 Request Body
- Step 1 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip, deflate",
4   "Connection": "keep-alive",
5   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
6   "password": "VGfuaXVtMjAxNSE=",
7   "username": "QWRtaW5pc3RyYXRvcg=="
8 }
```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-length": "135",
4   "content-type": "text/plain; charset=us-ascii"
5 }
```

### Step 2 - Get the server version via /info.json

- URL: <https://10.0.1.240:443/info.json>
- HTTP Method: GET
- Elapsed Time: 0:00:00.013419
- Step 2 Request Body
- Step 2 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip, deflate",
4   "Connection": "keep-alive",
5   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
6   "session": "1-6926-7c9f86afdc2c2f45e9ecbf221adbe2421485042a7374e8aeeb4f1784b59a9d7871e3e7f03dd101e"
7 }
```

- Response Headers:



```

1 {
2   "connection": "keep-alive",
3   "content-length": "86181",
4   "content-type": "application/json"
5 }

```

### Step 3 - Issue a GetObject to get the full object of a package for inclusion in an action

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.001934
- Step 3 Request Body
- Step 3 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "581",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6926-7c9f86afdc2c2f45e9ecbf221adbe2421485042a7374e8aeeb4f1784b59a9d7871e3e7f03dd101e"
9 }

```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }

```

### Step 4 - Issue a GetObject to get the full object of a sensor for inclusion in a Group for an Action

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.001778
- Step 4 Request Body
- Step 4 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "568",
6   "Content-Type": "text/xml; charset=utf-8",

```

```
7 "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8 "session": "1-6926-7c9f86afdc2c2f45e9ecbf221adbe2421485042a7374e8aeeb4f1784b59a9d7871e3e7f03dd101e
9 }
```

- Response Headers:

```
1 {
2 "connection": "keep-alive",
3 "content-encoding": "gzip",
4 "content-type": "text/xml; charset=UTF-8",
5 "transfer-encoding": "chunked"
6 }
```

#### Step 5 - Issue an AddObject to add a list of SavedActions (6.5 logic)

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.008466
- Step 5 Request Body
- Step 5 Response Body
- Request Headers:

```
1 {
2 "Accept": "*/*",
3 "Accept-Encoding": "gzip",
4 "Connection": "keep-alive",
5 "Content-Length": "1675",
6 "Content-Type": "text/xml; charset=utf-8",
7 "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8 "session": "1-6926-7c9f86afdc2c2f45e9ecbf221adbe2421485042a7374e8aeeb4f1784b59a9d7871e3e7f03dd101e
9 }
```

- Response Headers:

```
1 {
2 "connection": "keep-alive",
3 "content-encoding": "gzip",
4 "content-type": "text/xml; charset=UTF-8",
5 "transfer-encoding": "chunked"
6 }
```

#### Step 6 - Issue a GetObject on the recently added object in order to get the full object

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.003469
- Step 6 Request Body
- Step 6 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "1526",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6926-7c9f86afdc2c2f45e9ecbf221adbe2421485042a7374e8aeeb4f1784b59a9d7871e3e7f03dd101e"
9 }

```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }

```

#### Step 7 - Issue a GetObject to get the last action created for a SavedAction

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.002401
- Step 7 Request Body
- Step 7 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "560",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6926-7c9f86afdc2c2f45e9ecbf221adbe2421485042a7374e8aeeb4f1784b59a9d7871e3e7f03dd101e"
9 }

```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }

```

#### Step 8 - Issue a GetObject to get the package for an Action

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.001595

- Step 8 Request Body
- Step 8 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "600",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6926-7c9f86afdc2c2f45e9ecbf221adbe2421485042a7374e8aeeb4f1784b59a9d7871e3e7f03dd101e0",
9 }
```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }
```

#### Step 9 - Issue a GetResultInfo on an Action to have the Server create a question that tracks the results for a Deployed Action

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.002773
- Step 9 Request Body
- Step 9 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "552",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6926-7c9f86afdc2c2f45e9ecbf221adbe2421485042a7374e8aeeb4f1784b59a9d7871e3e7f03dd101e0",
9 }
```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }
```

**Step 10 - Issue a GetObject on the package for an action to get the full object**

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.001501
- Step 10 Request Body
- Step 10 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "625",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6926-7c9f86afdc2c2f45e9ecbf221adbe2421485042a7374e8aeeb4f1784b59a9d7871e3e7f03dd101e"
9 }

```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }

```

**Step 11 - Issue a GetObject on the target\_group for an action to get the full Group object**

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.010846
- Step 11 Request Body
- Step 11 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "507",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6926-7c9f86afdc2c2f45e9ecbf221adbe2421485042a7374e8aeeb4f1784b59a9d7871e3e7f03dd101e"
9 }

```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }
```

**Step 12 - ID 524: Issuing an AddObject of a Question object with no Selects and the same Group used by the Action object. The number of systems that should successfully run the Action will be taken from result\_info.passed\_count for the Question asked when all answers for the question have reported in.**

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.005671
- Step 12 Request Body
- Step 12 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "1144",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6926-7c9f86afdc2c2f45e9ecbf221adbe2421485042a7374e8aeeb4f1784b59a9d7871e3e7f03dd101e0"
9 }
```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-length": "769",
4   "content-type": "text/xml; charset=UTF-8"
5 }
```

**Step 13 - Issue a GetObject on the recently added object in order to get the full object**

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.011850
- Step 13 Request Body
- Step 13 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
```

```

5  "Content-Length": "494",
6  "Content-Type": "text/xml; charset=utf-8",
7  "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8  "session": "1-6926-7c9f86afdc2c2f45e9ecbf221adbe2421485042a7374e8aeeb4f1784b59a9d7871e3e7f03dd101e"
9  }

```

- Response Headers:

```

1  {
2  "connection": "keep-alive",
3  "content-encoding": "gzip",
4  "content-type": "text/xml; charset=UTF-8",
5  "transfer-encoding": "chunked"
6  }

```

#### Step 14 - Issue a GetResultInfo for a Question to check the current progress of answers

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.001651
- Step 14 Request Body
- Step 14 Response Body
- Request Headers:

```

1  {
2  "Accept": "*/*",
3  "Accept-Encoding": "gzip",
4  "Connection": "keep-alive",
5  "Content-Length": "498",
6  "Content-Type": "text/xml; charset=utf-8",
7  "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8  "session": "1-6926-7c9f86afdc2c2f45e9ecbf221adbe2421485042a7374e8aeeb4f1784b59a9d7871e3e7f03dd101e"
9  }

```

- Response Headers:

```

1  {
2  "connection": "keep-alive",
3  "content-encoding": "gzip",
4  "content-type": "text/xml; charset=UTF-8",
5  "transfer-encoding": "chunked"
6  }

```

#### Step 15 - Issue a GetResultInfo for a Question to check the current progress of answers

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.001431
- Step 15 Request Body
- Step 15 Response Body

- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "498",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6926-7c9f86afdc2c2f45e9ecbf221adbe2421485042a7374e8aeeb4f1784b59a9d7871e3e7f03dd101e",
9 }
```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }
```

#### Step 16 - Issue a GetResultInfo for a Question to check the current progress of answers

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.008313
- Step 16 Request Body
- Step 16 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "498",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6926-7c9f86afdc2c2f45e9ecbf221adbe2421485042a7374e8aeeb4f1784b59a9d7871e3e7f03dd101e",
9 }
```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }
```

#### Step 17 - Issue a GetObject for an Action in order to have access to the latest values for stopped\_flag and status

- URL: <https://10.0.1.240:443/soap>



- HTTP Method: POST
- Elapsed Time: 0:00:00.009190
- Step 17 Request Body
- Step 17 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "1463",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6926-7c9f86afdc2c2f45e9ecbf221adbe2421485042a7374e8aeeb4f1784b59a9d7871e3e7f03dd101e"
9 }

```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }

```

#### Step 18 - Issue a GetResultInfo for an Action to ensure fresh data is available for a GetResultData call

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.003839
- Step 18 Request Body
- Step 18 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "552",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6926-7c9f86afdc2c2f45e9ecbf221adbe2421485042a7374e8aeeb4f1784b59a9d7871e3e7f03dd101e"
9 }

```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }

```

**Step 19 - Issue a GetResultData with the aggregate option set to True.**This will return row counts of machines that have answered instead of all the data

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.093405
- Step 19 Request Body
- Step 19 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "626",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6926-7c9f86afdc2c2f45e9ecbf221adbe2421485042a7374e8aeeb4f1784b59a9d7871e3e7f03dd101e"
9 }
```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }
```

**Step 20 - Issue a GetObject for an Action in order to have access to the latest values for stopped\_flag and status**

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.003132
- Step 20 Request Body
- Step 20 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "1463",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6926-7c9f86afdc2c2f45e9ecbf221adbe2421485042a7374e8aeeb4f1784b59a9d7871e3e7f03dd101e"
9 }
```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }

```

#### Step 21 - Issue a GetResultInfo for an Action to ensure fresh data is available for a GetResultData call

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.003049
- Step 21 Request Body
- Step 21 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "552",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6926-7c9f86afdc2c2f45e9ecbf221adbe2421485042a7374e8aeeb4f1784b59a9d7871e3e7f03dd101e"
9 }

```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }

```

#### Step 22 - Issue a GetResultData with the aggregate option set to True. This will return row counts of machines that have answered instead of all the data

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.002825
- Step 22 Request Body
- Step 22 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",

```

```
5  "Content-Length": "626",
6  "Content-Type": "text/xml; charset=utf-8",
7  "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8  "session": "1-6926-7c9f86afdc2c2f45e9ecbf221adbe2421485042a7374e8aeeb4f1784b59a9d7871e3e7f03dd101e"
9  }
```

- Response Headers:

```
1  {
2  "connection": "keep-alive",
3  "content-encoding": "gzip",
4  "content-type": "text/xml; charset=UTF-8",
5  "transfer-encoding": "chunked"
6  }
```

### Step 23 - Issue a `GetObject` for an Action in order to have access to the latest values for `stopped_flag` and `status`

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.003021
- Step 23 Request Body
- Step 23 Response Body
- Request Headers:

```
1  {
2  "Accept": "*/*",
3  "Accept-Encoding": "gzip",
4  "Connection": "keep-alive",
5  "Content-Length": "1463",
6  "Content-Type": "text/xml; charset=utf-8",
7  "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8  "session": "1-6926-7c9f86afdc2c2f45e9ecbf221adbe2421485042a7374e8aeeb4f1784b59a9d7871e3e7f03dd101e"
9  }
```

- Response Headers:

```
1  {
2  "connection": "keep-alive",
3  "content-encoding": "gzip",
4  "content-type": "text/xml; charset=UTF-8",
5  "transfer-encoding": "chunked"
6  }
```

### Step 24 - Issue a `GetResultInfo` for an Action to ensure fresh data is available for a `GetResultData` call

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.002721
- Step 24 Request Body

- Step 24 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "552",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6926-7c9f86afdc2c2f45e9ecbf221adbe2421485042a7374e8aeeb4f1784b59a9d7871e3e7f03dd101e0"
9 }

```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }

```

**Step 25 - Issue a GetResultData with the aggregate option set to True.** This will return row counts of machines that have answered instead of all the data

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.002548
- Step 25 Request Body
- Step 25 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "626",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6926-7c9f86afdc2c2f45e9ecbf221adbe2421485042a7374e8aeeb4f1784b59a9d7871e3e7f03dd101e0"
9 }

```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }

```

**Step 26 - Issue a GetObject for an Action in order to have access to the latest values for stopped\_flag and status**

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.002953
- Step 26 Request Body
- Step 26 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "1463",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6926-7c9f86afdc2c2f45e9ecbf221adbe2421485042a7374e8aeeb4f1784b59a9d7871e3e7f03dd101e"
9 }
```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }
```

**Step 27 - Issue a GetResultInfo for an Action to ensure fresh data is available for a GetResultData call**

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.002760
- Step 27 Request Body
- Step 27 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "552",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6926-7c9f86afdc2c2f45e9ecbf221adbe2421485042a7374e8aeeb4f1784b59a9d7871e3e7f03dd101e"
9 }
```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }

```

**Step 28 - Issue a GetResultData with the aggregate option set to True.** This will return row counts of machines that have answered instead of all the data

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.002531
- Step 28 Request Body
- Step 28 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "626",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6926-7c9f86afdc2c2f45e9ecbf221adbe2421485042a7374e8aeeb4f1784b59a9d7871e3e7f03dd101e"
9 }

```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }

```

**Step 29 - Issue a GetObject for an Action in order to have access to the latest values for stopped\_flag and status**

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.002198
- Step 29 Request Body
- Step 29 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",

```

```
4 "Connection": "keep-alive",
5 "Content-Length": "1463",
6 "Content-Type": "text/xml; charset=utf-8",
7 "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8 "session": "1-6926-7c9f86afdc2c2f45e9ecbf221adbe2421485042a7374e8aeeb4f1784b59a9d7871e3e7f03dd101e"
9 }
```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }
```

**Step 30 - Issue a GetResultInfo for an Action to ensure fresh data is available for a GetResultData call**

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.002624
- Step 30 Request Body
- Step 30 Response Body
- Request Headers:

```
1 {
2     "Accept": "*/*",
3     "Accept-Encoding": "gzip",
4     "Connection": "keep-alive",
5     "Content-Length": "552",
6     "Content-Type": "text/xml; charset=utf-8",
7     "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8     "session": "1-6926-7c9f86afdc2c2f45e9ecbf221adbe2421485042a7374e8aeeb4f1784b59a9d7871e3e7f03dd101e0"
9 }
```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }
```

**Step 31 - Issue a GetResultData for an Action with the aggregate option set to False. This will return all of the Action Statuses for each computer that have run this Action**

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.002631
- Step 31 Request Body



- Step 31 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "580",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6926-7c9f86afdc2c2f45e9ecbf221adbe2421485042a7374e8aeeb4f1784b59a9d7871e3e7f03dd101e"
9 }

```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }

```

## Deploy Action With Params Against Windows Computers

Deploy an action using the package 'Custom Tagging - Add Tags' with parameter \$1 set to 'tag\_should\_be\_added' and parameter \$2 set to 'tag\_should\_be\_ignore' to all computers that pass the filter Operating System, that contains Windows, wait for result data to be complete, and then get result data

### Step 1 - Authenticate to the SOAP API via /auth

- URL: <https://10.0.1.240:443/auth>
- HTTP Method: GET
- Elapsed Time: 0:00:00.006693
- Step 1 Request Body
- Step 1 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip, deflate",
4   "Connection": "keep-alive",
5   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
6   "password": "VGFuaXVtMjAxNSE=",
7   "username": "QWRtaW5pc3RyYXRvcg=="
8 }

```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-length": "135",
4   "content-type": "text/plain; charset=us-ascii"
5 }

```

## Step 2 - Get the server version via /info.json

- URL: <https://10.0.1.240:443/info.json>
- HTTP Method: GET
- Elapsed Time: 0:00:00.013755
- Step 2 Request Body
- Step 2 Response Body
- Request Headers:

```
1 {  
2   "Accept": "*/*",  
3   "Accept-Encoding": "gzip, deflate",  
4   "Connection": "keep-alive",  
5   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",  
6   "session": "1-6927-2aa140382c7dcbeb4eab3476ec0021626f7fa2a6ac87f3ba55e7ab26add2b77a0461b7247e2ae3e",  
7 }
```

- Response Headers:

```
1 {  
2   "connection": "keep-alive",  
3   "content-length": "86181",  
4   "content-type": "application/json"  
5 }
```

## Step 3 - Issue a GetObject to get the full object of a package for inclusion in an action

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.002488
- Step 3 Request Body
- Step 3 Response Body
- Request Headers:

```
1 {  
2   "Accept": "*/*",  
3   "Accept-Encoding": "gzip",  
4   "Connection": "keep-alive",  
5   "Content-Length": "570",  
6   "Content-Type": "text/xml; charset=utf-8",  
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",  
8   "session": "1-6927-2aa140382c7dcbeb4eab3476ec0021626f7fa2a6ac87f3ba55e7ab26add2b77a0461b7247e2ae3e",  
9 }
```

- Response Headers:

```
1 {  
2   "connection": "keep-alive",  
3   "content-encoding": "gzip",  
4   "content-type": "text/xml; charset=UTF-8",
```

```

5  "transfer-encoding": "chunked"
6  }

```

#### Step 4 - Issue a GetObject to get the full object of a sensor for inclusion in a Group for an Action

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.001799
- Step 4 Request Body
- Step 4 Response Body
- Request Headers:

```

1  {
2  "Accept": "*/*",
3  "Accept-Encoding": "gzip",
4  "Connection": "keep-alive",
5  "Content-Length": "568",
6  "Content-Type": "text/xml; charset=utf-8",
7  "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8  "session": "1-6927-2aa140382c7dcbeb4eab3476ec0021626f7fa2a6ac87f3ba55e7ab26add2b77a0461b7247e2ae3e"
9  }

```

- Response Headers:

```

1  {
2  "connection": "keep-alive",
3  "content-encoding": "gzip",
4  "content-type": "text/xml; charset=UTF-8",
5  "transfer-encoding": "chunked"
6  }

```

#### Step 5 - Issue an AddObject to add a list of SavedActions (6.5 logic)

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.014973
- Step 5 Request Body
- Step 5 Response Body
- Request Headers:

```

1  {
2  "Accept": "*/*",
3  "Accept-Encoding": "gzip",
4  "Connection": "keep-alive",
5  "Content-Length": "2694",
6  "Content-Type": "text/xml; charset=utf-8",
7  "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8  "session": "1-6927-2aa140382c7dcbeb4eab3476ec0021626f7fa2a6ac87f3ba55e7ab26add2b77a0461b7247e2ae3e"
9  }

```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }
```

#### Step 6 - Issue a GetObject on the recently added object in order to get the full object

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.003445
- Step 6 Request Body
- Step 6 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "1452",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6927-2aa140382c7dcbeb4eab3476ec0021626f7fa2a6ac87f3ba55e7ab26add2b77a0461b7247e2ae3e"
9 }
```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }
```

#### Step 7 - Issue a GetObject to get the last action created for a SavedAction

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.002051
- Step 7 Request Body
- Step 7 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
```

```

5  "Content-Length": "560",
6  "Content-Type": "text/xml; charset=utf-8",
7  "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8  "session": "1-6927-2aa140382c7dcbeb4eab3476ec0021626f7fa2a6ac87f3ba55e7ab26add2b77a0461b7247e2ae3e"
9  }

```

- Response Headers:

```

1  {
2  "connection": "keep-alive",
3  "content-encoding": "gzip",
4  "content-type": "text/xml; charset=UTF-8",
5  "transfer-encoding": "chunked"
6  }

```

### Step 8 - Issue a GetObject to get the package for an Action

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.001578
- Step 8 Request Body
- Step 8 Response Body
- Request Headers:

```

1  {
2  "Accept": "*/*",
3  "Accept-Encoding": "gzip",
4  "Connection": "keep-alive",
5  "Content-Length": "538",
6  "Content-Type": "text/xml; charset=utf-8",
7  "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8  "session": "1-6927-2aa140382c7dcbeb4eab3476ec0021626f7fa2a6ac87f3ba55e7ab26add2b77a0461b7247e2ae3e"
9  }

```

- Response Headers:

```

1  {
2  "connection": "keep-alive",
3  "content-encoding": "gzip",
4  "content-type": "text/xml; charset=UTF-8",
5  "transfer-encoding": "chunked"
6  }

```

### Step 9 - Issue a GetResultInfo on an Action to have the Server create a question that tracks the results for a Deployed Action

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.002784
- Step 9 Request Body

- Step 9 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "541",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6927-2aa140382c7dcbeb4eab3476ec0021626f7fa2a6ac87f3ba55e7ab26add2b77a0461b7247e2ae3e"
9 }
```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }
```

#### Step 10 - Issue a GetObject on the package for an action to get the full object

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.001699
- Step 10 Request Body
- Step 10 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "619",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6927-2aa140382c7dcbeb4eab3476ec0021626f7fa2a6ac87f3ba55e7ab26add2b77a0461b7247e2ae3e"
9 }
```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }
```

#### Step 11 - Issue a GetObject on the target\_group for an action to get the full Group object

- URL: <https://10.0.1.240:443/soap>

- HTTP Method: POST
- Elapsed Time: 0:00:00.010375
- Step 11 Request Body
- Step 11 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "507",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6927-2aa140382c7dcbeb4eab3476ec0021626f7fa2a6ac87f3ba55e7ab26add2b77a0461b7247e2ae3e"
9 }
```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }
```

**Step 12 - ID 525: Issuing an AddObject of a Question object with no Selects and the same Group used by the Action object. The number of systems that should successfully run the Action will be taken from result\_info.passed\_count for the Question asked when all answers for the question have reported in.**

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.005768
- Step 12 Request Body
- Step 12 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "1144",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6927-2aa140382c7dcbeb4eab3476ec0021626f7fa2a6ac87f3ba55e7ab26add2b77a0461b7247e2ae3e"
9 }
```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-length": "769",
4   "content-type": "text/xml; charset=UTF-8"
5 }
```

**Step 13 - Issue a GetObject on the recently added object in order to get the full object**

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.011880
- Step 13 Request Body
- Step 13 Response Body
- Request Headers:

```
1 {  
2   "Accept": "*/*",  
3   "Accept-Encoding": "gzip",  
4   "Connection": "keep-alive",  
5   "Content-Length": "494",  
6   "Content-Type": "text/xml; charset=utf-8",  
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",  
8   "session": "1-6927-2aa140382c7dcbeb4eab3476ec0021626f7fa2a6ac87f3ba55e7ab26add2b77a0461b7247e2ae3e"  
9 }
```

- Response Headers:

```
1 {  
2   "connection": "keep-alive",  
3   "content-encoding": "gzip",  
4   "content-type": "text/xml; charset=UTF-8",  
5   "transfer-encoding": "chunked"  
6 }
```

**Step 14 - Issue a GetResultInfo for a Question to check the current progress of answers**

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.001525
- Step 14 Request Body
- Step 14 Response Body
- Request Headers:

```
1 {  
2   "Accept": "*/*",  
3   "Accept-Encoding": "gzip",  
4   "Connection": "keep-alive",  
5   "Content-Length": "498",  
6   "Content-Type": "text/xml; charset=utf-8",  
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",  
8   "session": "1-6927-2aa140382c7dcbeb4eab3476ec0021626f7fa2a6ac87f3ba55e7ab26add2b77a0461b7247e2ae3e"  
9 }
```

- Response Headers:



```

1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }

```

#### Step 15 - Issue a GetResultInfo for a Question to check the current progress of answers

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.001990
- Step 15 Request Body
- Step 15 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "498",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6927-2aa140382c7dcbeb4eab3476ec0021626f7fa2a6ac87f3ba55e7ab26add2b77a0461b7247e2ae3e"
9 }

```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }

```

#### Step 16 - Issue a GetResultInfo for a Question to check the current progress of answers

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.002057
- Step 16 Request Body
- Step 16 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "498",

```

```
6  "Content-Type": "text/xml; charset=utf-8",
7  "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8  "session": "1-6927-2aa140382c7dcbeb4eab3476ec0021626f7fa2a6ac87f3ba55e7ab26add2b77a0461b7247e2ae3e"
9 }
```

- Response Headers:

```
1 {
2  "connection": "keep-alive",
3  "content-encoding": "gzip",
4  "content-type": "text/xml; charset=UTF-8",
5  "transfer-encoding": "chunked"
6 }
```

#### Step 17 - Issue a `GetResultInfo` for a Question to check the current progress of answers

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.001479
- Step 17 Request Body
- Step 17 Response Body
- Request Headers:

```
1 {
2  "Accept": "*/*",
3  "Accept-Encoding": "gzip",
4  "Connection": "keep-alive",
5  "Content-Length": "498",
6  "Content-Type": "text/xml; charset=utf-8",
7  "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8  "session": "1-6927-2aa140382c7dcbeb4eab3476ec0021626f7fa2a6ac87f3ba55e7ab26add2b77a0461b7247e2ae3e"
9 }
```

- Response Headers:

```
1 {
2  "connection": "keep-alive",
3  "content-encoding": "gzip",
4  "content-type": "text/xml; charset=UTF-8",
5  "transfer-encoding": "chunked"
6 }
```

#### Step 18 - Issue a `GetObject` for an Action in order to have access to the latest values for `stopped_flag` and `status`

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.002301
- Step 18 Request Body
- Step 18 Response Body

- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "1445",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6927-2aa140382c7dcbeb4eab3476ec0021626f7fa2a6ac87f3ba55e7ab26add2b77a0461b7247e2ae3e"
9 }

```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }

```

#### Step 19 - Issue a GetResultInfo for an Action to ensure fresh data is available for a GetResultData call

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.002625
- Step 19 Request Body
- Step 19 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "541",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6927-2aa140382c7dcbeb4eab3476ec0021626f7fa2a6ac87f3ba55e7ab26add2b77a0461b7247e2ae3e"
9 }

```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }

```

#### Step 20 - Issue a GetResultData with the aggregate option set to True. This will return row counts of machines that have answered instead of all the data

- URL: <https://10.0.1.240:443/soap>

- HTTP Method: POST
- Elapsed Time: 0:00:00.002315
- Step 20 Request Body
- Step 20 Response Body
- Request Headers:

```
1 {  
2   "Accept": "*/*",  
3   "Accept-Encoding": "gzip",  
4   "Connection": "keep-alive",  
5   "Content-Length": "615",  
6   "Content-Type": "text/xml; charset=utf-8",  
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",  
8   "session": "1-6927-2aa140382c7dcbeb4eab3476ec0021626f7fa2a6ac87f3ba55e7ab26add2b77a0461b7247e2ae3e",  
9 }
```

- Response Headers:

```
1 {  
2   "connection": "keep-alive",  
3   "content-encoding": "gzip",  
4   "content-type": "text/xml; charset=UTF-8",  
5   "transfer-encoding": "chunked"  
6 }
```

#### Step 21 - Issue a GetObject for an Action in order to have access to the latest values for stopped\_flag and status

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.002761
- Step 21 Request Body
- Step 21 Response Body
- Request Headers:

```
1 {  
2   "Accept": "*/*",  
3   "Accept-Encoding": "gzip",  
4   "Connection": "keep-alive",  
5   "Content-Length": "1445",  
6   "Content-Type": "text/xml; charset=utf-8",  
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",  
8   "session": "1-6927-2aa140382c7dcbeb4eab3476ec0021626f7fa2a6ac87f3ba55e7ab26add2b77a0461b7247e2ae3e",  
9 }
```

- Response Headers:

```
1 {  
2   "connection": "keep-alive",  
3   "content-encoding": "gzip",  
4   "content-type": "text/xml; charset=UTF-8",  
5   "transfer-encoding": "chunked"  
6 }
```

**Step 22 - Issue a GetResultInfo for an Action to ensure fresh data is available for a GetResultData call**

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.002758
- Step 22 Request Body
- Step 22 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "541",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6927-2aa140382c7dcbeb4eab3476ec0021626f7fa2a6ac87f3ba55e7ab26add2b77a0461b7247e2ae3e"
9 }

```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }

```

**Step 23 - Issue a GetResultData with the aggregate option set to True. This will return row counts of machines that have answered instead of all the data**

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.002489
- Step 23 Request Body
- Step 23 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "615",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6927-2aa140382c7dcbeb4eab3476ec0021626f7fa2a6ac87f3ba55e7ab26add2b77a0461b7247e2ae3e"
9 }

```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }
```

**Step 24 - Issue a GetObject for an Action in order to have access to the latest values for stopped\_flag and status**

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.002660
- Step 24 Request Body
- Step 24 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "1445",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6927-2aa140382c7dcbeb4eab3476ec0021626f7fa2a6ac87f3ba55e7ab26add2b77a0461b7247e2ae3e"
9 }
```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }
```

**Step 25 - Issue a GetResultInfo for an Action to ensure fresh data is available for a GetResultData call**

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.003299
- Step 25 Request Body
- Step 25 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
```

```

5  "Content-Length": "541",
6  "Content-Type": "text/xml; charset=utf-8",
7  "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8  "session": "1-6927-2aa140382c7dcbeb4eab3476ec0021626f7fa2a6ac87f3ba55e7ab26add2b77a0461b7247e2ae3e"
9  }

```

- Response Headers:

```

1  {
2  "connection": "keep-alive",
3  "content-encoding": "gzip",
4  "content-type": "text/xml; charset=UTF-8",
5  "transfer-encoding": "chunked"
6  }

```

**Step 26 - Issue a GetResultData with the aggregate option set to True.**This will return row counts of machines that have answered instead of all the data

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.002618
- Step 26 Request Body
- Step 26 Response Body
- Request Headers:

```

1  {
2  "Accept": "*/*",
3  "Accept-Encoding": "gzip",
4  "Connection": "keep-alive",
5  "Content-Length": "615",
6  "Content-Type": "text/xml; charset=utf-8",
7  "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8  "session": "1-6927-2aa140382c7dcbeb4eab3476ec0021626f7fa2a6ac87f3ba55e7ab26add2b77a0461b7247e2ae3e"
9  }

```

- Response Headers:

```

1  {
2  "connection": "keep-alive",
3  "content-encoding": "gzip",
4  "content-type": "text/xml; charset=UTF-8",
5  "transfer-encoding": "chunked"
6  }

```

**Step 27 - Issue a GetObject for an Action in order to have access to the latest values for stopped\_flag and status**

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.002263
- Step 27 Request Body

- Step 27 Response Body
- Request Headers:

```
1 {  
2   "Accept": "*/*",  
3   "Accept-Encoding": "gzip",  
4   "Connection": "keep-alive",  
5   "Content-Length": "1445",  
6   "Content-Type": "text/xml; charset=utf-8",  
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",  
8   "session": "1-6927-2aa140382c7dcbeb4eab3476ec0021626f7fa2a6ac87f3ba55e7ab26add2b77a0461b7247e2ae3e",  
9 }
```

- Response Headers:

```
1 {  
2   "connection": "keep-alive",  
3   "content-encoding": "gzip",  
4   "content-type": "text/xml; charset=UTF-8",  
5   "transfer-encoding": "chunked"  
6 }
```

#### Step 28 - Issue a `GetResultInfo` for an Action to ensure fresh data is available for a `GetResultData` call

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.002522
- Step 28 Request Body
- Step 28 Response Body
- Request Headers:

```
1 {  
2   "Accept": "*/*",  
3   "Accept-Encoding": "gzip",  
4   "Connection": "keep-alive",  
5   "Content-Length": "541",  
6   "Content-Type": "text/xml; charset=utf-8",  
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",  
8   "session": "1-6927-2aa140382c7dcbeb4eab3476ec0021626f7fa2a6ac87f3ba55e7ab26add2b77a0461b7247e2ae3e",  
9 }
```

- Response Headers:

```
1 {  
2   "connection": "keep-alive",  
3   "content-encoding": "gzip",  
4   "content-type": "text/xml; charset=UTF-8",  
5   "transfer-encoding": "chunked"  
6 }
```



**Step 29 - Issue a GetResultData for an Action with the aggregate option set to False. This will return all of the Action Statuses for each computer that have run this Action**

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.002982
- Step 29 Request Body
- Step 29 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "569",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6927-2aa140382c7dcb4eab3476ec0021626f7fa2a6ac87f3ba55e7ab26add2b77a0461b7247e2ae3e"
9 }

```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }

```

## Get Action By Id

Get an action object by id

### Step 1 - Authenticate to the SOAP API via /auth

- URL: <https://10.0.1.240:443/auth>
- HTTP Method: GET
- Elapsed Time: 0:00:00.008103
- Step 1 Request Body
- Step 1 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip, deflate",
4   "Connection": "keep-alive",
5   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
6   "password": "VGfuaXVtMjAxNSE=",
7   "username": "QWRtaW5pc3RyYXRvcg=="
8 }

```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-length": "135",
4   "content-type": "text/plain; charset=us-ascii"
5 }
```

## Step 2 - Get the server version via /info.json

- URL: <https://10.0.1.240:443/info.json>
- HTTP Method: GET
- Elapsed Time: 0:00:00.012553
- Step 2 Request Body
- Step 2 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip, deflate",
4   "Connection": "keep-alive",
5   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
6   "session": "1-6955-1ecb5e037fa01e1d9a4f45f8fbff3e57f8dba89d70dacf4bd0a59848f85391bc00f9370a08b9748"
7 }
```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-length": "87505",
4   "content-type": "application/json"
5 }
```

## Step 3 - Issue a GetObject to find an object

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.002293
- Step 3 Request Body
- Step 3 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "486",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
```

```

8  "session": "1-6955-1ecb5e037fa01e1d9a4f45f8fbff3e57f8dba89d70dacf4bd0a59848f85391bc00f9370a08b9748c
9  }

```

- Response Headers:

```

1  {
2  "connection": "keep-alive",
3  "content-encoding": "gzip",
4  "content-type": "text/xml; charset=UTF-8",
5  "transfer-encoding": "chunked"
6  }

```

## Get Question By Id

Get a question object by id

### Step 1 - Authenticate to the SOAP API via /auth

- URL: <https://10.0.1.240:443/auth>
- HTTP Method: GET
- Elapsed Time: 0:00:00.006206
- Step 1 Request Body
- Step 1 Response Body
- Request Headers:

```

1  {
2  "Accept": "*/*",
3  "Accept-Encoding": "gzip, deflate",
4  "Connection": "keep-alive",
5  "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
6  "password": "VGfuaXVtMjAxNSE=",
7  "username": "QWRtaW5pc3RyYXRvcg=="
8  }

```

- Response Headers:

```

1  {
2  "connection": "keep-alive",
3  "content-length": "135",
4  "content-type": "text/plain; charset=us-ascii"
5  }

```

### Step 2 - Get the server version via /info.json

- URL: <https://10.0.1.240:443/info.json>
- HTTP Method: GET
- Elapsed Time: 0:00:00.006814
- Step 2 Request Body
- Step 2 Response Body

- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip, deflate",
4   "Connection": "keep-alive",
5   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
6   "session": "1-6956-97b438be2639ca6835344f0549d2f56bc4b585b29e59202a1eb3bc747a211ccd44d4964486c2b79"
7 }
```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-length": "87505",
4   "content-type": "application/json"
5 }
```

### Step 3 - Issue a GetObject to find an object

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.004008
- Step 3 Request Body
- Step 3 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "490",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6956-97b438be2639ca6835344f0549d2f56bc4b585b29e59202a1eb3bc747a211ccd44d4964486c2b79"
9 }
```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }
```

### Get Saved Question By Names

Get two saved question objects by name

#### Step 1 - Authenticate to the SOAP API via /auth

- URL: <https://10.0.1.240:443/auth>

- HTTP Method: GET
- Elapsed Time: 0:00:00.005978
- Step 1 Request Body
- Step 1 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip, deflate",
4   "Connection": "keep-alive",
5   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
6   "password": "VGfuaXVtMjAxNSE=",
7   "username": "QWRtaW5pc3RyYXRvcg=="
8 }
```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-length": "135",
4   "content-type": "text/plain; charset=us-ascii"
5 }
```

## Step 2 - Get the server version via /info.json

- URL: <https://10.0.1.240:443/info.json>
- HTTP Method: GET
- Elapsed Time: 0:00:00.013357
- Step 2 Request Body
- Step 2 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip, deflate",
4   "Connection": "keep-alive",
5   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
6   "session": "1-6957-2af3ae329097dec3d362d88a9d360653716b1835f171e5ec73f8bd72dc73e8a011f7c78e86081273"
7 }
```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-length": "87505",
4   "content-type": "application/json"
5 }
```

## Step 3 - Issue a GetObject to find an object

- URL: <https://10.0.1.240:443/soap>

- HTTP Method: POST
- Elapsed Time: 0:00:00.009865
- Step 3 Request Body
- Step 3 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "527",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6957-2af3ae329097dec3d362d88a9d360653716b1835f171e5ec73f8bd72dc73e8a011f7c78e86081273"
9 }
```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }
```

#### Step 4 - Issue a GetObject to find an object

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.004492
- Step 4 Request Body
- Step 4 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "518",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6957-2af3ae329097dec3d362d88a9d360653716b1835f171e5ec73f8bd72dc73e8a011f7c78e86081273"
9 }
```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }
```

## Get Userrole By Id

Get a user role object by id.

### Step 1 - Authenticate to the SOAP API via /auth

- URL: <https://10.0.1.240:443/auth>
- HTTP Method: GET
- Elapsed Time: 0:00:00.005994
- Step 1 Request Body
- Step 1 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip, deflate",
4   "Connection": "keep-alive",
5   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
6   "password": "VGfuaXVtMjAxNSE=",
7   "username": "QWRtaW5pc3RyYXRvcg=="
8 }
```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-length": "135",
4   "content-type": "text/plain; charset=us-ascii"
5 }
```

### Step 2 - Get the server version via /info.json

- URL: <https://10.0.1.240:443/info.json>
- HTTP Method: GET
- Elapsed Time: 0:00:00.006953
- Step 2 Request Body
- Step 2 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip, deflate",
4   "Connection": "keep-alive",
5   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
6   "session": "1-6958-2a673ff45a935ea342c03b91f5539bca0276815c34461c146e8b968c2c0af0b943df7e024ad1846"
7 }
```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-length": "87505",
4   "content-type": "application/json"
5 }
```

### Step 3 - Issue a GetObject to find an object

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.001628
- Step 3 Request Body
- Step 3 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "468",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6958-2a673ff45a935ea342c03b91f5539bca0276815c34461c146e8b968c2c0af0b943df7e024ad1846"
9 }
```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }
```

## Get Setting By Name

Get a system setting object by name

### Step 1 - Authenticate to the SOAP API via /auth

- URL: <https://10.0.1.240:443/auth>
- HTTP Method: GET
- Elapsed Time: 0:00:00.006244
- Step 1 Request Body
- Step 1 Response Body
- Request Headers:



```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip, deflate",
4   "Connection": "keep-alive",
5   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
6   "password": "VGFuaXVtMjAxNSE=",
7   "username": "QWRtaW5pc3RyYXRvcg=="
8 }

```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-length": "135",
4   "content-type": "text/plain; charset=us-ascii"
5 }

```

## Step 2 - Get the server version via /info.json

- URL: <https://10.0.1.240:443/info.json>
- HTTP Method: GET
- Elapsed Time: 0:00:00.007488
- Step 2 Request Body
- Step 2 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip, deflate",
4   "Connection": "keep-alive",
5   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
6   "session": "1-6960-11371e167410a3c3d27dcccab042b80e6663c4c924dd514fa147869596f6a38a56b9d2893c61815"
7 }

```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-length": "87506",
4   "content-type": "application/json"
5 }

```

## Step 3 - Issue a GetObject to find an object

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.001961
- Step 3 Request Body
- Step 3 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "555",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6960-11371e167410a3c3d27dcccab042b80e6663c4c924dd514fa147869596f6a38a56b9d2893c61815"
9 }
```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }
```

## Get User By Name

Get a user object by name

### Step 1 - Authenticate to the SOAP API via /auth

- URL: <https://10.0.1.240:443/auth>
- HTTP Method: GET
- Elapsed Time: 0:00:00.005801
- Step 1 Request Body
- Step 1 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip, deflate",
4   "Connection": "keep-alive",
5   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
6   "password": "VGfuaXVtMjAxNSE=",
7   "username": "QWRtaW5pc3RyYXRvcg=="
8 }
```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-length": "135",
4   "content-type": "text/plain; charset=us-ascii"
5 }
```

### Step 2 - Get the server version via /info.json

- URL: <https://10.0.1.240:443/info.json>

- HTTP Method: GET
- Elapsed Time: 0:00:00.006827
- Step 2 Request Body
- Step 2 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip, deflate",
4   "Connection": "keep-alive",
5   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
6   "session": "1-6961-a6c385d5e9bfee16b1c86d75340cfa4f213e626e46b600eb4433723c78872c2f705c9ca3df2ed8d"
7 }
```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-length": "87506",
4   "content-type": "application/json"
5 }
```

### Step 3 - Issue a GetObject to find an object

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.001992
- Step 3 Request Body
- Step 3 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "468",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6961-a6c385d5e9bfee16b1c86d75340cfa4f213e626e46b600eb4433723c78872c2f705c9ca3df2ed8d"
9 }
```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }
```

## Get Sensor By Id

Get a sensor object by id

### Step 1 - Authenticate to the SOAP API via /auth

- URL: <https://10.0.1.240:443/auth>
- HTTP Method: GET
- Elapsed Time: 0:00:00.006632
- Step 1 Request Body
- Step 1 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip, deflate",
4   "Connection": "keep-alive",
5   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
6   "password": "VGfuaXVtMjAxNSE=",
7   "username": "QWRtaW5pc3RyYXRvcg=="
8 }
```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-length": "135",
4   "content-type": "text/plain; charset=us-ascii"
5 }
```

### Step 2 - Get the server version via /info.json

- URL: <https://10.0.1.240:443/info.json>
- HTTP Method: GET
- Elapsed Time: 0:00:00.014478
- Step 2 Request Body
- Step 2 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip, deflate",
4   "Connection": "keep-alive",
5   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
6   "session": "1-6962-806e556663cea0b7b23615b0ff964cece8c6fa24d8938d887d43ee10f8e18358ea92408976181fe"
7 }
```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-length": "87507",
4   "content-type": "application/json"
5 }

```

### Step 3 - Issue a GetObject to find an object

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.001890
- Step 3 Request Body
- Step 3 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "505",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6962-806e556663cea0b7b23615b0ff964cece8c6fa24d8938d887d43ee10f8e18358ea92408976181fe"
9 }

```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }

```

## Get Sensor By Mixed

Get multiple sensor objects by id, name, and hash

### Step 1 - Authenticate to the SOAP API via /auth

- URL: <https://10.0.1.240:443/auth>
- HTTP Method: GET
- Elapsed Time: 0:00:00.006372
- Step 1 Request Body
- Step 1 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip, deflate",
4   "Connection": "keep-alive",
5   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
6   "password": "VGfuaXVtMjAxNSE=",
7   "username": "QWRtaW5pc3RyYXRvcg=="
8 }
```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-length": "135",
4   "content-type": "text/plain; charset=us-ascii"
5 }
```

## Step 2 - Get the server version via /info.json

- URL: <https://10.0.1.240:443/info.json>
- HTTP Method: GET
- Elapsed Time: 0:00:00.007601
- Step 2 Request Body
- Step 2 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip, deflate",
4   "Connection": "keep-alive",
5   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
6   "session": "1-6963-f44b617a66fdcecc597a2437863b9e76d3683d54c033c4bd505ea69c8b5ffec05d18e420e4b98f9"
7 }
```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-length": "87506",
4   "content-type": "application/json"
5 }
```

## Step 3 - Issue a GetObject to find an object

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.040141
- Step 3 Request Body
- Step 3 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "614",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6963-f44b617a66fdcecc597a2437863b9e76d3683d54c033c4bd505ea69c8b5ffec05d18e420e4b98f9"
9 }

```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }

```

## Get Whitelisted Url By Id

Get a whitelisted url object by id

### Step 1 - Authenticate to the SOAP API via /auth

- URL: <https://10.0.1.240:443/auth>
- HTTP Method: GET
- Elapsed Time: 0:00:00.005825
- Step 1 Request Body
- Step 1 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip, deflate",
4   "Connection": "keep-alive",
5   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
6   "password": "VGfuaXVtMjAxNSE=",
7   "username": "QWRtaW5pc3RyYXRvcg=="
8 }

```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-length": "135",
4   "content-type": "text/plain; charset=us-ascii"
5 }

```

### Step 2 - Get the server version via /info.json

- URL: <https://10.0.1.240:443/info.json>

- HTTP Method: GET
- Elapsed Time: 0:00:00.013926
- Step 2 Request Body
- Step 2 Response Body
- Request Headers:

```
1 {  
2   "Accept": "*/*",  
3   "Accept-Encoding": "gzip, deflate",  
4   "Connection": "keep-alive",  
5   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",  
6   "session": "1-6964-55a040c3652f545b45735d684822711158d05520869fc819ff66ac741b63e509269c89f3cbd1db9",  
7 }
```

- Response Headers:

```
1 {  
2   "connection": "keep-alive",  
3   "content-length": "87507",  
4   "content-type": "application/json"  
5 }
```

### Step 3 - Issue a GetObject to find an object

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.274445
- Step 3 Request Body
- Step 3 Response Body
- Request Headers:

```
1 {  
2   "Accept": "*/*",  
3   "Accept-Encoding": "gzip",  
4   "Connection": "keep-alive",  
5   "Content-Length": "480",  
6   "Content-Type": "text/xml; charset=utf-8",  
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",  
8   "session": "1-6964-55a040c3652f545b45735d684822711158d05520869fc819ff66ac741b63e509269c89f3cbd1db9",  
9 }
```

- Response Headers:

```
1 {  
2   "connection": "keep-alive",  
3   "content-encoding": "gzip",  
4   "content-type": "text/xml; charset=UTF-8",  
5   "transfer-encoding": "chunked"  
6 }
```



## Get Group By Name

Get a group object by name

### Step 1 - Authenticate to the SOAP API via /auth

- URL: <https://10.0.1.240:443/auth>
- HTTP Method: GET
- Elapsed Time: 0:00:00.008251
- Step 1 Request Body
- Step 1 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip, deflate",
4   "Connection": "keep-alive",
5   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
6   "password": "VGfuaXVtMjAxNSE=",
7   "username": "QWRtaW5pc3RyYXRvcg=="
8 }
```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-length": "135",
4   "content-type": "text/plain; charset=us-ascii"
5 }
```

### Step 2 - Get the server version via /info.json

- URL: <https://10.0.1.240:443/info.json>
- HTTP Method: GET
- Elapsed Time: 0:00:00.013705
- Step 2 Request Body
- Step 2 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip, deflate",
4   "Connection": "keep-alive",
5   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
6   "session": "1-6965-3a521fa0c9310ccd02c32be07145308cf883214615912fe07587f45591f1a674c8e1474f8029d4a"
7 }
```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-length": "87507",
4   "content-type": "application/json"
5 }
```

### Step 3 - Issue a GetObject to find an object

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.002950
- Step 3 Request Body
- Step 3 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "517",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6965-3a521fa0c9310ccd02c32be07145308cf883214615912fe07587f45591f1a674c8e1474f8029d4a"
9 }
```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-length": "941",
4   "content-type": "text/xml; charset=UTF-8"
5 }
```

## Get Sensor By Hash

Get a sensor object by hash

### Step 1 - Authenticate to the SOAP API via /auth

- URL: <https://10.0.1.240:443/auth>
- HTTP Method: GET
- Elapsed Time: 0:00:00.005621
- Step 1 Request Body
- Step 1 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip, deflate",
4   "Connection": "keep-alive",
5   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
6   "password": "VGFuaXVtMjAxNSE=",
7   "username": "QWRtaW5pc3RyYXRvcg=="
8 }

```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-length": "135",
4   "content-type": "text/plain; charset=us-ascii"
5 }

```

## Step 2 - Get the server version via /info.json

- URL: <https://10.0.1.240:443/info.json>
- HTTP Method: GET
- Elapsed Time: 0:00:00.006867
- Step 2 Request Body
- Step 2 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip, deflate",
4   "Connection": "keep-alive",
5   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
6   "session": "1-6966-405a4d6d785287833998f5725a162795ddcb8e8e32b5165dafd33bfbe48c3a908ae37584235c6fa"
7 }

```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-length": "87507",
4   "content-type": "application/json"
5 }

```

## Step 3 - Issue a GetObject to find an object

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.038555
- Step 3 Request Body
- Step 3 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "517",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6966-405a4d6d785287833998f5725a162795ddcb8e8e32b5165dafd33bfbe48c3a908ae37584235c6fa"
9 }
```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }
```

## Get Package By Name

Get a package object by name

### Step 1 - Authenticate to the SOAP API via /auth

- URL: <https://10.0.1.240:443/auth>
- HTTP Method: GET
- Elapsed Time: 0:00:00.005193
- Step 1 Request Body
- Step 1 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip, deflate",
4   "Connection": "keep-alive",
5   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
6   "password": "VGfuaXVtMjAxNSE=",
7   "username": "QWRtaW5pc3RyYXRvcg=="
8 }
```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-length": "135",
4   "content-type": "text/plain; charset=us-ascii"
5 }
```

### Step 2 - Get the server version via /info.json

- URL: <https://10.0.1.240:443/info.json>

- HTTP Method: GET
- Elapsed Time: 0:00:00.012524
- Step 2 Request Body
- Step 2 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip, deflate",
4   "Connection": "keep-alive",
5   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
6   "session": "1-6967-f398b1e06b8bbd61b08d16467a69eca7c2993423ab70b1f8445be973cd243cdc18f5dd4e80adc89"
7 }
```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-length": "87507",
4   "content-type": "application/json"
5 }
```

### Step 3 - Issue a GetObject to find an object

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.002900
- Step 3 Request Body
- Step 3 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "537",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6967-f398b1e06b8bbd61b08d16467a69eca7c2993423ab70b1f8445be973cd243cdc18f5dd4e80adc89"
9 }
```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }
```

## Get Sensor By Names

Get multiple sensor objects by name

### Step 1 - Authenticate to the SOAP API via /auth

- URL: <https://10.0.1.240:443/auth>
- HTTP Method: GET
- Elapsed Time: 0:00:00.006388
- Step 1 Request Body
- Step 1 Response Body
- Request Headers:

```
1 {  
2   "Accept": "*/*",  
3   "Accept-Encoding": "gzip, deflate",  
4   "Connection": "keep-alive",  
5   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",  
6   "password": "VGfuaXVtMjAxNSE=",  
7   "username": "QWRtaW5pc3RyYXRvcg=="  
8 }
```

- Response Headers:

```
1 {  
2   "connection": "keep-alive",  
3   "content-length": "135",  
4   "content-type": "text/plain; charset=us-ascii"  
5 }
```

### Step 2 - Get the server version via /info.json

- URL: <https://10.0.1.240:443/info.json>
- HTTP Method: GET
- Elapsed Time: 0:00:00.006546
- Step 2 Request Body
- Step 2 Response Body
- Request Headers:

```
1 {  
2   "Accept": "*/*",  
3   "Accept-Encoding": "gzip, deflate",  
4   "Connection": "keep-alive",  
5   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",  
6   "session": "1-6968-a86e7e1ee74b43478510f38ffd809826454e5e91dc7afac5b30e9da4e9577a094fa2531983283af"  
7 }
```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-length": "87507",
4   "content-type": "application/json"
5 }

```

### Step 3 - Issue a GetObject to find an object

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.002528
- Step 3 Request Body
- Step 3 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "566",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6968-a86e7e1ee74b43478510f38ffd809826454e5e91dc7afac5b30e9da4e9577a094fa2531983283af"
9 }

```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }

```

## Get Saved Question By Name

Get saved question object by name

### Step 1 - Authenticate to the SOAP API via /auth

- URL: <https://10.0.1.240:443/auth>
- HTTP Method: GET
- Elapsed Time: 0:00:00.006750
- Step 1 Request Body
- Step 1 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip, deflate",
4   "Connection": "keep-alive",
5   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
6   "password": "VGFuaXVtMjAxNSE=",
7   "username": "QWRtaW5pc3RyYXRvcg=="
8 }
```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-length": "135",
4   "content-type": "text/plain; charset=us-ascii"
5 }
```

## Step 2 - Get the server version via /info.json

- URL: <https://10.0.1.240:443/info.json>
- HTTP Method: GET
- Elapsed Time: 0:00:00.012952
- Step 2 Request Body
- Step 2 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip, deflate",
4   "Connection": "keep-alive",
5   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
6   "session": "1-6969-51a3179a5e2ce7650869dbdf99edb480bd4007f47e90f2714da6a1025a5605bf0944747d5021732"
7 }
```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-length": "87506",
4   "content-type": "application/json"
5 }
```

## Step 3 - Issue a GetObject to find an object

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.009170
- Step 3 Request Body
- Step 3 Response Body
- Request Headers:



```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "527",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6969-51a3179a5e2ce7650869dbdf99edb480bd4007f47e90f2714da6a1025a5605bf0944747d5021732"
9 }

```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }

```

## Get User By Id

Get a user object by id

### Step 1 - Authenticate to the SOAP API via /auth

- URL: <https://10.0.1.240:443/auth>
- HTTP Method: GET
- Elapsed Time: 0:00:00.006342
- Step 1 Request Body
- Step 1 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip, deflate",
4   "Connection": "keep-alive",
5   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
6   "password": "VGfuaXVtMjAxNSE=",
7   "username": "QWRtaW5pc3RyYXRvcg=="
8 }

```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-length": "135",
4   "content-type": "text/plain; charset=us-ascii"
5 }

```

### Step 2 - Get the server version via /info.json

- URL: <https://10.0.1.240:443/info.json>

- HTTP Method: GET
- Elapsed Time: 0:00:00.006269
- Step 2 Request Body
- Step 2 Response Body
- Request Headers:

```
1 {  
2   "Accept": "*/*",  
3   "Accept-Encoding": "gzip, deflate",  
4   "Connection": "keep-alive",  
5   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",  
6   "session": "1-6970-aeb0c551b38133b56d9d3cc4f1753d9e6298940c000b442f0d93cdaafc3fda4e92d1317b4e4587c",  
7 }
```

- Response Headers:

```
1 {  
2   "connection": "keep-alive",  
3   "content-length": "87506",  
4   "content-type": "application/json"  
5 }
```

### Step 3 - Issue a GetObject to find an object

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.001448
- Step 3 Request Body
- Step 3 Response Body
- Request Headers:

```
1 {  
2   "Accept": "*/*",  
3   "Accept-Encoding": "gzip",  
4   "Connection": "keep-alive",  
5   "Content-Length": "482",  
6   "Content-Type": "text/xml; charset=utf-8",  
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",  
8   "session": "1-6970-aeb0c551b38133b56d9d3cc4f1753d9e6298940c000b442f0d93cdaafc3fda4e92d1317b4e4587c",  
9 }
```

- Response Headers:

```
1 {  
2   "connection": "keep-alive",  
3   "content-encoding": "gzip",  
4   "content-type": "text/xml; charset=UTF-8",  
5   "transfer-encoding": "chunked"  
6 }
```

## Get Sensor By Name

Get a sensor object by name

### Step 1 - Authenticate to the SOAP API via /auth

- URL: <https://10.0.1.240:443/auth>
- HTTP Method: GET
- Elapsed Time: 0:00:00.005740
- Step 1 Request Body
- Step 1 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip, deflate",
4   "Connection": "keep-alive",
5   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
6   "password": "VGfuaXVtMjAxNSE=",
7   "username": "QWRtaW5pc3RyYXRvcg=="
8 }
```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-length": "135",
4   "content-type": "text/plain; charset=us-ascii"
5 }
```

### Step 2 - Get the server version via /info.json

- URL: <https://10.0.1.240:443/info.json>
- HTTP Method: GET
- Elapsed Time: 0:00:00.012447
- Step 2 Request Body
- Step 2 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip, deflate",
4   "Connection": "keep-alive",
5   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
6   "session": "1-6971-a91cc2515dd0e572df8a068cef2c0e2af12b7bb513b16cf31b587ad91a8b3d49a9322cb50637487"
7 }
```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-length": "87506",
4   "content-type": "application/json"
5 }
```

### Step 3 - Issue a GetObject to find an object

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.001938
- Step 3 Request Body
- Step 3 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "521",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6971-a91cc2515dd0e572df8a068cef2c0e2af12b7bb513b16cf31b587ad91a8b3d49a9322cb506374874"
9 }
```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }
```

## Get Saved Action By Name

Get a saved action object by name

### Step 1 - Authenticate to the SOAP API via /auth

- URL: <https://10.0.1.240:443/auth>
- HTTP Method: GET
- Elapsed Time: 0:00:00.005577
- Step 1 Request Body
- Step 1 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip, deflate",
4   "Connection": "keep-alive",
5   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
6   "password": "VGFuaXVtMjAxNSE=",
7   "username": "QWRtaW5pc3RyYXRvcg=="
8 }

```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-length": "135",
4   "content-type": "text/plain; charset=us-ascii"
5 }

```

## Step 2 - Get the server version via /info.json

- URL: <https://10.0.1.240:443/info.json>
- HTTP Method: GET
- Elapsed Time: 0:00:00.006742
- Step 2 Request Body
- Step 2 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip, deflate",
4   "Connection": "keep-alive",
5   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
6   "session": "1-6972-50f269071b20007ff7331b7e80b2f53e064b3ae3645d9fb08fb07332e445bd1d3ddc5052ce8a620"
7 }

```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-length": "87506",
4   "content-type": "application/json"
5 }

```

## Step 3 - Issue a GetObject to find an object

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.003075
- Step 3 Request Body
- Step 3 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "568",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6972-50f269071b20007ff7331b7e80b2f53e064b3ae3645d9fb08fb07332e445bd1d3ddc5052ce8a6203"
9 }
```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }
```

## Get All Users

Get all user objects

### Step 1 - Authenticate to the SOAP API via /auth

- URL: <https://10.0.1.240:443/auth>
- HTTP Method: GET
- Elapsed Time: 0:00:00.005774
- Step 1 Request Body
- Step 1 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip, deflate",
4   "Connection": "keep-alive",
5   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
6   "password": "VGfuaXVtMjAxNSE=",
7   "username": "QWRtaW5pc3RyYXRvcg=="
8 }
```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-length": "135",
4   "content-type": "text/plain; charset=us-ascii"
5 }
```

### Step 2 - Get the server version via /info.json

- URL: <https://10.0.1.240:443/info.json>

- HTTP Method: GET
- Elapsed Time: 0:00:00.006743
- Step 2 Request Body
- Step 2 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip, deflate",
4   "Connection": "keep-alive",
5   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
6   "session": "1-6973-0cf98fbc7fc7482bd48d0b58a7b70a0ee4dc948d97fa8ba35f4bda38397d1e2cabd2a52abdcce9"
7 }
```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-length": "87506",
4   "content-type": "application/json"
5 }
```

### Step 3 - Issue a GetObject to find an object

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.001660
- Step 3 Request Body
- Step 3 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "468",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6973-0cf98fbc7fc7482bd48d0b58a7b70a0ee4dc948d97fa8ba35f4bda38397d1e2cabd2a52abdcce9"
9 }
```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }
```

## Get All Saved Actions

Get all saved action objects

### Step 1 - Authenticate to the SOAP API via /auth

- URL: <https://10.0.1.240:443/auth>
- HTTP Method: GET
- Elapsed Time: 0:00:00.005721
- Step 1 Request Body
- Step 1 Response Body
- Request Headers:

```
1 {  
2   "Accept": "*/*",  
3   "Accept-Encoding": "gzip, deflate",  
4   "Connection": "keep-alive",  
5   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",  
6   "password": "VGFuaXVtMjAxNSE=",  
7   "username": "QWRtaW5pc3RyYXRvcg=="  
8 }
```

- Response Headers:

```
1 {  
2   "connection": "keep-alive",  
3   "content-length": "135",  
4   "content-type": "text/plain; charset=us-ascii"  
5 }
```

### Step 2 - Get the server version via /info.json

- URL: <https://10.0.1.240:443/info.json>
- HTTP Method: GET
- Elapsed Time: 0:00:00.013732
- Step 2 Request Body
- Step 2 Response Body
- Request Headers:

```
1 {  
2   "Accept": "*/*",  
3   "Accept-Encoding": "gzip, deflate",  
4   "Connection": "keep-alive",  
5   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",  
6   "session": "1-6974-f336dcb44b8422d72afd30cc2b9071d1cb13aa8c5f0b31a7eefc2d698d2f95ed80634b0d4793006"  
7 }
```

- Response Headers:



```

1 {
2   "connection": "keep-alive",
3   "content-length": "87506",
4   "content-type": "application/json"
5 }

```

### Step 3 - Issue a GetObject to find an object

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.015676
- Step 3 Request Body
- Step 3 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "476",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6974-f336dcb44b8422d72afd30cc2b9071d1cb13aa8c5f0b31a7eefc2d698d2f95ed80634b0d4793006"
9 }

```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }

```

## Get All Settings

Get all system setting objects

### Step 1 - Authenticate to the SOAP API via /auth

- URL: <https://10.0.1.240:443/auth>
- HTTP Method: GET
- Elapsed Time: 0:00:00.006038
- Step 1 Request Body
- Step 1 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip, deflate",
4   "Connection": "keep-alive",
5   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
6   "password": "VGfuaXVtMjAxNSE=",
7   "username": "QWRtaW5pc3RyYXRvcg=="
8 }
```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-length": "135",
4   "content-type": "text/plain; charset=us-ascii"
5 }
```

## Step 2 - Get the server version via /info.json

- URL: <https://10.0.1.240:443/info.json>
- HTTP Method: GET
- Elapsed Time: 0:00:00.007142
- Step 2 Request Body
- Step 2 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip, deflate",
4   "Connection": "keep-alive",
5   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
6   "session": "1-6975-a8d699f0849961cb92259f33dc15362b54461d9ac054e2f57035be829e854b29df18a4f5f5a36bf"
7 }
```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-length": "87507",
4   "content-type": "application/json"
5 }
```

## Step 3 - Issue a GetObject to find an object

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.004569
- Step 3 Request Body
- Step 3 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "478",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6975-a8d699f0849961cb92259f33dc15362b54461d9ac054e2f57035be829e854b29df18a4f5f5a36bf"
9 }

```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }

```

## Get All Saved Questions

Get all saved question objects

### Step 1 - Authenticate to the SOAP API via /auth

- URL: <https://10.0.1.240:443/auth>
- HTTP Method: GET
- Elapsed Time: 0:00:00.005370
- Step 1 Request Body
- Step 1 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip, deflate",
4   "Connection": "keep-alive",
5   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
6   "password": "VGfuaXVtMjAxNSE=",
7   "username": "QWRtaW5pc3RyYXRvcg=="
8 }

```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-length": "135",
4   "content-type": "text/plain; charset=us-ascii"
5 }

```

### Step 2 - Get the server version via /info.json

- URL: <https://10.0.1.240:443/info.json>

- HTTP Method: GET
- Elapsed Time: 0:00:00.013595
- Step 2 Request Body
- Step 2 Response Body
- Request Headers:

```
1 {  
2   "Accept": "*/*",  
3   "Accept-Encoding": "gzip, deflate",  
4   "Connection": "keep-alive",  
5   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",  
6   "session": "1-6976-f335d76fd02a7c82ccef7a33f9b3a7632350bc44aa32e237182c35213c777fb28910b86ce0f2cc9a"  
7 }
```

- Response Headers:

```
1 {  
2   "connection": "keep-alive",  
3   "content-length": "87508",  
4   "content-type": "application/json"  
5 }
```

### Step 3 - Issue a GetObject to find an object

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.072889
- Step 3 Request Body
- Step 3 Response Body
- Request Headers:

```
1 {  
2   "Accept": "*/*",  
3   "Accept-Encoding": "gzip",  
4   "Connection": "keep-alive",  
5   "Content-Length": "478",  
6   "Content-Type": "text/xml; charset=utf-8",  
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",  
8   "session": "1-6976-f335d76fd02a7c82ccef7a33f9b3a7632350bc44aa32e237182c35213c777fb28910b86ce0f2cc9a"  
9 }
```

- Response Headers:

```
1 {  
2   "connection": "keep-alive",  
3   "content-encoding": "gzip",  
4   "content-type": "text/xml; charset=UTF-8",  
5   "transfer-encoding": "chunked"  
6 }
```

## Get All Userroless

Get all user role objects

### Step 1 - Authenticate to the SOAP API via /auth

- URL: <https://10.0.1.240:443/auth>
- HTTP Method: GET
- Elapsed Time: 0:00:00.007100
- Step 1 Request Body
- Step 1 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip, deflate",
4   "Connection": "keep-alive",
5   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
6   "password": "VGFuaXVtMjAxNSE=",
7   "username": "QWRtaW5pc3RyYXRvcg=="
8 }
```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-length": "135",
4   "content-type": "text/plain; charset=us-ascii"
5 }
```

### Step 2 - Get the server version via /info.json

- URL: <https://10.0.1.240:443/info.json>
- HTTP Method: GET
- Elapsed Time: 0:00:00.013325
- Step 2 Request Body
- Step 2 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip, deflate",
4   "Connection": "keep-alive",
5   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
6   "session": "1-6977-2b2641c688b30a79bf2deb78457fdae93098553e71f19c87666687f760e13ab5c07e27d1118efd3"
7 }
```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-length": "87508",
4   "content-type": "application/json"
5 }
```

### Step 3 - Issue a GetObject to find an object

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.001701
- Step 3 Request Body
- Step 3 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "468",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6977-2b2641c688b30a79bf2deb78457fdae93098553e71f19c87666687f760e13ab5c07e27d1118efd3b"
9 }
```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }
```

## Get All Questions

Get all question objects

### Step 1 - Authenticate to the SOAP API via /auth

- URL: <https://10.0.1.240:443/auth>
- HTTP Method: GET
- Elapsed Time: 0:00:00.005505
- Step 1 Request Body
- Step 1 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip, deflate",
4   "Connection": "keep-alive",
5   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
6   "password": "VGFuaXVtMjAxNSE=",
7   "username": "QWRtaW5pc3RyYXRvcg=="
8 }

```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-length": "135",
4   "content-type": "text/plain; charset=us-ascii"
5 }

```

## Step 2 - Get the server version via /info.json

- URL: <https://10.0.1.240:443/info.json>
- HTTP Method: GET
- Elapsed Time: 0:00:00.007130
- Step 2 Request Body
- Step 2 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip, deflate",
4   "Connection": "keep-alive",
5   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
6   "session": "1-6978-61e3b03c15a9ca14d1703f5e6f87356c3c51edbf0bfcd4196b8449918de1fcfac4540b6e65518"
7 }

```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-length": "87508",
4   "content-type": "application/json"
5 }

```

## Step 3 - Issue a GetObject to find an object

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.143477
- Step 3 Request Body
- Step 3 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "472",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6978-61e3b03c15a9ca14d1703f5e6f87356c3c51edbf0bfcd4196b8449918de1fcfac4540b6e65518"
9 }
```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }
```

## Get All Groups

Get all group objects

### Step 1 - Authenticate to the SOAP API via /auth

- URL: <https://10.0.1.240:443/auth>
- HTTP Method: GET
- Elapsed Time: 0:00:00.007643
- Step 1 Request Body
- Step 1 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip, deflate",
4   "Connection": "keep-alive",
5   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
6   "password": "VGfuaXVtMjAxNSE=",
7   "username": "QWRtaW5pc3RyYXRvcg=="
8 }
```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-length": "135",
4   "content-type": "text/plain; charset=us-ascii"
5 }
```

### Step 2 - Get the server version via /info.json

- URL: <https://10.0.1.240:443/info.json>



- HTTP Method: GET
- Elapsed Time: 0:00:00.012033
- Step 2 Request Body
- Step 2 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip, deflate",
4   "Connection": "keep-alive",
5   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
6   "session": "1-6979-a466eac2892421e3ba6fc16249ce7d935b80f2443b7cb85369c9dcfe15a72e87da119c65e762eda"
7 }
```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-length": "87508",
4   "content-type": "application/json"
5 }
```

### Step 3 - Issue a GetObject to find an object

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.003675
- Step 3 Request Body
- Step 3 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "469",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6979-a466eac2892421e3ba6fc16249ce7d935b80f2443b7cb85369c9dcfe15a72e87da119c65e762eda"
9 }
```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }
```

## Get All Sensors

Get all sensor objects

### Step 1 - Authenticate to the SOAP API via /auth

- URL: <https://10.0.1.240:443/auth>
- HTTP Method: GET
- Elapsed Time: 0:00:00.005903
- Step 1 Request Body
- Step 1 Response Body
- Request Headers:

```
1 {  
2   "Accept": "*/*",  
3   "Accept-Encoding": "gzip, deflate",  
4   "Connection": "keep-alive",  
5   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",  
6   "password": "VGfuaXVtMjAxNSE=",  
7   "username": "QWRtaW5pc3RyYXRvcg=="  
8 }
```

- Response Headers:

```
1 {  
2   "connection": "keep-alive",  
3   "content-length": "135",  
4   "content-type": "text/plain; charset=us-ascii"  
5 }
```

### Step 2 - Get the server version via /info.json

- URL: <https://10.0.1.240:443/info.json>
- HTTP Method: GET
- Elapsed Time: 0:00:00.011725
- Step 2 Request Body
- Step 2 Response Body
- Request Headers:

```
1 {  
2   "Accept": "*/*",  
3   "Accept-Encoding": "gzip, deflate",  
4   "Connection": "keep-alive",  
5   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",  
6   "session": "1-6980-98f0a0a83553198841a7a094980b0145afa44fe9a9fa7f7628c067edc4df738f6f54a0069d38790"  
7 }
```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-length": "87508",
4   "content-type": "application/json"
5 }

```

### Step 3 - Issue a GetObject to find an object

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.290027
- Step 3 Request Body
- Step 3 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "470",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6980-98f0a0a83553198841a7a094980b0145afa44fe9a9fa7f7628c067edc4df738f6f54a0069d38790"
9 }

```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }

```

## Get All Whitelisted Urls

Get all whitelisted url objects

### Step 1 - Authenticate to the SOAP API via /auth

- URL: <https://10.0.1.240:443/auth>
- HTTP Method: GET
- Elapsed Time: 0:00:00.008187
- Step 1 Request Body
- Step 1 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip, deflate",
4   "Connection": "keep-alive",
5   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
6   "password": "VGFuaXVtMjAxNSE=",
7   "username": "QWRtaW5pc3RyYXRvcg=="
8 }
```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-length": "135",
4   "content-type": "text/plain; charset=us-ascii"
5 }
```

## Step 2 - Get the server version via /info.json

- URL: <https://10.0.1.240:443/info.json>
- HTTP Method: GET
- Elapsed Time: 0:00:00.013009
- Step 2 Request Body
- Step 2 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip, deflate",
4   "Connection": "keep-alive",
5   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
6   "session": "1-6981-c60f9ad57629fa6b39fa005d3d0dd5c2de7ee10f0a088f9290cfdeb67e9bdb682c80c5d7a89172b"
7 }
```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-length": "87510",
4   "content-type": "application/json"
5 }
```

## Step 3 - Issue a GetObject to find an object

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.268061
- Step 3 Request Body
- Step 3 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "480",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6981-c60f9ad57629fa6b39fa005d3d0dd5c2de7ee10f0a088f9290cfdeb67e9bdb682c80c5d7a89172b1"
9 }

```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }

```

## Get All Clients

Get all client objects

### Step 1 - Authenticate to the SOAP API via /auth

- URL: <https://10.0.1.240:443/auth>
- HTTP Method: GET
- Elapsed Time: 0:00:00.008631
- Step 1 Request Body
- Step 1 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip, deflate",
4   "Connection": "keep-alive",
5   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
6   "password": "VGfuaXVtMjAxNSE=",
7   "username": "QWRtaW5pc3RyYXRvcg=="
8 }

```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-length": "135",
4   "content-type": "text/plain; charset=us-ascii"
5 }

```

### Step 2 - Get the server version via /info.json

- URL: <https://10.0.1.240:443/info.json>

- HTTP Method: GET
- Elapsed Time: 0:00:00.014325
- Step 2 Request Body
- Step 2 Response Body
- Request Headers:

```
1 {  
2   "Accept": "*/*",  
3   "Accept-Encoding": "gzip, deflate",  
4   "Connection": "keep-alive",  
5   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",  
6   "session": "1-6982-93453cd0ca79099442b16bfccd073eb355105c088557acca3dd8e3b9708e812692b8a8272047fa3",  
7 }
```

- Response Headers:

```
1 {  
2   "connection": "keep-alive",  
3   "content-length": "87510",  
4   "content-type": "application/json"  
5 }
```

### Step 3 - Issue a GetObject to find an object

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.001785
- Step 3 Request Body
- Step 3 Response Body
- Request Headers:

```
1 {  
2   "Accept": "*/*",  
3   "Accept-Encoding": "gzip",  
4   "Connection": "keep-alive",  
5   "Content-Length": "476",  
6   "Content-Type": "text/xml; charset=utf-8",  
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",  
8   "session": "1-6982-93453cd0ca79099442b16bfccd073eb355105c088557acca3dd8e3b9708e812692b8a8272047fa3",  
9 }
```

- Response Headers:

```
1 {  
2   "connection": "keep-alive",  
3   "content-encoding": "gzip",  
4   "content-type": "text/xml; charset=UTF-8",  
5   "transfer-encoding": "chunked"  
6 }
```

## Get All Packages

Get all package objects

### Step 1 - Authenticate to the SOAP API via /auth

- URL: <https://10.0.1.240:443/auth>
- HTTP Method: GET
- Elapsed Time: 0:00:00.006304
- Step 1 Request Body
- Step 1 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip, deflate",
4   "Connection": "keep-alive",
5   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
6   "password": "VGfuaXVtMjAxNSE=",
7   "username": "QWRtaW5pc3RyYXRvcg=="
8 }
```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-length": "135",
4   "content-type": "text/plain; charset=us-ascii"
5 }
```

### Step 2 - Get the server version via /info.json

- URL: <https://10.0.1.240:443/info.json>
- HTTP Method: GET
- Elapsed Time: 0:00:00.007158
- Step 2 Request Body
- Step 2 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip, deflate",
4   "Connection": "keep-alive",
5   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
6   "session": "1-6983-3cc1ff3c8446559d98a5700a15ce3106b11b317becff4716b35e037470c711ca51079fd83f54c80"
7 }
```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-length": "87510",
4   "content-type": "application/json"
5 }
```

### Step 3 - Issue a GetObject to find an object

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.007834
- Step 3 Request Body
- Step 3 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "475",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6983-3cc1ff3c8446559d98a5700a15ce3106b11b317becff4716b35e037470c711ca51079fd83f54c80"
9 }
```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }
```

## Get All Actions

Get all action objects

### Step 1 - Authenticate to the SOAP API via /auth

- URL: <https://10.0.1.240:443/auth>
- HTTP Method: GET
- Elapsed Time: 0:00:00.007050
- Step 1 Request Body
- Step 1 Response Body
- Request Headers:



```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip, deflate",
4   "Connection": "keep-alive",
5   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
6   "password": "VGFuaXVtMjAxNSE=",
7   "username": "QWRtaW5pc3RyYXRvcg=="
8 }

```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-length": "135",
4   "content-type": "text/plain; charset=us-ascii"
5 }

```

## Step 2 - Get the server version via /info.json

- URL: <https://10.0.1.240:443/info.json>
- HTTP Method: GET
- Elapsed Time: 0:00:00.014342
- Step 2 Request Body
- Step 2 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip, deflate",
4   "Connection": "keep-alive",
5   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
6   "session": "1-6984-bb4f26ed649c869eb5b67f115eb1497f835c4631cbe1dda8cfa17b0d9167d3116690b9500840506"
7 }

```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-length": "87511",
4   "content-type": "application/json"
5 }

```

## Step 3 - Issue a GetObject to find an object

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.015045
- Step 3 Request Body
- Step 3 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "470",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6984-bb4f26ed649c869eb5b67f115eb1497f835c4631cbe1dda8cfa17b0d9167d3116690b9500840506"
9 }
```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }
```

## Ask Manual Question Simple Multiple Sensors

Ask the question ‘Get Computer Name and Installed Applications from all machines’, wait for result data to be complete, and get result data

### Step 1 - Authenticate to the SOAP API via /auth

- URL: <https://10.0.1.240:443/auth>
- HTTP Method: GET
- Elapsed Time: 0:00:00.008149
- Step 1 Request Body
- Step 1 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip, deflate",
4   "Connection": "keep-alive",
5   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
6   "password": "VGfuaXVtMjAxNSE=",
7   "username": "QWRtaW5pc3RyYXRvcg=="
8 }
```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-length": "135",
4   "content-type": "text/plain; charset=us-ascii"
5 }
```

**Step 2 - Get the server version via /info.json**

- URL: <https://10.0.1.240:443/info.json>
- HTTP Method: GET
- Elapsed Time: 0:00:00.014113
- Step 2 Request Body
- Step 2 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip, deflate",
4   "Connection": "keep-alive",
5   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
6   "session": "1-6985-df19baee504e368b3605cb68783a8097e76e834044ae23e5ae3bb5cffa9e098b0e7784fe873a3a1a"
7 }

```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-length": "87512",
4   "content-type": "application/json"
5 }

```

**Step 3 - Issue a GetObject to get the full object of a sensor for inclusion in a Select for a Question**

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.002321
- Step 3 Request Body
- Step 3 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "565",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6985-df19baee504e368b3605cb68783a8097e76e834044ae23e5ae3bb5cffa9e098b0e7784fe873a3a1a"
9 }

```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",

```

```
5  "transfer-encoding": "chunked"
6  }
```

#### Step 4 - Issue a GetObject to get the full object of a sensor for inclusion in a Select for a Question

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.001886
- Step 4 Request Body
- Step 4 Response Body
- Request Headers:

```
1  {
2  "Accept": "*/*",
3  "Accept-Encoding": "gzip",
4  "Connection": "keep-alive",
5  "Content-Length": "574",
6  "Content-Type": "text/xml; charset=utf-8",
7  "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8  "session": "1-6985-df19baee504e368b3605cb68783a8097e76e834044ae23e5ae3bb5cffa9e098b0e7784fe873a3a1a",
9  }
```

- Response Headers:

```
1  {
2  "connection": "keep-alive",
3  "content-encoding": "gzip",
4  "content-type": "text/xml; charset=UTF-8",
5  "transfer-encoding": "chunked"
6  }
```

#### Step 5 - Issue an AddObject to add a Question object

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.007882
- Step 5 Request Body
- Step 5 Response Body
- Request Headers:

```
1  {
2  "Accept": "*/*",
3  "Accept-Encoding": "gzip",
4  "Connection": "keep-alive",
5  "Content-Length": "753",
6  "Content-Type": "text/xml; charset=utf-8",
7  "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8  "session": "1-6985-df19baee504e368b3605cb68783a8097e76e834044ae23e5ae3bb5cffa9e098b0e7784fe873a3a1a",
9  }
```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-length": "769",
4   "content-type": "text/xml; charset=UTF-8"
5 }
```

#### Step 6 - Issue a GetObject on the recently added object in order to get the full object

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.013309
- Step 6 Request Body
- Step 6 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "494",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6985-df19baee504e368b3605cb68783a8097e76e834044ae23e5ae3bb5cffa9e098b0e7784fe873a3a1a"
9 }
```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }
```

#### Step 7 - Issue a GetResultInfo for a Question to check the current progress of answers

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.001560
- Step 7 Request Body
- Step 7 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "498",
```

```
6  "Content-Type": "text/xml; charset=utf-8",
7  "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8  "session": "1-6985-df19baee504e368b3605cb68783a8097e76e834044ae23e5ae3bb5cffa9e098b0e7784fe873a3a1"
9  }
```

- Response Headers:

```
1  {
2  "connection": "keep-alive",
3  "content-encoding": "gzip",
4  "content-type": "text/xml; charset=UTF-8",
5  "transfer-encoding": "chunked"
6  }
```

#### Step 8 - Issue a GetResultInfo for a Question to check the current progress of answers

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.001449
- Step 8 Request Body
- Step 8 Response Body
- Request Headers:

```
1  {
2  "Accept": "*/*",
3  "Accept-Encoding": "gzip",
4  "Connection": "keep-alive",
5  "Content-Length": "498",
6  "Content-Type": "text/xml; charset=utf-8",
7  "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8  "session": "1-6985-df19baee504e368b3605cb68783a8097e76e834044ae23e5ae3bb5cffa9e098b0e7784fe873a3a1"
9  }
```

- Response Headers:

```
1  {
2  "connection": "keep-alive",
3  "content-encoding": "gzip",
4  "content-type": "text/xml; charset=UTF-8",
5  "transfer-encoding": "chunked"
6  }
```

#### Step 9 - Issue a GetResultInfo for a Question to check the current progress of answers

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.001554
- Step 9 Request Body
- Step 9 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "498",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6985-df19baee504e368b3605cb68783a8097e76e834044ae23e5ae3bb5cffa9e098b0e7784fe873a3a1"
9 }

```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }

```

#### Step 10 - Issue a GetResultInfo for a Question to check the current progress of answers

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.002098
- Step 10 Request Body
- Step 10 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "498",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6985-df19baee504e368b3605cb68783a8097e76e834044ae23e5ae3bb5cffa9e098b0e7784fe873a3a1"
9 }

```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }

```

#### Step 11 - Issue a GetResultInfo for a Question to check the current progress of answers

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.002153

- Step 11 Request Body
- Step 11 Response Body
- Request Headers:

```
1 {  
2  "Accept": "*/*",  
3  "Accept-Encoding": "gzip",  
4  "Connection": "keep-alive",  
5  "Content-Length": "498",  
6  "Content-Type": "text/xml; charset=utf-8",  
7  "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",  
8  "session": "1-6985-df19baee504e368b3605cb68783a8097e76e834044ae23e5ae3bb5cffa9e098b0e7784fe873a3a1a",  
9 }
```

- Response Headers:

```
1 {  
2  "connection": "keep-alive",  
3  "content-encoding": "gzip",  
4  "content-type": "text/xml; charset=UTF-8",  
5  "transfer-encoding": "chunked"  
6 }
```

## Step 12 - Issue a GetResultInfo for a Question to check the current progress of answers

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.002380
- Step 12 Request Body
- Step 12 Response Body
- Request Headers:

```
1 {  
2  "Accept": "*/*",  
3  "Accept-Encoding": "gzip",  
4  "Connection": "keep-alive",  
5  "Content-Length": "498",  
6  "Content-Type": "text/xml; charset=utf-8",  
7  "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",  
8  "session": "1-6985-df19baee504e368b3605cb68783a8097e76e834044ae23e5ae3bb5cffa9e098b0e7784fe873a3a1a",  
9 }
```

- Response Headers:

```
1 {  
2  "connection": "keep-alive",  
3  "content-encoding": "gzip",  
4  "content-type": "text/xml; charset=UTF-8",  
5  "transfer-encoding": "chunked"  
6 }
```



**Step 13 - Issue a GetResultData to get answers for a question**

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.003278
- Step 13 Request Body
- Step 13 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "526",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6985-df19baee504e368b3605cb68783a8097e76e834044ae23e5ae3bb5cffa9e098b0e7784fe873a3a1"
9 }

```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }

```

**Ask Manual Question Simple Single Sensor**

Ask the question ‘Get Computer Name from all machines’, wait for result data to be complete, and get result data

**Step 1 - Authenticate to the SOAP API via /auth**

- URL: <https://10.0.1.240:443/auth>
- HTTP Method: GET
- Elapsed Time: 0:00:00.007940
- Step 1 Request Body
- Step 1 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip, deflate",
4   "Connection": "keep-alive",
5   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
6   "password": "VGfuaXVtMjAxNSE=",
7   "username": "QWRtaW5pc3RyYXRvcg=="
8 }

```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-length": "135",
4   "content-type": "text/plain; charset=us-ascii"
5 }
```

## Step 2 - Get the server version via /info.json

- URL: <https://10.0.1.240:443/info.json>
- HTTP Method: GET
- Elapsed Time: 0:00:00.014139
- Step 2 Request Body
- Step 2 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip, deflate",
4   "Connection": "keep-alive",
5   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
6   "session": "1-6986-5a7cd189759baf6790709943f871eef02e8ac71d33313c8ffbd9c6e55aecc814c38a584d9741721"
7 }
```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-length": "87511",
4   "content-type": "application/json"
5 }
```

## Step 3 - Issue a GetObject to get the full object of a sensor for inclusion in a Select for a Question

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.001766
- Step 3 Request Body
- Step 3 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "565",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
```

```

8  "session": "1-6986-5a7cd189759baf6790709943f871eef02e8ac71d33313c8ffbd9c6e55aecc814c38a584d97417211"
9  }

```

- Response Headers:

```

1  {
2  "connection": "keep-alive",
3  "content-encoding": "gzip",
4  "content-type": "text/xml; charset=UTF-8",
5  "transfer-encoding": "chunked"
6  }

```

#### Step 4 - Issue an AddObject to add a Question object

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.007892
- Step 4 Request Body
- Step 4 Response Body
- Request Headers:

```

1  {
2  "Accept": "*/*",
3  "Accept-Encoding": "gzip",
4  "Connection": "keep-alive",
5  "Content-Length": "639",
6  "Content-Type": "text/xml; charset=utf-8",
7  "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8  "session": "1-6986-5a7cd189759baf6790709943f871eef02e8ac71d33313c8ffbd9c6e55aecc814c38a584d97417211"
9  }

```

- Response Headers:

```

1  {
2  "connection": "keep-alive",
3  "content-length": "769",
4  "content-type": "text/xml; charset=UTF-8"
5  }

```

#### Step 5 - Issue a GetObject on the recently added object in order to get the full object

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.013276
- Step 5 Request Body
- Step 5 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "494",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6986-5a7cd189759baf6790709943f871eef02e8ac71d33313c8ffbd9c6e55aecc814c38a584d97417211"
9 }
```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }
```

#### Step 6 - Issue a GetResultInfo for a Question to check the current progress of answers

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.001613
- Step 6 Request Body
- Step 6 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "498",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6986-5a7cd189759baf6790709943f871eef02e8ac71d33313c8ffbd9c6e55aecc814c38a584d97417211"
9 }
```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }
```

#### Step 7 - Issue a GetResultInfo for a Question to check the current progress of answers

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.002290

- Step 7 Request Body
- Step 7 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "498",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6986-5a7cd189759baf6790709943f871eef02e8ac71d33313c8ffbd9c6e55aecc814c38a584d9741721"
9 }

```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }

```

#### Step 8 - Issue a GetResultData to get answers for a question

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.001622
- Step 8 Request Body
- Step 8 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "526",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6986-5a7cd189759baf6790709943f871eef02e8ac71d33313c8ffbd9c6e55aecc814c38a584d9741721"
9 }

```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }

```

## Ask Manual Question Sensor With Parameters And Some Supplied Parameters

Ask the question 'Get Folder Name Search with RegEx Match[Program Files,Microsoft.\*] from all machines', wait for result data to be complete, and get result data

### Step 1 - Authenticate to the SOAP API via /auth

- URL: <https://10.0.1.240:443/auth>
- HTTP Method: GET
- Elapsed Time: 0:00:00.008065
- Step 1 Request Body
- Step 1 Response Body
- Request Headers:

```
1 {  
2   "Accept": "*/*",  
3   "Accept-Encoding": "gzip, deflate",  
4   "Connection": "keep-alive",  
5   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",  
6   "password": "VGFuaXVtMjAxNSE=",  
7   "username": "QWRtaW5pc3RyYXRvcg=="  
8 }
```

- Response Headers:

```
1 {  
2   "connection": "keep-alive",  
3   "content-length": "135",  
4   "content-type": "text/plain; charset=us-ascii"  
5 }
```

### Step 2 - Get the server version via /info.json

- URL: <https://10.0.1.240:443/info.json>
- HTTP Method: GET
- Elapsed Time: 0:00:00.013871
- Step 2 Request Body
- Step 2 Response Body
- Request Headers:

```
1 {  
2   "Accept": "*/*",  
3   "Accept-Encoding": "gzip, deflate",  
4   "Connection": "keep-alive",  
5   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",  
6   "session": "1-6988-05836d41c7be5addca66f76227a78b0d47fead12ac9dd8b6b2af69ce4387d294ba47baf393a4fd0"  
7 }
```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-length": "87511",
4   "content-type": "application/json"
5 }

```

### Step 3 - Issue a GetObject to get the full object of a sensor for inclusion in a Select for a Question

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.002496
- Step 3 Request Body
- Step 3 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "587",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6988-05836d41c7be5addca66f76227a78b0d47fead12ac9dd8b6b2af69ce4387d294ba47baf393a4fd0"
9 }

```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }

```

### Step 4 - Issue an AddObject to add a Question object

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.021312
- Step 4 Request Body
- Step 4 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "1003",
6   "Content-Type": "text/xml; charset=utf-8",

```

```
7 "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8 "session": "1-6988-05836d41c7be5addca66f76227a78b0d47fead12ac9dd8b6b2af69ce4387d294ba47baf393a4fd0e",
9 }
```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-length": "769",
4   "content-type": "text/xml; charset=UTF-8"
5 }
```

#### Step 5 - Issue a GetObject on the recently added object in order to get the full object

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.040338
- Step 5 Request Body
- Step 5 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "494",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6988-05836d41c7be5addca66f76227a78b0d47fead12ac9dd8b6b2af69ce4387d294ba47baf393a4fd0e",
9 }
```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }
```

#### Step 6 - Issue a GetResultInfo for a Question to check the current progress of answers

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.001756
- Step 6 Request Body
- Step 6 Response Body
- Request Headers:



```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "498",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6988-05836d41c7be5addca66f76227a78b0d47fead12ac9dd8b6b2af69ce4387d294ba47baf393a4fd0"
9 }

```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }

```

#### Step 7 - Issue a GetResultInfo for a Question to check the current progress of answers

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.002215
- Step 7 Request Body
- Step 7 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "498",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6988-05836d41c7be5addca66f76227a78b0d47fead12ac9dd8b6b2af69ce4387d294ba47baf393a4fd0"
9 }

```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }

```

#### Step 8 - Issue a GetResultInfo for a Question to check the current progress of answers

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.002240

- Step 8 Request Body
- Step 8 Response Body
- Request Headers:

```
1 {  
2   "Accept": "*/*",  
3   "Accept-Encoding": "gzip",  
4   "Connection": "keep-alive",  
5   "Content-Length": "498",  
6   "Content-Type": "text/xml; charset=utf-8",  
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",  
8   "session": "1-6988-05836d41c7be5addca66f76227a78b0d47fead12ac9dd8b6b2af69ce4387d294ba47baf393a4fd0e",  
9 }
```

- Response Headers:

```
1 {  
2   "connection": "keep-alive",  
3   "content-encoding": "gzip",  
4   "content-type": "text/xml; charset=UTF-8",  
5   "transfer-encoding": "chunked"  
6 }
```

#### Step 9 - Issue a GetResultInfo for a Question to check the current progress of answers

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.002076
- Step 9 Request Body
- Step 9 Response Body
- Request Headers:

```
1 {  
2   "Accept": "*/*",  
3   "Accept-Encoding": "gzip",  
4   "Connection": "keep-alive",  
5   "Content-Length": "498",  
6   "Content-Type": "text/xml; charset=utf-8",  
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",  
8   "session": "1-6988-05836d41c7be5addca66f76227a78b0d47fead12ac9dd8b6b2af69ce4387d294ba47baf393a4fd0e",  
9 }
```

- Response Headers:

```
1 {  
2   "connection": "keep-alive",  
3   "content-encoding": "gzip",  
4   "content-type": "text/xml; charset=UTF-8",  
5   "transfer-encoding": "chunked"  
6 }
```

**Step 10 - Issue a GetResultInfo for a Question to check the current progress of answers**

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.002232
- Step 10 Request Body
- Step 10 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "498",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6988-05836d41c7be5addca66f76227a78b0d47fead12ac9dd8b6b2af69ce4387d294ba47baf393a4fd0e"
9 }

```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }

```

**Step 11 - Issue a GetResultInfo for a Question to check the current progress of answers**

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.002211
- Step 11 Request Body
- Step 11 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "498",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6988-05836d41c7be5addca66f76227a78b0d47fead12ac9dd8b6b2af69ce4387d294ba47baf393a4fd0e"
9 }

```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }
```

#### Step 12 - Issue a GetResultInfo for a Question to check the current progress of answers

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.001850
- Step 12 Request Body
- Step 12 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "498",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6988-05836d41c7be5addca66f76227a78b0d47fead12ac9dd8b6b2af69ce4387d294ba47baf393a4fd0e"
9 }
```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }
```

#### Step 13 - Issue a GetResultInfo for a Question to check the current progress of answers

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.002142
- Step 13 Request Body
- Step 13 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "498",
```

```

6  "Content-Type": "text/xml; charset=utf-8",
7  "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8  "session": "1-6988-05836d41c7be5addca66f76227a78b0d47fead12ac9dd8b6b2af69ce4387d294ba47baf393a4fd0e"
9  }

```

- Response Headers:

```

1  {
2  "connection": "keep-alive",
3  "content-encoding": "gzip",
4  "content-type": "text/xml; charset=UTF-8",
5  "transfer-encoding": "chunked"
6  }

```

#### Step 14 - Issue a GetResultInfo for a Question to check the current progress of answers

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.002194
- Step 14 Request Body
- Step 14 Response Body
- Request Headers:

```

1  {
2  "Accept": "*/*",
3  "Accept-Encoding": "gzip",
4  "Connection": "keep-alive",
5  "Content-Length": "498",
6  "Content-Type": "text/xml; charset=utf-8",
7  "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8  "session": "1-6988-05836d41c7be5addca66f76227a78b0d47fead12ac9dd8b6b2af69ce4387d294ba47baf393a4fd0e"
9  }

```

- Response Headers:

```

1  {
2  "connection": "keep-alive",
3  "content-encoding": "gzip",
4  "content-type": "text/xml; charset=UTF-8",
5  "transfer-encoding": "chunked"
6  }

```

#### Step 15 - Issue a GetResultInfo for a Question to check the current progress of answers

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.002162
- Step 15 Request Body
- Step 15 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "498",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6988-05836d41c7be5addca66f76227a78b0d47fead12ac9dd8b6b2af69ce4387d294ba47baf393a4fd0"
9 }
```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }
```

#### Step 16 - Issue a `GetResultInfo` for a Question to check the current progress of answers

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.002877
- Step 16 Request Body
- Step 16 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "498",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6988-05836d41c7be5addca66f76227a78b0d47fead12ac9dd8b6b2af69ce4387d294ba47baf393a4fd0"
9 }
```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }
```

#### Step 17 - Issue a `GetResultData` to get answers for a question

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.005290

- Step 17 Request Body
- Step 17 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "526",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6988-05836d41c7be5addca66f76227a78b0d47fead12ac9dd8b6b2af69ce4387d294ba47baf393a4fd06"
9 }

```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }

```

## Ask Manual Question Multiple Sensors With Parameters And Some Supplied Parameters

Ask the question ‘Get Folder Name Search with RegEx Match[Program Files, , No, No, Microsoft.\*] and Computer Name from all machines’, wait for result data to be complete, and get result data

### Step 1 - Authenticate to the SOAP API via /auth

- URL: <https://10.0.1.240:443/auth>
- HTTP Method: GET
- Elapsed Time: 0:00:00.016327
- Step 1 Request Body
- Step 1 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip, deflate",
4   "Connection": "keep-alive",
5   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
6   "password": "VGFuaXVtMjAxNSE=",
7   "username": "QWRtaW5pc3RyYXRvcg=="
8 }

```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-length": "135",
4   "content-type": "text/plain; charset=us-ascii"
5 }

```

**Step 2 - Get the server version via /info.json**

- URL: <https://10.0.1.240:443/info.json>
- HTTP Method: GET
- Elapsed Time: 0:00:00.014295
- Step 2 Request Body
- Step 2 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip, deflate",
4   "Connection": "keep-alive",
5   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
6   "session": "1-6989-0a6f5bd674c41d8e3da5b1a0fb91cefaad9e2d713b26ceb4911bd3fc82fce71420e9b4013e42581"
7 }
```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-length": "87614",
4   "content-type": "application/json"
5 }
```

**Step 3 - Issue a GetObject to get the full object of a sensor for inclusion in a Select for a Question**

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.002678
- Step 3 Request Body
- Step 3 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "587",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6989-0a6f5bd674c41d8e3da5b1a0fb91cefaad9e2d713b26ceb4911bd3fc82fce71420e9b4013e42581"
9 }
```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
```



```

5  "transfer-encoding": "chunked"
6  }

```

#### Step 4 - Issue a GetObject to get the full object of a sensor for inclusion in a Select for a Question

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.002104
- Step 4 Request Body
- Step 4 Response Body
- Request Headers:

```

1  {
2  "Accept": "*/*",
3  "Accept-Encoding": "gzip",
4  "Connection": "keep-alive",
5  "Content-Length": "565",
6  "Content-Type": "text/xml; charset=utf-8",
7  "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8  "session": "1-6989-0a6f5bd674c41d8e3da5b1a0fb91cefaad9e2d713b26ceb4911bd3fc82fce71420e9b4013e425810"
9  }

```

- Response Headers:

```

1  {
2  "connection": "keep-alive",
3  "content-encoding": "gzip",
4  "content-type": "text/xml; charset=UTF-8",
5  "transfer-encoding": "chunked"
6  }

```

#### Step 5 - Issue an AddObject to add a Question object

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.019997
- Step 5 Request Body
- Step 5 Response Body
- Request Headers:

```

1  {
2  "Accept": "*/*",
3  "Accept-Encoding": "gzip",
4  "Connection": "keep-alive",
5  "Content-Length": "1117",
6  "Content-Type": "text/xml; charset=utf-8",
7  "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8  "session": "1-6989-0a6f5bd674c41d8e3da5b1a0fb91cefaad9e2d713b26ceb4911bd3fc82fce71420e9b4013e425810"
9  }

```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-length": "769",
4   "content-type": "text/xml; charset=UTF-8"
5 }
```

#### Step 6 - Issue a GetObject on the recently added object in order to get the full object

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.038632
- Step 6 Request Body
- Step 6 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "494",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6989-0a6f5bd674c41d8e3da5b1a0fb91cefaad9e2d713b26ceb4911bd3fc82fce71420e9b4013e425810"
9 }
```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }
```

#### Step 7 - Issue a GetResultInfo for a Question to check the current progress of answers

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.001769
- Step 7 Request Body
- Step 7 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "498",
```

```

6  "Content-Type": "text/xml; charset=utf-8",
7  "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8  "session": "1-6989-0a6f5bd674c41d8e3da5b1a0fb91cefaad9e2d713b26ceb4911bd3fc82fce71420e9b4013e42581"
9  }

```

- Response Headers:

```

1  {
2  "connection": "keep-alive",
3  "content-encoding": "gzip",
4  "content-type": "text/xml; charset=UTF-8",
5  "transfer-encoding": "chunked"
6  }

```

#### Step 8 - Issue a GetResultInfo for a Question to check the current progress of answers

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.002101
- Step 8 Request Body
- Step 8 Response Body
- Request Headers:

```

1  {
2  "Accept": "*/*",
3  "Accept-Encoding": "gzip",
4  "Connection": "keep-alive",
5  "Content-Length": "498",
6  "Content-Type": "text/xml; charset=utf-8",
7  "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8  "session": "1-6989-0a6f5bd674c41d8e3da5b1a0fb91cefaad9e2d713b26ceb4911bd3fc82fce71420e9b4013e42581"
9  }

```

- Response Headers:

```

1  {
2  "connection": "keep-alive",
3  "content-encoding": "gzip",
4  "content-type": "text/xml; charset=UTF-8",
5  "transfer-encoding": "chunked"
6  }

```

#### Step 9 - Issue a GetResultInfo for a Question to check the current progress of answers

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.002241
- Step 9 Request Body
- Step 9 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "498",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6989-0a6f5bd674c41d8e3da5b1a0fb91cefaad9e2d713b26ceb4911bd3fc82fce71420e9b4013e42581"
9 }
```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }
```

#### Step 10 - Issue a `GetResultInfo` for a Question to check the current progress of answers

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.001812
- Step 10 Request Body
- Step 10 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "498",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6989-0a6f5bd674c41d8e3da5b1a0fb91cefaad9e2d713b26ceb4911bd3fc82fce71420e9b4013e42581"
9 }
```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }
```

#### Step 11 - Issue a `GetResultInfo` for a Question to check the current progress of answers

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.001735

- Step 11 Request Body
- Step 11 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "498",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6989-0a6f5bd674c41d8e3da5b1a0fb91cefaad9e2d713b26ceb4911bd3fc82fce71420e9b4013e42581"
9 }

```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }

```

## Step 12 - Issue a GetResultInfo for a Question to check the current progress of answers

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.002245
- Step 12 Request Body
- Step 12 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "498",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6989-0a6f5bd674c41d8e3da5b1a0fb91cefaad9e2d713b26ceb4911bd3fc82fce71420e9b4013e42581"
9 }

```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }

```

### Step 13 - Issue a GetResultInfo for a Question to check the current progress of answers

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.002252
- Step 13 Request Body
- Step 13 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "498",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6989-0a6f5bd674c41d8e3da5b1a0fb91cefaad9e2d713b26ceb4911bd3fc82fce71420e9b4013e42581"
9 }
```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }
```

### Step 14 - Issue a GetResultInfo for a Question to check the current progress of answers

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.002166
- Step 14 Request Body
- Step 14 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "498",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6989-0a6f5bd674c41d8e3da5b1a0fb91cefaad9e2d713b26ceb4911bd3fc82fce71420e9b4013e42581"
9 }
```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }

```

#### Step 15 - Issue a GetResultInfo for a Question to check the current progress of answers

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.001882
- Step 15 Request Body
- Step 15 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "498",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6989-0a6f5bd674c41d8e3da5b1a0fb91cefaad9e2d713b26ceb4911bd3fc82fce71420e9b4013e42581"
9 }

```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }

```

#### Step 16 - Issue a GetResultInfo for a Question to check the current progress of answers

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.002162
- Step 16 Request Body
- Step 16 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "498",

```

```
6  "Content-Type": "text/xml; charset=utf-8",
7  "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8  "session": "1-6989-0a6f5bd674c41d8e3da5b1a0fb91cefaad9e2d713b26ceb4911bd3fc82fce71420e9b4013e42581"
9  }
```

- Response Headers:

```
1  {
2  "connection": "keep-alive",
3  "content-encoding": "gzip",
4  "content-type": "text/xml; charset=UTF-8",
5  "transfer-encoding": "chunked"
6  }
```

#### Step 17 - Issue a GetResultInfo for a Question to check the current progress of answers

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.002169
- Step 17 Request Body
- Step 17 Response Body
- Request Headers:

```
1  {
2  "Accept": "*/*",
3  "Accept-Encoding": "gzip",
4  "Connection": "keep-alive",
5  "Content-Length": "498",
6  "Content-Type": "text/xml; charset=utf-8",
7  "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8  "session": "1-6989-0a6f5bd674c41d8e3da5b1a0fb91cefaad9e2d713b26ceb4911bd3fc82fce71420e9b4013e42581"
9  }
```

- Response Headers:

```
1  {
2  "connection": "keep-alive",
3  "content-encoding": "gzip",
4  "content-type": "text/xml; charset=UTF-8",
5  "transfer-encoding": "chunked"
6  }
```

#### Step 18 - Issue a GetResultInfo for a Question to check the current progress of answers

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.001497
- Step 18 Request Body
- Step 18 Response Body
- Request Headers:



```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "498",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6989-0a6f5bd674c41d8e3da5b1a0fb91cefaad9e2d713b26ceb4911bd3fc82fce71420e9b4013e42581"
9 }

```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }

```

### Step 19 - Issue a GetResultInfo for a Question to check the current progress of answers

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.002106
- Step 19 Request Body
- Step 19 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "498",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6989-0a6f5bd674c41d8e3da5b1a0fb91cefaad9e2d713b26ceb4911bd3fc82fce71420e9b4013e42581"
9 }

```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }

```

### Step 20 - Issue a GetResultInfo for a Question to check the current progress of answers

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.002004

- Step 20 Request Body
- Step 20 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "498",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6989-0a6f5bd674c41d8e3da5b1a0fb91cefaad9e2d713b26ceb4911bd3fc82fce71420e9b4013e425810",
9 }
```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }
```

#### Step 21 - Issue a GetResultInfo for a Question to check the current progress of answers

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.002156
- Step 21 Request Body
- Step 21 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "498",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6989-0a6f5bd674c41d8e3da5b1a0fb91cefaad9e2d713b26ceb4911bd3fc82fce71420e9b4013e425810",
9 }
```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }
```

**Step 22 - Issue a GetResultInfo for a Question to check the current progress of answers**

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.002059
- Step 22 Request Body
- Step 22 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "498",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6989-0a6f5bd674c41d8e3da5b1a0fb91cefaad9e2d713b26ceb4911bd3fc82fce71420e9b4013e425810"
9 }

```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }

```

**Step 23 - Issue a GetResultInfo for a Question to check the current progress of answers**

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.002309
- Step 23 Request Body
- Step 23 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "498",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6989-0a6f5bd674c41d8e3da5b1a0fb91cefaad9e2d713b26ceb4911bd3fc82fce71420e9b4013e425810"
9 }

```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }
```

#### Step 24 - Issue a GetResultInfo for a Question to check the current progress of answers

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.002313
- Step 24 Request Body
- Step 24 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "498",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6989-0a6f5bd674c41d8e3da5b1a0fb91cefaad9e2d713b26ceb4911bd3fc82fce71420e9b4013e42581"
9 }
```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }
```

#### Step 25 - Issue a GetResultInfo for a Question to check the current progress of answers

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.002819
- Step 25 Request Body
- Step 25 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "498",
```

```

6  "Content-Type": "text/xml; charset=utf-8",
7  "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8  "session": "1-6989-0a6f5bd674c41d8e3da5b1a0fb91cefaad9e2d713b26ceb4911bd3fc82fce71420e9b4013e42581"
9  }

```

- Response Headers:

```

1  {
2  "connection": "keep-alive",
3  "content-encoding": "gzip",
4  "content-type": "text/xml; charset=UTF-8",
5  "transfer-encoding": "chunked"
6  }

```

#### Step 26 - Issue a GetResultInfo for a Question to check the current progress of answers

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.002749
- Step 26 Request Body
- Step 26 Response Body
- Request Headers:

```

1  {
2  "Accept": "*/*",
3  "Accept-Encoding": "gzip",
4  "Connection": "keep-alive",
5  "Content-Length": "498",
6  "Content-Type": "text/xml; charset=utf-8",
7  "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8  "session": "1-6989-0a6f5bd674c41d8e3da5b1a0fb91cefaad9e2d713b26ceb4911bd3fc82fce71420e9b4013e42581"
9  }

```

- Response Headers:

```

1  {
2  "connection": "keep-alive",
3  "content-encoding": "gzip",
4  "content-type": "text/xml; charset=UTF-8",
5  "transfer-encoding": "chunked"
6  }

```

#### Step 27 - Issue a GetResultInfo for a Question to check the current progress of answers

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.002601
- Step 27 Request Body
- Step 27 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "498",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6989-0a6f5bd674c41d8e3da5b1a0fb91cefaad9e2d713b26ceb4911bd3fc82fce71420e9b4013e42581"
9 }
```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }
```

#### Step 28 - Issue a GetResultInfo for a Question to check the current progress of answers

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.002821
- Step 28 Request Body
- Step 28 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "498",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6989-0a6f5bd674c41d8e3da5b1a0fb91cefaad9e2d713b26ceb4911bd3fc82fce71420e9b4013e42581"
9 }
```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }
```

#### Step 29 - Issue a GetResultInfo for a Question to check the current progress of answers

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.002798

- Step 29 Request Body
- Step 29 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "498",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6989-0a6f5bd674c41d8e3da5b1a0fb91cefaad9e2d713b26ceb4911bd3fc82fce71420e9b4013e42581"
9 }

```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }

```

### Step 30 - Issue a GetResultInfo for a Question to check the current progress of answers

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.002945
- Step 30 Request Body
- Step 30 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "498",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6989-0a6f5bd674c41d8e3da5b1a0fb91cefaad9e2d713b26ceb4911bd3fc82fce71420e9b4013e42581"
9 }

```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }

```

### Step 31 - Issue a `GetResultInfo` for a Question to check the current progress of answers

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.002904
- Step 31 Request Body
- Step 31 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "498",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6989-0a6f5bd674c41d8e3da5b1a0fb91cefaad9e2d713b26ceb4911bd3fc82fce71420e9b4013e425810"
9 }
```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }
```

### Step 32 - Issue a `GetResultData` to get answers for a question

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.002898
- Step 32 Request Body
- Step 32 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "526",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6989-0a6f5bd674c41d8e3da5b1a0fb91cefaad9e2d713b26ceb4911bd3fc82fce71420e9b4013e425810"
9 }
```

- Response Headers:



```

1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }

```

### Ask Manual Question Sensor With Parameters And No Supplied Parameters

Ask the question ‘Get Folder Name Search with RegEx Match from all machines’ using sane defaults for parameters, wait for result data to be complete, and get result data

#### Step 1 - Authenticate to the SOAP API via /auth

- URL: <https://10.0.1.240:443/auth>
- HTTP Method: GET
- Elapsed Time: 0:00:00.006460
- Step 1 Request Body
- Step 1 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip, deflate",
4   "Connection": "keep-alive",
5   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
6   "password": "VGFuaXVtMjAxNSE=",
7   "username": "QWRtaW5pc3RyYXRvcg=="
8 }

```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-length": "135",
4   "content-type": "text/plain; charset=us-ascii"
5 }

```

#### Step 2 - Get the server version via /info.json

- URL: <https://10.0.1.240:443/info.json>
- HTTP Method: GET
- Elapsed Time: 0:00:00.012795
- Step 2 Request Body
- Step 2 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip, deflate",
4   "Connection": "keep-alive",
5   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
6   "session": "1-6991-050b6d41b647d68310f9deb32de806e13b27ac42be5cecefe7b740685b54e740d4af331c939f94b1"
7 }
```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-length": "87813",
4   "content-type": "application/json"
5 }
```

### Step 3 - Issue a GetObject to get the full object of a sensor for inclusion in a Select for a Question

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.002139
- Step 3 Request Body
- Step 3 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "587",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6991-050b6d41b647d68310f9deb32de806e13b27ac42be5cecefe7b740685b54e740d4af331c939f94b1"
9 }
```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }
```

### Step 4 - Issue an AddObject to add a Question object

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.013984
- Step 4 Request Body

- Step 4 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "915",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6991-050b6d41b647d68310f9deb32de806e13b27ac42be5cecefe7b740685b54e740d4af331c939f94b1"
9 }

```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-length": "769",
4   "content-type": "text/xml; charset=UTF-8"
5 }

```

#### Step 5 - Issue a GetObject on the recently added object in order to get the full object

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.034328
- Step 5 Request Body
- Step 5 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "494",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6991-050b6d41b647d68310f9deb32de806e13b27ac42be5cecefe7b740685b54e740d4af331c939f94b1"
9 }

```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }

```

#### Step 6 - Issue a GetResultInfo for a Question to check the current progress of answers

- URL: <https://10.0.1.240:443/soap>

- HTTP Method: POST
- Elapsed Time: 0:00:00.001614
- Step 6 Request Body
- Step 6 Response Body
- Request Headers:

```
1 {  
2   "Accept": "*/*",  
3   "Accept-Encoding": "gzip",  
4   "Connection": "keep-alive",  
5   "Content-Length": "498",  
6   "Content-Type": "text/xml; charset=utf-8",  
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",  
8   "session": "1-6991-050b6d41b647d68310f9deb32de806e13b27ac42be5cecefe7b740685b54e740d4af331c939f94b1",  
9 }
```

- Response Headers:

```
1 {  
2   "connection": "keep-alive",  
3   "content-encoding": "gzip",  
4   "content-type": "text/xml; charset=UTF-8",  
5   "transfer-encoding": "chunked"  
6 }
```

#### Step 7 - Issue a GetResultInfo for a Question to check the current progress of answers

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.002247
- Step 7 Request Body
- Step 7 Response Body
- Request Headers:

```
1 {  
2   "Accept": "*/*",  
3   "Accept-Encoding": "gzip",  
4   "Connection": "keep-alive",  
5   "Content-Length": "498",  
6   "Content-Type": "text/xml; charset=utf-8",  
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",  
8   "session": "1-6991-050b6d41b647d68310f9deb32de806e13b27ac42be5cecefe7b740685b54e740d4af331c939f94b1",  
9 }
```

- Response Headers:

```
1 {  
2   "connection": "keep-alive",  
3   "content-encoding": "gzip",  
4   "content-type": "text/xml; charset=UTF-8",  
5   "transfer-encoding": "chunked"  
6 }
```

**Step 8 - Issue a GetResultInfo for a Question to check the current progress of answers**

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.002092
- Step 8 Request Body
- Step 8 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "498",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6991-050b6d41b647d68310f9deb32de806e13b27ac42be5cecefe7b740685b54e740d4af331c939f94b7"
9 }

```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }

```

**Step 9 - Issue a GetResultInfo for a Question to check the current progress of answers**

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.002088
- Step 9 Request Body
- Step 9 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "498",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6991-050b6d41b647d68310f9deb32de806e13b27ac42be5cecefe7b740685b54e740d4af331c939f94b7"
9 }

```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }
```

#### Step 10 - Issue a GetResultInfo for a Question to check the current progress of answers

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.002095
- Step 10 Request Body
- Step 10 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "498",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6991-050b6d41b647d68310f9deb32de806e13b27ac42be5cecefe7b740685b54e740d4af331c939f94b"
9 }
```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }
```

#### Step 11 - Issue a GetResultInfo for a Question to check the current progress of answers

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.002077
- Step 11 Request Body
- Step 11 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "498",
```

```

6  "Content-Type": "text/xml; charset=utf-8",
7  "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8  "session": "1-6991-050b6d41b647d68310f9deb32de806e13b27ac42be5cecefe7b740685b54e740d4af331c939f94b"
9  }

```

- Response Headers:

```

1  {
2  "connection": "keep-alive",
3  "content-encoding": "gzip",
4  "content-type": "text/xml; charset=UTF-8",
5  "transfer-encoding": "chunked"
6  }

```

### Step 12 - Issue a GetResultInfo for a Question to check the current progress of answers

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.001843
- Step 12 Request Body
- Step 12 Response Body
- Request Headers:

```

1  {
2  "Accept": "*/*",
3  "Accept-Encoding": "gzip",
4  "Connection": "keep-alive",
5  "Content-Length": "498",
6  "Content-Type": "text/xml; charset=utf-8",
7  "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8  "session": "1-6991-050b6d41b647d68310f9deb32de806e13b27ac42be5cecefe7b740685b54e740d4af331c939f94b"
9  }

```

- Response Headers:

```

1  {
2  "connection": "keep-alive",
3  "content-encoding": "gzip",
4  "content-type": "text/xml; charset=UTF-8",
5  "transfer-encoding": "chunked"
6  }

```

### Step 13 - Issue a GetResultInfo for a Question to check the current progress of answers

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.001758
- Step 13 Request Body
- Step 13 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "498",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6991-050b6d41b647d68310f9deb32de806e13b27ac42be5cecefe7b740685b54e740d4af331c939f94b1"
9 }
```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }
```

#### Step 14 - Issue a GetResultInfo for a Question to check the current progress of answers

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.002647
- Step 14 Request Body
- Step 14 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "498",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6991-050b6d41b647d68310f9deb32de806e13b27ac42be5cecefe7b740685b54e740d4af331c939f94b1"
9 }
```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }
```

#### Step 15 - Issue a GetResultInfo for a Question to check the current progress of answers

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.001766



- Step 15 Request Body
- Step 15 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "498",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6991-050b6d41b647d68310f9deb32de806e13b27ac42be5cecefe7b740685b54e740d4af331c939f94b1"
9 }

```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }

```

#### Step 16 - Issue a GetResultInfo for a Question to check the current progress of answers

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.001431
- Step 16 Request Body
- Step 16 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "498",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6991-050b6d41b647d68310f9deb32de806e13b27ac42be5cecefe7b740685b54e740d4af331c939f94b1"
9 }

```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }

```

**Step 17 - Issue a GetResultInfo for a Question to check the current progress of answers**

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.001358
- Step 17 Request Body
- Step 17 Response Body
- Request Headers:

```
1 {  
2   "Accept": "*/*",  
3   "Accept-Encoding": "gzip",  
4   "Connection": "keep-alive",  
5   "Content-Length": "498",  
6   "Content-Type": "text/xml; charset=utf-8",  
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",  
8   "session": "1-6991-050b6d41b647d68310f9deb32de806e13b27ac42be5cecefe7b740685b54e740d4af331c939f94b1",  
9 }
```

- Response Headers:

```
1 {  
2   "connection": "keep-alive",  
3   "content-encoding": "gzip",  
4   "content-type": "text/xml; charset=UTF-8",  
5   "transfer-encoding": "chunked"  
6 }
```

**Step 18 - Issue a GetResultInfo for a Question to check the current progress of answers**

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.001872
- Step 18 Request Body
- Step 18 Response Body
- Request Headers:

```
1 {  
2   "Accept": "*/*",  
3   "Accept-Encoding": "gzip",  
4   "Connection": "keep-alive",  
5   "Content-Length": "498",  
6   "Content-Type": "text/xml; charset=utf-8",  
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",  
8   "session": "1-6991-050b6d41b647d68310f9deb32de806e13b27ac42be5cecefe7b740685b54e740d4af331c939f94b1",  
9 }
```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }

```

#### Step 19 - Issue a GetResultInfo for a Question to check the current progress of answers

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.002165
- Step 19 Request Body
- Step 19 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "498",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6991-050b6d41b647d68310f9deb32de806e13b27ac42be5cecefe7b740685b54e740d4af331c939f94b"
9 }

```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }

```

#### Step 20 - Issue a GetResultInfo for a Question to check the current progress of answers

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.002035
- Step 20 Request Body
- Step 20 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "498",

```

```
6  "Content-Type": "text/xml; charset=utf-8",
7  "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8  "session": "1-6991-050b6d41b647d68310f9deb32de806e13b27ac42be5cecefe7b740685b54e740d4af331c939f94b"
9  }
```

- Response Headers:

```
1  {
2  "connection": "keep-alive",
3  "content-encoding": "gzip",
4  "content-type": "text/xml; charset=UTF-8",
5  "transfer-encoding": "chunked"
6  }
```

### Step 21 - Issue a GetResultInfo for a Question to check the current progress of answers

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.002250
- Step 21 Request Body
- Step 21 Response Body
- Request Headers:

```
1  {
2  "Accept": "*/*",
3  "Accept-Encoding": "gzip",
4  "Connection": "keep-alive",
5  "Content-Length": "498",
6  "Content-Type": "text/xml; charset=utf-8",
7  "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8  "session": "1-6991-050b6d41b647d68310f9deb32de806e13b27ac42be5cecefe7b740685b54e740d4af331c939f94b"
9  }
```

- Response Headers:

```
1  {
2  "connection": "keep-alive",
3  "content-encoding": "gzip",
4  "content-type": "text/xml; charset=UTF-8",
5  "transfer-encoding": "chunked"
6  }
```

### Step 22 - Issue a GetResultInfo for a Question to check the current progress of answers

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.002073
- Step 22 Request Body
- Step 22 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "498",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6991-050b6d41b647d68310f9deb32de806e13b27ac42be5cecefe7b740685b54e740d4af331c939f94b1"
9 }

```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }

```

### Step 23 - Issue a GetResultInfo for a Question to check the current progress of answers

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.001999
- Step 23 Request Body
- Step 23 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "498",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6991-050b6d41b647d68310f9deb32de806e13b27ac42be5cecefe7b740685b54e740d4af331c939f94b1"
9 }

```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }

```

### Step 24 - Issue a GetResultInfo for a Question to check the current progress of answers

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.002042

- Step 24 Request Body
- Step 24 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "498",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6991-050b6d41b647d68310f9deb32de806e13b27ac42be5cecefe7b740685b54e740d4af331c939f94b1",
9 }
```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }
```

#### Step 25 - Issue a GetResultInfo for a Question to check the current progress of answers

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.002114
- Step 25 Request Body
- Step 25 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "498",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6991-050b6d41b647d68310f9deb32de806e13b27ac42be5cecefe7b740685b54e740d4af331c939f94b1",
9 }
```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }
```

**Step 26 - Issue a GetResultInfo for a Question to check the current progress of answers**

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.002965
- Step 26 Request Body
- Step 26 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "498",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6991-050b6d41b647d68310f9deb32de806e13b27ac42be5cecefe7b740685b54e740d4af331c939f94b7"
9 }

```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }

```

**Step 27 - Issue a GetResultInfo for a Question to check the current progress of answers**

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.003084
- Step 27 Request Body
- Step 27 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "498",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6991-050b6d41b647d68310f9deb32de806e13b27ac42be5cecefe7b740685b54e740d4af331c939f94b7"
9 }

```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }
```

#### Step 28 - Issue a GetResultData to get answers for a question

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.005488
- Step 28 Request Body
- Step 28 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "526",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6991-050b6d41b647d68310f9deb32de806e13b27ac42be5cecefe7b740685b54e740d4af331c939f94b"
9 }
```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }
```

#### Ask Manual Question Sensor With Parameters And Filter

Ask the question ‘Get Folder Name Search with RegEx Match[Program Files, , No, No, Microsoft.\*] containing “Shared” from all machines’, wait for result data to be complete, and get result data

#### Step 1 - Authenticate to the SOAP API via /auth

- URL: <https://10.0.1.240:443/auth>
- HTTP Method: GET
- Elapsed Time: 0:00:00.007992
- Step 1 Request Body
- Step 1 Response Body
- Request Headers:



```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip, deflate",
4   "Connection": "keep-alive",
5   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
6   "password": "VGFuaXVtMjAxNSE=",
7   "username": "QWRtaW5pc3RyYXRvcg=="
8 }

```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-length": "135",
4   "content-type": "text/plain; charset=us-ascii"
5 }

```

## Step 2 - Get the server version via /info.json

- URL: <https://10.0.1.240:443/info.json>
- HTTP Method: GET
- Elapsed Time: 0:00:00.014992
- Step 2 Request Body
- Step 2 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip, deflate",
4   "Connection": "keep-alive",
5   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
6   "session": "1-6992-c40d5e60dacfd24c809d9fa0385d0c0798b093201fc30229a8eed981c76f6dd42e081763b835f2d"
7 }

```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-length": "87914",
4   "content-type": "application/json"
5 }

```

## Step 3 - Issue a GetObject to get the full object of a sensor for inclusion in a Select for a Question

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.002108
- Step 3 Request Body
- Step 3 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "587",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6992-c40d5e60dacfd24c809d9fa0385d0c0798b093201fc30229a8eed981c76f6dd42e081763b835f2d"
9 }
```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }
```

#### Step 4 - Issue an AddObject to add a Question object

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.019908
- Step 4 Request Body
- Step 4 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "1081",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6992-c40d5e60dacfd24c809d9fa0385d0c0798b093201fc30229a8eed981c76f6dd42e081763b835f2d"
9 }
```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-length": "769",
4   "content-type": "text/xml; charset=UTF-8"
5 }
```

#### Step 5 - Issue a GetObject on the recently added object in order to get the full object

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.045453

- Step 5 Request Body
- Step 5 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "494",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6992-c40d5e60dacfd24c809d9fa0385d0c0798b093201fc30229a8eed981c76f6dd42e081763b835f2d0"
9 }

```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }

```

#### Step 6 - Issue a GetResultInfo for a Question to check the current progress of answers

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.001612
- Step 6 Request Body
- Step 6 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "498",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6992-c40d5e60dacfd24c809d9fa0385d0c0798b093201fc30229a8eed981c76f6dd42e081763b835f2d0"
9 }

```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }

```

**Step 7 - Issue a GetResultInfo for a Question to check the current progress of answers**

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.002106
- Step 7 Request Body
- Step 7 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "498",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6992-c40d5e60dacfd24c809d9fa0385d0c0798b093201fc30229a8eed981c76f6dd42e081763b835f2d"
9 }
```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }
```

**Step 8 - Issue a GetResultInfo for a Question to check the current progress of answers**

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.002144
- Step 8 Request Body
- Step 8 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "498",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6992-c40d5e60dacfd24c809d9fa0385d0c0798b093201fc30229a8eed981c76f6dd42e081763b835f2d"
9 }
```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }

```

#### Step 9 - Issue a GetResultInfo for a Question to check the current progress of answers

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.002064
- Step 9 Request Body
- Step 9 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "498",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6992-c40d5e60dacfd24c809d9fa0385d0c0798b093201fc30229a8eed981c76f6dd42e081763b835f2d"
9 }

```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }

```

#### Step 10 - Issue a GetResultInfo for a Question to check the current progress of answers

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.002143
- Step 10 Request Body
- Step 10 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "498",

```

```

"Content-Type": "text/xml; charset=utf-8",
"User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
"session": "1-6992-c40d5e60dacfd24c809d9fa0385d0c0798b093201fc30229a8eed981c76f6dd42e"
}

```

- Response Headers:

```
{
  "connection": "keep-alive",
  "content-encoding": "gzip",
  "content-type": "text/xml; charset=UTF-8",
  "transfer-encoding": "chunked"
}
```

### Step 11 - Issue a GetResultInfo for a Question to check the current progress of answers

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.002220
- Step 11 Request Body
- Step 11 Response Body
- Request Headers:

```
{
  "Accept": "*/*",
  "Accept-Encoding": "gzip",
  "Connection": "keep-alive",
  "Content-Length": "498",
  "Content-Type": "text/xml; charset=utf-8",
  "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
  "session": "1-6992-c40d5e60dacfd24c809d9fa0385d0c0798b093201fc30229a8eed981c76f6dd42e"
}
```

- Response Headers:

```
{
  "connection": "keep-alive",
  "content-encoding": "gzip",
  "content-type": "text/xml; charset=UTF-8",
  "transfer-encoding": "chunked"
}
```

### Step 12 - Issue a GetResultInfo for a Question to check the current progress of answers

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.002236
- Step 12 Request Body
- Step 12 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "498",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6992-c40d5e60dacfd24c809d9fa0385d0c0798b093201fc30229a8eed981c76f6dd42e081763b835f2d"
9 }

```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }

```

### Step 13 - Issue a GetResultInfo for a Question to check the current progress of answers

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.002221
- Step 13 Request Body
- Step 13 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "498",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6992-c40d5e60dacfd24c809d9fa0385d0c0798b093201fc30229a8eed981c76f6dd42e081763b835f2d"
9 }

```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }

```

### Step 14 - Issue a GetResultInfo for a Question to check the current progress of answers

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.002199

- Step 14 Request Body
- Step 14 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "498",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6992-c40d5e60dacfd24c809d9fa0385d0c0798b093201fc30229a8eed981c76f6dd42e081763b835f2d0"
9 }
```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }
```

#### Step 15 - Issue a GetResultInfo for a Question to check the current progress of answers

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.002099
- Step 15 Request Body
- Step 15 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "498",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6992-c40d5e60dacfd24c809d9fa0385d0c0798b093201fc30229a8eed981c76f6dd42e081763b835f2d0"
9 }
```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }
```



**Step 16 - Issue a GetResultInfo for a Question to check the current progress of answers**

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.001708
- Step 16 Request Body
- Step 16 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "498",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6992-c40d5e60dacfd24c809d9fa0385d0c0798b093201fc30229a8eed981c76f6dd42e081763b835f2d
9 }
```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }
```

**Step 17 - Issue a GetResultInfo for a Question to check the current progress of answers**

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.002067
- Step 17 Request Body
- Step 17 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "498",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6992-c40d5e60dacfd24c809d9fa0385d0c0798b093201fc30229a8eed981c76f6dd42e081763b835f2d
9 }
```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }
```

#### Step 18 - Issue a GetResultInfo for a Question to check the current progress of answers

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.002111
- Step 18 Request Body
- Step 18 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "498",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6992-c40d5e60dacfd24c809d9fa0385d0c0798b093201fc30229a8eed981c76f6dd42e081763b835f2d"
9 }
```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }
```

#### Step 19 - Issue a GetResultInfo for a Question to check the current progress of answers

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.002176
- Step 19 Request Body
- Step 19 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "498",
```

```

6  "Content-Type": "text/xml; charset=utf-8",
7  "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8  "session": "1-6992-c40d5e60dacfd24c809d9fa0385d0c0798b093201fc30229a8eed981c76f6dd42e081763b835f2d
9  }

```

- Response Headers:

```

1  {
2  "connection": "keep-alive",
3  "content-encoding": "gzip",
4  "content-type": "text/xml; charset=UTF-8",
5  "transfer-encoding": "chunked"
6  }

```

#### Step 20 - Issue a GetResultInfo for a Question to check the current progress of answers

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.002106
- Step 20 Request Body
- Step 20 Response Body
- Request Headers:

```

1  {
2  "Accept": "*/*",
3  "Accept-Encoding": "gzip",
4  "Connection": "keep-alive",
5  "Content-Length": "498",
6  "Content-Type": "text/xml; charset=utf-8",
7  "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8  "session": "1-6992-c40d5e60dacfd24c809d9fa0385d0c0798b093201fc30229a8eed981c76f6dd42e081763b835f2d
9  }

```

- Response Headers:

```

1  {
2  "connection": "keep-alive",
3  "content-encoding": "gzip",
4  "content-type": "text/xml; charset=UTF-8",
5  "transfer-encoding": "chunked"
6  }

```

#### Step 21 - Issue a GetResultInfo for a Question to check the current progress of answers

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.002305
- Step 21 Request Body
- Step 21 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "498",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6992-c40d5e60dacfd24c809d9fa0385d0c0798b093201fc30229a8eed981c76f6dd42e081763b835f2d"
9 }
```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }
```

#### Step 22 - Issue a GetResultInfo for a Question to check the current progress of answers

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.002164
- Step 22 Request Body
- Step 22 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "498",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6992-c40d5e60dacfd24c809d9fa0385d0c0798b093201fc30229a8eed981c76f6dd42e081763b835f2d"
9 }
```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }
```

#### Step 23 - Issue a GetResultInfo for a Question to check the current progress of answers

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.002386

- Step 23 Request Body
- Step 23 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "498",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6992-c40d5e60dacfd24c809d9fa0385d0c0798b093201fc30229a8eed981c76f6dd42e081763b835f2d0"
9 }

```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }

```

#### Step 24 - Issue a GetResultInfo for a Question to check the current progress of answers

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.002185
- Step 24 Request Body
- Step 24 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "498",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6992-c40d5e60dacfd24c809d9fa0385d0c0798b093201fc30229a8eed981c76f6dd42e081763b835f2d0"
9 }

```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }

```

**Step 25 - Issue a GetResultInfo for a Question to check the current progress of answers**

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.002139
- Step 25 Request Body
- Step 25 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "498",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6992-c40d5e60dacfd24c809d9fa0385d0c0798b093201fc30229a8eed981c76f6dd42e081763b835f2d"
9 }
```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }
```

**Step 26 - Issue a GetResultInfo for a Question to check the current progress of answers**

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.001818
- Step 26 Request Body
- Step 26 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "498",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6992-c40d5e60dacfd24c809d9fa0385d0c0798b093201fc30229a8eed981c76f6dd42e081763b835f2d"
9 }
```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }

```

#### Step 27 - Issue a GetResultInfo for a Question to check the current progress of answers

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.002167
- Step 27 Request Body
- Step 27 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "498",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6992-c40d5e60dacfd24c809d9fa0385d0c0798b093201fc30229a8eed981c76f6dd42e081763b835f2d"
9 }

```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }

```

#### Step 28 - Issue a GetResultInfo for a Question to check the current progress of answers

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.002295
- Step 28 Request Body
- Step 28 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "498",

```

```
6  "Content-Type": "text/xml; charset=utf-8",
7  "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8  "session": "1-6992-c40d5e60dacfd24c809d9fa0385d0c0798b093201fc30229a8eed981c76f6dd42e081763b835f2d
9  }
```

- Response Headers:

```
1  {
2  "connection": "keep-alive",
3  "content-encoding": "gzip",
4  "content-type": "text/xml; charset=UTF-8",
5  "transfer-encoding": "chunked"
6  }
```

#### Step 29 - Issue a GetResultInfo for a Question to check the current progress of answers

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.001825
- Step 29 Request Body
- Step 29 Response Body
- Request Headers:

```
1  {
2  "Accept": "*/*",
3  "Accept-Encoding": "gzip",
4  "Connection": "keep-alive",
5  "Content-Length": "498",
6  "Content-Type": "text/xml; charset=utf-8",
7  "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8  "session": "1-6992-c40d5e60dacfd24c809d9fa0385d0c0798b093201fc30229a8eed981c76f6dd42e081763b835f2d
9  }
```

- Response Headers:

```
1  {
2  "connection": "keep-alive",
3  "content-encoding": "gzip",
4  "content-type": "text/xml; charset=UTF-8",
5  "transfer-encoding": "chunked"
6  }
```

#### Step 30 - Issue a GetResultInfo for a Question to check the current progress of answers

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.002186
- Step 30 Request Body
- Step 30 Response Body
- Request Headers:



```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "498",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6992-c40d5e60dacfd24c809d9fa0385d0c0798b093201fc30229a8eed981c76f6dd42e081763b835f2d"
9 }

```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }

```

### Step 31 - Issue a GetResultData to get answers for a question

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.002053
- Step 31 Request Body
- Step 31 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "526",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6992-c40d5e60dacfd24c809d9fa0385d0c0798b093201fc30229a8eed981c76f6dd42e081763b835f2d"
9 }

```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }

```

### Ask Manual Question Sensor With Filter And 2 Options

Ask the question ‘Get Operating System containing “Windows” from all machines’ and set max\_age\_seconds to 3600 and value\_type to 1 on the Operating System sensor, then wait for result data to be complete, and get result data

### Step 1 - Authenticate to the SOAP API via /auth

- URL: <https://10.0.1.240:443/auth>
- HTTP Method: GET
- Elapsed Time: 0:00:00.008899
- Step 1 Request Body
- Step 1 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip, deflate",
4   "Connection": "keep-alive",
5   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
6   "password": "VGfuaXVtMjAxNSE=",
7   "username": "QWRtaW5pc3RyYXRvcg=="
8 }
```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-length": "135",
4   "content-type": "text/plain; charset=us-ascii"
5 }
```

### Step 2 - Get the server version via /info.json

- URL: <https://10.0.1.240:443/info.json>
- HTTP Method: GET
- Elapsed Time: 0:00:00.014913
- Step 2 Request Body
- Step 2 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip, deflate",
4   "Connection": "keep-alive",
5   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
6   "session": "1-6993-17fa9da998b1727bf636c12c4388c86cbd275fef1bcd8dd676fb15b62a9a60e410dca9d5d481c6f3"
7 }
```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-length": "88017",
4   "content-type": "application/json"
5 }
```

**Step 3 - Issue a GetObject to get the full object of a sensor for inclusion in a Select for a Question**

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.002268
- Step 3 Request Body
- Step 3 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "568",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6993-17fa9da998b1727bf636c12c4388c86cbd275fef1bcd8dd676fb15b62a9a60e410dca9d5d481c6f3"
9 }

```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }

```

**Step 4 - Issue an AddObject to add a Question object**

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.009065
- Step 4 Request Body
- Step 4 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "784",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6993-17fa9da998b1727bf636c12c4388c86cbd275fef1bcd8dd676fb15b62a9a60e410dca9d5d481c6f3"
9 }

```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-length": "769",
4   "content-type": "text/xml; charset=UTF-8"
5 }
```

#### Step 5 - Issue a `GetObject` on the recently added object in order to get the full object

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.013911
- Step 5 Request Body
- Step 5 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "494",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6993-17fa9da998b1727bf636c12c4388c86cbd275fef1bcd8dd676fb15b62a9a60e410dca9d5d481c6f"
9 }
```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }
```

#### Step 6 - Issue a `GetResultInfo` for a Question to check the current progress of answers

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.001928
- Step 6 Request Body
- Step 6 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "498",
6   "Content-Type": "text/xml; charset=utf-8",
```

```

7  "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8  "session": "1-6993-17fa9da998b1727bf636c12c4388c86cbd275fef1bcd8dd676fb15b62a9a60e410dca9d5d481c6f3"
9  }

```

- Response Headers:

```

1  {
2  "connection": "keep-alive",
3  "content-encoding": "gzip",
4  "content-type": "text/xml; charset=UTF-8",
5  "transfer-encoding": "chunked"
6  }

```

#### Step 7 - Issue a GetResultInfo for a Question to check the current progress of answers

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.002026
- Step 7 Request Body
- Step 7 Response Body
- Request Headers:

```

1  {
2  "Accept": "*/*",
3  "Accept-Encoding": "gzip",
4  "Connection": "keep-alive",
5  "Content-Length": "498",
6  "Content-Type": "text/xml; charset=utf-8",
7  "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8  "session": "1-6993-17fa9da998b1727bf636c12c4388c86cbd275fef1bcd8dd676fb15b62a9a60e410dca9d5d481c6f3"
9  }

```

- Response Headers:

```

1  {
2  "connection": "keep-alive",
3  "content-encoding": "gzip",
4  "content-type": "text/xml; charset=UTF-8",
5  "transfer-encoding": "chunked"
6  }

```

#### Step 8 - Issue a GetResultInfo for a Question to check the current progress of answers

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.002111
- Step 8 Request Body
- Step 8 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "498",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6993-17fa9da998b1727bf636c12c4388c86cbd275fef1bcd8dd676fb15b62a9a60e410dca9d5d481c6f3"
9 }
```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }
```

### Step 9 - Issue a GetResultData to get answers for a question

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.001795
- Step 9 Request Body
- Step 9 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "526",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6993-17fa9da998b1727bf636c12c4388c86cbd275fef1bcd8dd676fb15b62a9a60e410dca9d5d481c6f3"
9 }
```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }
```

### Ask Manual Question Sensor With Filter

Ask the question ‘Get Operating System containing “Windows” from all machines’, then wait for result data to be complete, and get result data

**Step 1 - Authenticate to the SOAP API via /auth**

- URL: <https://10.0.1.240:443/auth>
- HTTP Method: GET
- Elapsed Time: 0:00:00.006415
- Step 1 Request Body
- Step 1 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip, deflate",
4   "Connection": "keep-alive",
5   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
6   "password": "VGfuaXVtMjAxNSE=",
7   "username": "QWRtaW5pc3RyYXRvcg=="
8 }

```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-length": "135",
4   "content-type": "text/plain; charset=us-ascii"
5 }

```

**Step 2 - Get the server version via /info.json**

- URL: <https://10.0.1.240:443/info.json>
- HTTP Method: GET
- Elapsed Time: 0:00:00.017046
- Step 2 Request Body
- Step 2 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip, deflate",
4   "Connection": "keep-alive",
5   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
6   "session": "1-6994-82a216fcbf87d26d95d8d0dd477147367bab210c9d2aef59f5d8a31401c429700341c38ab52e8c5"
7 }

```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-length": "88017",
4   "content-type": "application/json"
5 }

```

### Step 3 - Issue a GetObject to get the full object of a sensor for inclusion in a Select for a Question

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.002471
- Step 3 Request Body
- Step 3 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "568",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6994-82a216fc87d26d95d8d0dd477147367bab210c9d2aef59f5d8a31401c429700341c38ab52e8c5"
9 }
```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }
```

### Step 4 - Issue an AddObject to add a Question object

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.008750
- Step 4 Request Body
- Step 4 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "714",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6994-82a216fc87d26d95d8d0dd477147367bab210c9d2aef59f5d8a31401c429700341c38ab52e8c5"
9 }
```

- Response Headers:



```

1 {
2   "connection": "keep-alive",
3   "content-length": "769",
4   "content-type": "text/xml; charset=UTF-8"
5 }

```

#### Step 5 - Issue a GetObject on the recently added object in order to get the full object

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.013267
- Step 5 Request Body
- Step 5 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "494",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6994-82a216fcbf87d26d95d8d0dd477147367bab210c9d2aef59f5d8a31401c429700341c38ab52e8c5"
9 }

```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }

```

#### Step 6 - Issue a GetResultInfo for a Question to check the current progress of answers

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.001604
- Step 6 Request Body
- Step 6 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "498",
6   "Content-Type": "text/xml; charset=utf-8",

```

```
7 "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8 "session": "1-6994-82a216fcfb87d26d95d8d0dd477147367bab210c9d2aef59f5d8a31401c429700341c38ab52e8c5"
9 }
```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }
```

### Step 7 - Issue a `GetResultInfo` for a Question to check the current progress of answers

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.002184
- Step 7 Request Body
- Step 7 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "498",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6994-82a216fcfb87d26d95d8d0dd477147367bab210c9d2aef59f5d8a31401c429700341c38ab52e8c5"
9 }
```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }
```

### Step 8 - Issue a `GetResultData` to get answers for a question

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.001704
- Step 8 Request Body
- Step 8 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "526",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6994-82a216fc87d26d95d8d0dd477147367bab210c9d2aef59f5d8a31401c429700341c38ab52e8c5"
9 }

```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }

```

## Ask Manual Question Sensor With Parameters And Filter And Options

Ask the question ‘Get Folder Name Search with RegEx Match[Program Files, , No, No, Microsoft.\*] containing “Shared” from all machines’ and set max\_age\_seconds to 3600 on the Folder Name Search with RegEx Match sensor, then wait for result data to be complete, and get result data

### Step 1 - Authenticate to the SOAP API via /auth

- URL: <https://10.0.1.240:443/auth>
- HTTP Method: GET
- Elapsed Time: 0:00:00.007706
- Step 1 Request Body
- Step 1 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip, deflate",
4   "Connection": "keep-alive",
5   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
6   "password": "VGfuaXVtMjAxNSE=",
7   "username": "QWRtaW5pc3RyYXRvcg=="
8 }

```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-length": "135",
4   "content-type": "text/plain; charset=us-ascii"
5 }

```

## Step 2 - Get the server version via /info.json

- URL: <https://10.0.1.240:443/info.json>
- HTTP Method: GET
- Elapsed Time: 0:00:00.014205
- Step 2 Request Body
- Step 2 Response Body
- Request Headers:

```
1 {  
2   "Accept": "*/*",  
3   "Accept-Encoding": "gzip, deflate",  
4   "Connection": "keep-alive",  
5   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",  
6   "session": "1-6995-5602dcc91eee5e83246cd8affc00dae46fea49c258488a5456894e0bc6f91aac05fb2d73881976fa",  
7 }
```

- Response Headers:

```
1 {  
2   "connection": "keep-alive",  
3   "content-length": "88017",  
4   "content-type": "application/json"  
5 }
```

## Step 3 - Issue a GetObject to get the full object of a sensor for inclusion in a Select for a Question

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.002391
- Step 3 Request Body
- Step 3 Response Body
- Request Headers:

```
1 {  
2   "Accept": "*/*",  
3   "Accept-Encoding": "gzip",  
4   "Connection": "keep-alive",  
5   "Content-Length": "587",  
6   "Content-Type": "text/xml; charset=utf-8",  
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",  
8   "session": "1-6995-5602dcc91eee5e83246cd8affc00dae46fea49c258488a5456894e0bc6f91aac05fb2d73881976fa",  
9 }
```

- Response Headers:

```
1 {  
2   "connection": "keep-alive",  
3   "content-encoding": "gzip",  
4   "content-type": "text/xml; charset=UTF-8",
```

```

5  "transfer-encoding": "chunked"
6  }

```

#### Step 4 - Issue an AddObject to add a Question object

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.019317
- Step 4 Request Body
- Step 4 Response Body
- Request Headers:

```

1  {
2  "Accept": "*/*",
3  "Accept-Encoding": "gzip",
4  "Connection": "keep-alive",
5  "Content-Length": "1120",
6  "Content-Type": "text/xml; charset=utf-8",
7  "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8  "session": "1-6995-5602dcc91eee5e83246cd8affc00dae46fea49c258488a5456894e0bc6f91aac05fb2d73881976fa"
9  }

```

- Response Headers:

```

1  {
2  "connection": "keep-alive",
3  "content-length": "769",
4  "content-type": "text/xml; charset=UTF-8"
5  }

```

#### Step 5 - Issue a GetObject on the recently added object in order to get the full object

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.043114
- Step 5 Request Body
- Step 5 Response Body
- Request Headers:

```

1  {
2  "Accept": "*/*",
3  "Accept-Encoding": "gzip",
4  "Connection": "keep-alive",
5  "Content-Length": "494",
6  "Content-Type": "text/xml; charset=utf-8",
7  "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8  "session": "1-6995-5602dcc91eee5e83246cd8affc00dae46fea49c258488a5456894e0bc6f91aac05fb2d73881976fa"
9  }

```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }
```

#### Step 6 - Issue a GetResultInfo for a Question to check the current progress of answers

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.001575
- Step 6 Request Body
- Step 6 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "498",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6995-5602dcc91eee5e83246cd8affc00dae46fea49c258488a5456894e0bc6f91aac05fb2d73881976fa"
9 }
```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }
```

#### Step 7 - Issue a GetResultInfo for a Question to check the current progress of answers

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.002203
- Step 7 Request Body
- Step 7 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "498",
```

```

6  "Content-Type": "text/xml; charset=utf-8",
7  "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8  "session": "1-6995-5602dcc91eee5e83246cd8affc00dae46fea49c258488a5456894e0bc6f91aac05fb2d73881976f"
9  }

```

- Response Headers:

```

1  {
2  "connection": "keep-alive",
3  "content-encoding": "gzip",
4  "content-type": "text/xml; charset=UTF-8",
5  "transfer-encoding": "chunked"
6  }

```

#### Step 8 - Issue a GetResultInfo for a Question to check the current progress of answers

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.002143
- Step 8 Request Body
- Step 8 Response Body
- Request Headers:

```

1  {
2  "Accept": "*/*",
3  "Accept-Encoding": "gzip",
4  "Connection": "keep-alive",
5  "Content-Length": "498",
6  "Content-Type": "text/xml; charset=utf-8",
7  "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8  "session": "1-6995-5602dcc91eee5e83246cd8affc00dae46fea49c258488a5456894e0bc6f91aac05fb2d73881976f"
9  }

```

- Response Headers:

```

1  {
2  "connection": "keep-alive",
3  "content-encoding": "gzip",
4  "content-type": "text/xml; charset=UTF-8",
5  "transfer-encoding": "chunked"
6  }

```

#### Step 9 - Issue a GetResultInfo for a Question to check the current progress of answers

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.002126
- Step 9 Request Body
- Step 9 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "498",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6995-5602dcc91eee5e83246cd8affc00dae46fea49c258488a5456894e0bc6f91aac05fb2d73881976fa"
9 }
```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }
```

#### Step 10 - Issue a `GetResultInfo` for a Question to check the current progress of answers

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.002192
- Step 10 Request Body
- Step 10 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "498",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6995-5602dcc91eee5e83246cd8affc00dae46fea49c258488a5456894e0bc6f91aac05fb2d73881976fa"
9 }
```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }
```

#### Step 11 - Issue a `GetResultInfo` for a Question to check the current progress of answers

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.002215



- Step 11 Request Body
- Step 11 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "498",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6995-5602dcc91eee5e83246cd8affc00dae46fea49c258488a5456894e0bc6f91aac05fb2d73881976fa"
9 }

```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }

```

## Step 12 - Issue a GetResultInfo for a Question to check the current progress of answers

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.001611
- Step 12 Request Body
- Step 12 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "498",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6995-5602dcc91eee5e83246cd8affc00dae46fea49c258488a5456894e0bc6f91aac05fb2d73881976fa"
9 }

```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }

```

### Step 13 - Issue a `GetResultData` to get answers for a question

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.001594
- Step 13 Request Body
- Step 13 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "526",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6995-5602dcc91eee5e83246cd8affc00dae46fea49c258488a5456894e0bc6f91aac05fb2d73881976fa"
9 }
```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }
```

### Ask Manual Question Sensor With Filter And 3 Options

Ask the question ‘Get Operating System containing “Windows” from all machines’ and set `max_age_seconds` to 3600, `all_values_flag` to 1, and `ignore_case_flag` to 1 on the Operating System sensor, then wait for result data to be complete, and get result data

#### Step 1 - Authenticate to the SOAP API via `/auth`

- URL: <https://10.0.1.240:443/auth>
- HTTP Method: GET
- Elapsed Time: 0:00:00.006153
- Step 1 Request Body
- Step 1 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip, deflate",
4   "Connection": "keep-alive",
5   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
6   "password": "VGfUaXVtMjAxNSE=",
```

```

7  "username": "QWRtaW5pc3RyYXRvcg=="
8  }

```

- Response Headers:

```

1  {
2  "connection": "keep-alive",
3  "content-length": "135",
4  "content-type": "text/plain; charset=us-ascii"
5  }

```

## Step 2 - Get the server version via /info.json

- URL: <https://10.0.1.240:443/info.json>
- HTTP Method: GET
- Elapsed Time: 0:00:00.014004
- Step 2 Request Body
- Step 2 Response Body
- Request Headers:

```

1  {
2  "Accept": "*/*",
3  "Accept-Encoding": "gzip, deflate",
4  "Connection": "keep-alive",
5  "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
6  "session": "1-6996-83e914379f96cc824a16ccdd4491c1cdd37f4144d45536b9a8c9d16d094452da2c06e64cf496ea3"
7  }

```

- Response Headers:

```

1  {
2  "connection": "keep-alive",
3  "content-length": "88119",
4  "content-type": "application/json"
5  }

```

## Step 3 - Issue a GetObject to get the full object of a sensor for inclusion in a Select for a Question

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.002142
- Step 3 Request Body
- Step 3 Response Body
- Request Headers:

```

1  {
2  "Accept": "*/*",
3  "Accept-Encoding": "gzip",
4  "Connection": "keep-alive",
5  "Content-Length": "568",

```

```
6  "Content-Type": "text/xml; charset=utf-8",
7  "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8  "session": "1-6996-83e914379f96cc824a16ccdd4491c1cdd37f4144d45536b9a8c9d16d094452da2c06e64cf496ea3"
9  }
```

- Response Headers:

```
1  {
2  "connection": "keep-alive",
3  "content-encoding": "gzip",
4  "content-type": "text/xml; charset=UTF-8",
5  "transfer-encoding": "chunked"
6  }
```

#### Step 4 - Issue an AddObject to add a Question object

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.009947
- Step 4 Request Body
- Step 4 Response Body
- Request Headers:

```
1  {
2  "Accept": "*/*",
3  "Accept-Encoding": "gzip",
4  "Connection": "keep-alive",
5  "Content-Length": "861",
6  "Content-Type": "text/xml; charset=utf-8",
7  "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8  "session": "1-6996-83e914379f96cc824a16ccdd4491c1cdd37f4144d45536b9a8c9d16d094452da2c06e64cf496ea3"
9  }
```

- Response Headers:

```
1  {
2  "connection": "keep-alive",
3  "content-length": "769",
4  "content-type": "text/xml; charset=UTF-8"
5  }
```

#### Step 5 - Issue a GetObject on the recently added object in order to get the full object

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.013336
- Step 5 Request Body
- Step 5 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "494",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6996-83e914379f96cc824a16ccdd4491c1cdd37f4144d45536b9a8c9d16d094452da2c06e64cf496ea3"
9 }

```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }

```

#### Step 6 - Issue a GetResultInfo for a Question to check the current progress of answers

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.001812
- Step 6 Request Body
- Step 6 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "498",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6996-83e914379f96cc824a16ccdd4491c1cdd37f4144d45536b9a8c9d16d094452da2c06e64cf496ea3"
9 }

```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }

```

#### Step 7 - Issue a GetResultInfo for a Question to check the current progress of answers

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.002107

- Step 7 Request Body
- Step 7 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "498",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6996-83e914379f96cc824a16ccdd4491c1cdd37f4144d45536b9a8c9d16d094452da2c06e64cf496ea34",
9 }
```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }
```

#### Step 8 - Issue a GetResultInfo for a Question to check the current progress of answers

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.002105
- Step 8 Request Body
- Step 8 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "498",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6996-83e914379f96cc824a16ccdd4491c1cdd37f4144d45536b9a8c9d16d094452da2c06e64cf496ea34",
9 }
```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }
```

**Step 9 - Issue a GetResultData to get answers for a question**

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.001625
- Step 9 Request Body
- Step 9 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "526",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6996-83e914379f96cc824a16ccdd4491c1cdd37f4144d45536b9a8c9d16d094452da2c06e64cf496ea3"
9 }

```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }

```

**Ask Manual Question Complex Query1**

Ask the question ‘Get Computer Name and Folder Name Search with RegEx Match[Program Files, , No, No, Microsoft.\*, test] containing “Shared” from all machines with ( Operating System containing “Windows” or any Operating System not containing “Windows” )’ and set ignore\_case\_flag to 1 and or\_flag to 1 on the Operating System sensors on the right hand side of the question, then wait for result data to be complete, and get result data

**Step 1 - Authenticate to the SOAP API via /auth**

- URL: <https://10.0.1.240:443/auth>
- HTTP Method: GET
- Elapsed Time: 0:00:00.007867
- Step 1 Request Body
- Step 1 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip, deflate",
4   "Connection": "keep-alive",
5   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",

```

```
6  "password": "VGfuaXVtMjAxNSE=",
7  "username": "QWRtaW5pc3RyYXRvcg=="
8  }
```

- Response Headers:

```
1  {
2  "connection": "keep-alive",
3  "content-length": "135",
4  "content-type": "text/plain; charset=us-ascii"
5  }
```

### Step 2 - Get the server version via /info.json

- URL: <https://10.0.1.240:443/info.json>
- HTTP Method: GET
- Elapsed Time: 0:00:00.013975
- Step 2 Request Body
- Step 2 Response Body
- Request Headers:

```
1  {
2  "Accept": "*/*",
3  "Accept-Encoding": "gzip, deflate",
4  "Connection": "keep-alive",
5  "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
6  "session": "1-6997-76265f412df09760c20cf2d988647cf481da6d8fc9a003da180e2e7675a08d5ff29d0cbdaf1e4e2a"
7  }
```

- Response Headers:

```
1  {
2  "connection": "keep-alive",
3  "content-length": "88120",
4  "content-type": "application/json"
5  }
```

### Step 3 - Issue a GetObject to get the full object of a sensor for inclusion in a Select for a Question

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.002020
- Step 3 Request Body
- Step 3 Response Body
- Request Headers:

```
1  {
2  "Accept": "*/*",
3  "Accept-Encoding": "gzip",
4  "Connection": "keep-alive",
```



```

5  "Content-Length": "565",
6  "Content-Type": "text/xml; charset=utf-8",
7  "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8  "session": "1-6997-76265f412df09760c20cf2d988647cf481da6d8fc9a003da180e2e7675a08d5ff29d0cbdaf1e4e2"
9  }

```

- Response Headers:

```

1  {
2  "connection": "keep-alive",
3  "content-encoding": "gzip",
4  "content-type": "text/xml; charset=UTF-8",
5  "transfer-encoding": "chunked"
6  }

```

#### Step 4 - Issue a GetObject to get the full object of a sensor for inclusion in a Select for a Question

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.002436
- Step 4 Request Body
- Step 4 Response Body
- Request Headers:

```

1  {
2  "Accept": "*/*",
3  "Accept-Encoding": "gzip",
4  "Connection": "keep-alive",
5  "Content-Length": "587",
6  "Content-Type": "text/xml; charset=utf-8",
7  "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8  "session": "1-6997-76265f412df09760c20cf2d988647cf481da6d8fc9a003da180e2e7675a08d5ff29d0cbdaf1e4e2"
9  }

```

- Response Headers:

```

1  {
2  "connection": "keep-alive",
3  "content-encoding": "gzip",
4  "content-type": "text/xml; charset=UTF-8",
5  "transfer-encoding": "chunked"
6  }

```

#### Step 5 - Issue a GetObject to get the full object of a sensor for inclusion in a Group for a Question

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.001824
- Step 5 Request Body
- Step 5 Response Body

- Request Headers:

```
1 {  
2   "Accept": "*/*",  
3   "Accept-Encoding": "gzip",  
4   "Connection": "keep-alive",  
5   "Content-Length": "568",  
6   "Content-Type": "text/xml; charset=utf-8",  
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",  
8   "session": "1-6997-76265f412df09760c20cf2d988647cf481da6d8fc9a003da180e2e7675a08d5ff29d0cbdaf1e4e2a",  
9 }
```

- Response Headers:

```
1 {  
2   "connection": "keep-alive",  
3   "content-encoding": "gzip",  
4   "content-type": "text/xml; charset=UTF-8",  
5   "transfer-encoding": "chunked"  
6 }
```

#### Step 6 - Issue a `GetObject` to get the full object of a sensor for inclusion in a Group for a Question

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.001954
- Step 6 Request Body
- Step 6 Response Body
- Request Headers:

```
1 {  
2   "Accept": "*/*",  
3   "Accept-Encoding": "gzip",  
4   "Connection": "keep-alive",  
5   "Content-Length": "568",  
6   "Content-Type": "text/xml; charset=utf-8",  
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",  
8   "session": "1-6997-76265f412df09760c20cf2d988647cf481da6d8fc9a003da180e2e7675a08d5ff29d0cbdaf1e4e2a",  
9 }
```

- Response Headers:

```
1 {  
2   "connection": "keep-alive",  
3   "content-encoding": "gzip",  
4   "content-type": "text/xml; charset=UTF-8",  
5   "transfer-encoding": "chunked"  
6 }
```

#### Step 7 - Issue an `AddObject` to add a Question object

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST

- Elapsed Time: 0:00:00.027102
- Step 7 Request Body
- Step 7 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "1678",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6997-76265f412df09760c20cf2d988647cf481da6d8fc9a003da180e2e7675a08d5ff29d0cbdaf1e4e2a"
9 }

```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-length": "769",
4   "content-type": "text/xml; charset=UTF-8"
5 }

```

#### Step 8 - Issue a GetObject on the recently added object in order to get the full object

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.041812
- Step 8 Request Body
- Step 8 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "494",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6997-76265f412df09760c20cf2d988647cf481da6d8fc9a003da180e2e7675a08d5ff29d0cbdaf1e4e2a"
9 }

```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }

```

### Step 9 - Issue a GetResultInfo for a Question to check the current progress of answers

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.001588
- Step 9 Request Body
- Step 9 Response Body
- Request Headers:

```
1 {  
2   "Accept": "*/*",  
3   "Accept-Encoding": "gzip",  
4   "Connection": "keep-alive",  
5   "Content-Length": "498",  
6   "Content-Type": "text/xml; charset=utf-8",  
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",  
8   "session": "1-6997-76265f412df09760c20cf2d988647cf481da6d8fc9a003da180e2e7675a08d5ff29d0cbdaf1e4e2a",  
9 }
```

- Response Headers:

```
1 {  
2   "connection": "keep-alive",  
3   "content-encoding": "gzip",  
4   "content-type": "text/xml; charset=UTF-8",  
5   "transfer-encoding": "chunked"  
6 }
```

### Step 10 - Issue a GetResultInfo for a Question to check the current progress of answers

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.001816
- Step 10 Request Body
- Step 10 Response Body
- Request Headers:

```
1 {  
2   "Accept": "*/*",  
3   "Accept-Encoding": "gzip",  
4   "Connection": "keep-alive",  
5   "Content-Length": "498",  
6   "Content-Type": "text/xml; charset=utf-8",  
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",  
8   "session": "1-6997-76265f412df09760c20cf2d988647cf481da6d8fc9a003da180e2e7675a08d5ff29d0cbdaf1e4e2a",  
9 }
```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }

```

### Step 11 - Issue a GetResultInfo for a Question to check the current progress of answers

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.002226
- Step 11 Request Body
- Step 11 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "498",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6997-76265f412df09760c20cf2d988647cf481da6d8fc9a003da180e2e7675a08d5ff29d0cbdaf1e4e2"
9 }

```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }

```

### Step 12 - Issue a GetResultInfo for a Question to check the current progress of answers

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.002222
- Step 12 Request Body
- Step 12 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "498",

```

```
6  "Content-Type": "text/xml; charset=utf-8",
7  "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8  "session": "1-6997-76265f412df09760c20cf2d988647cf481da6d8fc9a003da180e2e7675a08d5ff29d0cbdaf1e4e2"
9  }
```

- Response Headers:

```
1  {
2  "connection": "keep-alive",
3  "content-encoding": "gzip",
4  "content-type": "text/xml; charset=UTF-8",
5  "transfer-encoding": "chunked"
6  }
```

### Step 13 - Issue a GetResultInfo for a Question to check the current progress of answers

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.002423
- Step 13 Request Body
- Step 13 Response Body
- Request Headers:

```
1  {
2  "Accept": "*/*",
3  "Accept-Encoding": "gzip",
4  "Connection": "keep-alive",
5  "Content-Length": "498",
6  "Content-Type": "text/xml; charset=utf-8",
7  "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8  "session": "1-6997-76265f412df09760c20cf2d988647cf481da6d8fc9a003da180e2e7675a08d5ff29d0cbdaf1e4e2"
9  }
```

- Response Headers:

```
1  {
2  "connection": "keep-alive",
3  "content-encoding": "gzip",
4  "content-type": "text/xml; charset=UTF-8",
5  "transfer-encoding": "chunked"
6  }
```

### Step 14 - Issue a GetResultInfo for a Question to check the current progress of answers

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.002290
- Step 14 Request Body
- Step 14 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "498",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6997-76265f412df09760c20cf2d988647cf481da6d8fc9a003da180e2e7675a08d5ff29d0cbdaf1e4e2a"
9 }

```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }

```

### Step 15 - Issue a GetResultData to get answers for a question

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.002092
- Step 15 Request Body
- Step 15 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "526",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6997-76265f412df09760c20cf2d988647cf481da6d8fc9a003da180e2e7675a08d5ff29d0cbdaf1e4e2a"
9 }

```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }

```

### Ask Manual Question Complex Query2

Ask the question ‘Get Computer Name and Last Logged In User and Installed Applications containing “Google (Search|Chrome)” from all machines with Installed Applications containing “Google (Search|Chrome)”’ and set ignore\_case\_flag to 1 and or\_flag to 1 on the Installed Applications sensors on the right hand side of the question, then wait for result data to be complete, and get result data

### Step 1 - Authenticate to the SOAP API via /auth

- URL: <https://10.0.1.240:443/auth>
- HTTP Method: GET
- Elapsed Time: 0:00:00.007472
- Step 1 Request Body
- Step 1 Response Body
- Request Headers:

```
1 {  
2   "Accept": "*/*",  
3   "Accept-Encoding": "gzip, deflate",  
4   "Connection": "keep-alive",  
5   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",  
6   "password": "VGfuaXVtMjAxNSE=",  
7   "username": "QWRtaW5pc3RyYXRvcg=="  
8 }
```

- Response Headers:

```
1 {  
2   "connection": "keep-alive",  
3   "content-length": "135",  
4   "content-type": "text/plain; charset=us-ascii"  
5 }
```

### Step 2 - Get the server version via /info.json

- URL: <https://10.0.1.240:443/info.json>
- HTTP Method: GET
- Elapsed Time: 0:00:00.014728
- Step 2 Request Body
- Step 2 Response Body
- Request Headers:

```
1 {  
2   "Accept": "*/*",  
3   "Accept-Encoding": "gzip, deflate",  
4   "Connection": "keep-alive",  
5   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",  
6   "session": "1-6998-ec9c0cc41fcb67257e819c5c8f1ff7f7dc5028d8ee80f6d3ccc0d53f196486a1ed62c4b1f4f045a"  
7 }
```

- Response Headers:

```
1 {  
2   "connection": "keep-alive",  
3   "content-length": "88223",  
4   "content-type": "application/json"  
5 }
```



**Step 3 - Issue a GetObject to get the full object of a sensor for inclusion in a Select for a Question**

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.002128
- Step 3 Request Body
- Step 3 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "565",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6998-ec9c0cc41fcb67257e819c5c8f1ff7f7dc5028d8ee80f6d3ccc0d53f196486aled62c4b1f4f045a"
9 }

```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }

```

**Step 4 - Issue a GetObject to get the full object of a sensor for inclusion in a Select for a Question**

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.002414
- Step 4 Request Body
- Step 4 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "571",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6998-ec9c0cc41fcb67257e819c5c8f1ff7f7dc5028d8ee80f6d3ccc0d53f196486aled62c4b1f4f045a"
9 }

```

- Response Headers:

```
1 {  
2   "connection": "keep-alive",  
3   "content-encoding": "gzip",  
4   "content-type": "text/xml; charset=UTF-8",  
5   "transfer-encoding": "chunked"  
6 }
```

#### Step 5 - Issue a GetObject to get the full object of a sensor for inclusion in a Select for a Question

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.002335
- Step 5 Request Body
- Step 5 Response Body
- Request Headers:

```
1 {  
2   "Accept": "*/*",  
3   "Accept-Encoding": "gzip",  
4   "Connection": "keep-alive",  
5   "Content-Length": "574",  
6   "Content-Type": "text/xml; charset=utf-8",  
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",  
8   "session": "1-6998-ec9c0cc41fcb67257e819c5c8f1ff7f7dc5028d8ee80f6d3ccc0d53f196486aled62c4b1f4f045a"  
9 }
```

- Response Headers:

```
1 {  
2   "connection": "keep-alive",  
3   "content-encoding": "gzip",  
4   "content-type": "text/xml; charset=UTF-8",  
5   "transfer-encoding": "chunked"  
6 }
```

#### Step 6 - Issue a GetObject to get the full object of a sensor for inclusion in a Group for a Question

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.002212
- Step 6 Request Body
- Step 6 Response Body
- Request Headers:

```
1 {  
2   "Accept": "*/*",  
3   "Accept-Encoding": "gzip",  
4   "Connection": "keep-alive",  
5   "Content-Length": "574",
```

```

6  "Content-Type": "text/xml; charset=utf-8",
7  "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8  "session": "1-6998-ec9c0cc41fcb67257e819c5c8f1ff7f7dc5028d8ee80f6d3ccc0d53f196486aled62c4b1f4f045a
9  }

```

- Response Headers:

```

1  {
2  "connection": "keep-alive",
3  "content-encoding": "gzip",
4  "content-type": "text/xml; charset=UTF-8",
5  "transfer-encoding": "chunked"
6  }

```

### Step 7 - Issue an AddObject to add a Question object

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.012807
- Step 7 Request Body
- Step 7 Response Body
- Request Headers:

```

1  {
2  "Accept": "*/*",
3  "Accept-Encoding": "gzip",
4  "Connection": "keep-alive",
5  "Content-Length": "1174",
6  "Content-Type": "text/xml; charset=utf-8",
7  "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8  "session": "1-6998-ec9c0cc41fcb67257e819c5c8f1ff7f7dc5028d8ee80f6d3ccc0d53f196486aled62c4b1f4f045a
9  }

```

- Response Headers:

```

1  {
2  "connection": "keep-alive",
3  "content-length": "769",
4  "content-type": "text/xml; charset=UTF-8"
5  }

```

### Step 8 - Issue a GetObject on the recently added object in order to get the full object

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.012531
- Step 8 Request Body
- Step 8 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "494",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6998-ec9c0cc41fcb67257e819c5c8f1ff7f7dc5028d8ee80f6d3ccc0d53f196486aled62c4b1f4f045a"
9 }
```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }
```

#### Step 9 - Issue a GetResultInfo for a Question to check the current progress of answers

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.001716
- Step 9 Request Body
- Step 9 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "498",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6998-ec9c0cc41fcb67257e819c5c8f1ff7f7dc5028d8ee80f6d3ccc0d53f196486aled62c4b1f4f045a"
9 }
```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }
```

#### Step 10 - Issue a GetResultInfo for a Question to check the current progress of answers

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.001635

- Step 10 Request Body
- Step 10 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "498",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6998-ec9c0cc41fcb67257e819c5c8f1ff7f7dc5028d8ee80f6d3ccc0d53f196486aled62c4b1f4f045a",
9 }

```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }

```

### Step 11 - Issue a GetResultInfo for a Question to check the current progress of answers

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.002136
- Step 11 Request Body
- Step 11 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "498",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6998-ec9c0cc41fcb67257e819c5c8f1ff7f7dc5028d8ee80f6d3ccc0d53f196486aled62c4b1f4f045a",
9 }

```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }

```

### Step 12 - Issue a `GetResultData` to get answers for a question

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.001741
- Step 12 Request Body
- Step 12 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "526",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-6998-ec9c0cc41fcb67257e819c5c8f1ff7f7dc5028d8ee80f6d3ccc0d53f196486aled62c4b1f4f045a"
9 }
```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }
```

### Ask Saved Question Refresh Data

Get the Saved Question object for Installed Applications, ask the server to refresh the data available, wait for the new question spawned to complete results, then get the latest result data available for that Saved Question

### Step 1 - Authenticate to the SOAP API via `/auth`

- URL: <https://10.0.1.240:443/auth>
- HTTP Method: GET
- Elapsed Time: 0:00:00.006322
- Step 1 Request Body
- Step 1 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip, deflate",
4   "Connection": "keep-alive",
5   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
6   "password": "VGfuaXVtMjAxNSE=",
7   "username": "QWRtaW5pc3RyYXRvcg=="
8 }
```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-length": "135",
4   "content-type": "text/plain; charset=us-ascii"
5 }
```

## Step 2 - Get the server version via /info.json

- URL: <https://10.0.1.240:443/info.json>
- HTTP Method: GET
- Elapsed Time: 0:00:00.014431
- Step 2 Request Body
- Step 2 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip, deflate",
4   "Connection": "keep-alive",
5   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
6   "session": "1-7000-0b3a77745c282aea1f47fc9b35ea6d94e18e7f95336eb79657a3b449756d4ce45f272fff7ec4e30a"
7 }
```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-length": "88325",
4   "content-type": "application/json"
5 }
```

## Step 3 - Issue a GetObject to find saved question objects

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.013249
- Step 3 Request Body
- Step 3 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "527",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
```

```
8  "session": "1-7000-0b3a77745c282aea1f47fc9b35ea6d94e18e7f95336eb79657a3b449756d4ce45f272fff7ec4e30a
9  }
```

- Response Headers:

```
1  {
2  "connection": "keep-alive",
3  "content-encoding": "gzip",
4  "content-type": "text/xml; charset=UTF-8",
5  "transfer-encoding": "chunked"
6  }
```

#### Step 4 - Issue a `GetObject` to get the full object of the last question asked by a saved question

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.003820
- Step 4 Request Body
- Step 4 Response Body
- Request Headers:

```
1  {
2  "Accept": "*/*",
3  "Accept-Encoding": "gzip",
4  "Connection": "keep-alive",
5  "Content-Length": "21616",
6  "Content-Type": "text/xml; charset=utf-8",
7  "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8  "session": "1-7000-0b3a77745c282aea1f47fc9b35ea6d94e18e7f95336eb79657a3b449756d4ce45f272fff7ec4e30a
9  }
```

- Response Headers:

```
1  {
2  "connection": "keep-alive",
3  "content-encoding": "gzip",
4  "content-type": "text/xml; charset=UTF-8",
5  "transfer-encoding": "chunked"
6  }
```

#### Step 5 - Issue a `GetResultInfo` for a saved question in order to issue a new question, which refreshes the data for that saved question

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.006532
- Step 5 Request Body
- Step 5 Response Body
- Request Headers:



```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "542",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-7000-0b3a77745c282aea1f47fc9b35ea6d94e18e7f95336eb79657a3b449756d4ce45f272fff7ec4e30a"
9 }

```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }

```

#### Step 6 - Issue a GetObject for the saved question in order get the ID of the newly asked question

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.005835
- Step 6 Request Body
- Step 6 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "538",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-7000-0b3a77745c282aea1f47fc9b35ea6d94e18e7f95336eb79657a3b449756d4ce45f272fff7ec4e30a"
9 }

```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }

```

#### Step 7 - Issue a GetObject to get the full object of the last question asked by a saved question

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.002829

- Step 7 Request Body
- Step 7 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "942",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-7000-0b3a77745c282aea1f47fc9b35ea6d94e18e7f95336eb79657a3b449756d4ce45f272fff7ec4e30a"
9 }
```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }
```

#### Step 8 - Issue a GetResultInfo for a Question to check the current progress of answers

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.001634
- Step 8 Request Body
- Step 8 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "498",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-7000-0b3a77745c282aea1f47fc9b35ea6d94e18e7f95336eb79657a3b449756d4ce45f272fff7ec4e30a"
9 }
```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }
```

**Step 9 - Issue a GetResultInfo for a Question to check the current progress of answers**

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.002004
- Step 9 Request Body
- Step 9 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "498",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-7000-0b3a77745c282aea1f47fc9b35ea6d94e18e7f95336eb79657a3b449756d4ce45f272fff7ec4e30a"
9 }

```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }

```

**Step 10 - Issue a GetResultInfo for a Question to check the current progress of answers**

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.002525
- Step 10 Request Body
- Step 10 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "498",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-7000-0b3a77745c282aea1f47fc9b35ea6d94e18e7f95336eb79657a3b449756d4ce45f272fff7ec4e30a"
9 }

```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }
```

#### Step 11 - Issue a GetResultData to get the answers for the last asked question of this saved question

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.004172
- Step 11 Request Body
- Step 11 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "526",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-7000-0b3a77745c282aea1f47fc9b35ea6d94e18e7f95336eb79657a3b449756d4ce45f272fff7ec4e30a"
9 }
```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }
```

#### Ask Saved Question By Name

Get the Saved Question object for Installed Applications then get the latest result data available for that Saved Question

#### Step 1 - Authenticate to the SOAP API via /auth

- URL: <https://10.0.1.240:443/auth>
- HTTP Method: GET
- Elapsed Time: 0:00:00.006143
- Step 1 Request Body
- Step 1 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip, deflate",
4   "Connection": "keep-alive",
5   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
6   "password": "VGFuaXVtMjAxNSE=",
7   "username": "QWRtaW5pc3RyYXRvcg=="
8 }

```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-length": "135",
4   "content-type": "text/plain; charset=us-ascii"
5 }

```

## Step 2 - Get the server version via /info.json

- URL: <https://10.0.1.240:443/info.json>
- HTTP Method: GET
- Elapsed Time: 0:00:00.013685
- Step 2 Request Body
- Step 2 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip, deflate",
4   "Connection": "keep-alive",
5   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
6   "session": "1-7001-95ad8b27d21e8c51a44e4bb4f8e125350dd62c00fb17dbb01569a4b251a5702950f46c77201f743"
7 }

```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-length": "88324",
4   "content-type": "application/json"
5 }

```

## Step 3 - Issue a GetObject to find saved question objects

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.012031
- Step 3 Request Body
- Step 3 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "527",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-7001-95ad8b27d21e8c51a44e4bb4f8e125350dd62c00fb17dbb01569a4b251a5702950f46c77201f7432"
9 }
```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }
```

#### Step 4 - Issue a GetObject to get the full object of the last question asked by a saved question

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.003569
- Step 4 Request Body
- Step 4 Response Body
- Request Headers:

```
1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "21616",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-7001-95ad8b27d21e8c51a44e4bb4f8e125350dd62c00fb17dbb01569a4b251a5702950f46c77201f7432"
9 }
```

- Response Headers:

```
1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }
```

#### Step 5 - Issue a GetResultData to get the answers for the last asked question of this saved question

- URL: <https://10.0.1.240:443/soap>
- HTTP Method: POST
- Elapsed Time: 0:00:00.004121

- Step 5 Request Body
- Step 5 Response Body
- Request Headers:

```

1 {
2   "Accept": "*/*",
3   "Accept-Encoding": "gzip",
4   "Connection": "keep-alive",
5   "Content-Length": "526",
6   "Content-Type": "text/xml; charset=utf-8",
7   "User-Agent": "python-requests/2.6.0 CPython/2.7.10 Darwin/14.5.0",
8   "session": "1-7001-95ad8b27d21e8c51a44e4bb4f8e125350dd62c00fb17dbb01569a4b251a5702950f46c77201f7432"
9 }

```

- Response Headers:

```

1 {
2   "connection": "keep-alive",
3   "content-encoding": "gzip",
4   "content-type": "text/xml; charset=UTF-8",
5   "transfer-encoding": "chunked"
6 }

```

## 1.3 taniumpy package

A python package that handles the serialization/deserialization of XML SOAP requests/responses from Taniump to/from python objects.

```

class taniumpy.object_types.action.Action
    Bases: taniumpy.object_types.base.BaseType
    _soap_tag = 'action'

class taniumpy.object_types.action_list.ActionList
    Bases: taniumpy.object_types.base.BaseType
    _soap_tag = 'actions'

class taniumpy.object_types.action_list_info.ActionListInfo
    Bases: taniumpy.object_types.base.BaseType
    _soap_tag = 'info'

class taniumpy.object_types.action_stop.ActionStop
    Bases: taniumpy.object_types.base.BaseType
    _soap_tag = 'action_stop'

class taniumpy.object_types.action_stop_list.ActionStopList
    Bases: taniumpy.object_types.base.BaseType
    _soap_tag = 'action_stops'

class taniumpy.object_types.archived_question.ArchivedQuestion
    Bases: taniumpy.object_types.base.BaseType
    _soap_tag = 'archived_question'

class taniumpy.object_types.archived_question_list.ArchivedQuestionList
    Bases: taniumpy.object_types.base.BaseType

```

```
_soap_tag = 'archived_questions'
```

```
class taniumpy.object_types.audit_data.AuditData  
    Bases: taniumpy.object_types.base.BaseType
```

```
_soap_tag = 'audit_data'
```

```
class taniumpy.object_types.base.BaseType (simple_properties,           complex_properties,  
                                           list_properties)  
    Bases: object
```

```
classmethod _from_json (jsonable)  
    Private helper to parse from JSON after type is instantiated
```

```
_soap_tag = None
```

```
append (n)  
    Allow adding to list.  
  
    Only supported on types that have a single property that is in list_properties
```

```
explode_json (val)
```

```
flatten_jsonable (val, prefix)
```

```
classmethod fromSOAPBody (body)  
    Parse body (text) and produce Python tanium objects.  
  
    This method assumes a single result_object, which may be a list or a single object.
```

```
classmethod fromSOAPElement (el)
```

```
static from_jsonable (jsonable)  
    Inverse of to_jsonable, with explode_json_string_values=False.  
  
    This can be used to import objects from serialized JSON. This JSON should come from 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 or a list of BaseType
```

```
to_jsonable (explode_json_string_values=False, include_type=True)
```



**static write\_csv** (*fd, val, explode\_json\_string\_values=False, \*\*kwargs*)

Write 'val' to CSV. val can be a BaseType instance or a list of BaseType

This does a two-pass, calling to\_flat\_dict for each object, then finding the union of all headers, then writing out the value of each column for each object sorted by header name

explode\_json\_string\_values attempts to see if any of the str values are parseable by json.loads, and if so treat each property as a column value

fd is a file-like object

**exception** `taniumpy.object_types.base.IncorrectTypeException` (*property, expected, actual*)

Bases: `exceptions.Exception`

Raised when a property is not of the expected type

**class** `taniumpy.object_types.cache_filter.CacheFilter`

Bases: `taniumpy.object_types.base.BaseType`

`_soap_tag = 'filter'`

**class** `taniumpy.object_types.cache_filter_list.CacheFilterList`

Bases: `taniumpy.object_types.base.BaseType`

`_soap_tag = 'cache_filters'`

**class** `taniumpy.object_types.cache_info.CacheInfo`

Bases: `taniumpy.object_types.base.BaseType`

`_soap_tag = 'cache_info'`

**class** `taniumpy.object_types.client_count.ClientCount`

Bases: `taniumpy.object_types.base.BaseType`

`_soap_tag = 'client_count'`

**class** `taniumpy.object_types.client_status.ClientStatus`

Bases: `taniumpy.object_types.base.BaseType`

`_soap_tag = 'client_status'`

**class** `taniumpy.object_types.column.Column`

Bases: `object`

**classmethod** `fromSOAPElement (el)`

**class** `taniumpy.object_types.column_set.ColumnSet`

Bases: `object`

**classmethod** `fromSOAPElement (el)`

**class** `taniumpy.object_types.computer_group.ComputerGroup`

Bases: `taniumpy.object_types.base.BaseType`

`_soap_tag = 'computer_group'`

**class** `taniumpy.object_types.computer_group_list.ComputerGroupList`

Bases: `taniumpy.object_types.base.BaseType`

`_soap_tag = 'computer_groups'`

**class** `taniumpy.object_types.computer_group_spec.ComputerGroupSpec`

Bases: `taniumpy.object_types.base.BaseType`

`_soap_tag = 'computer_spec'`

```
class taniumpy.object_types.computer_spec_list.ComputerSpecList
    Bases: taniumpy.object_types.base.BaseType
    _soap_tag = 'computer_specs'

class taniumpy.object_types.error_list.ErrorList
    Bases: taniumpy.object_types.base.BaseType
    _soap_tag = 'errors'

class taniumpy.object_types.filter.Filter
    Bases: taniumpy.object_types.base.BaseType
    _soap_tag = 'filter'

class taniumpy.object_types.filter_list.FilterList
    Bases: taniumpy.object_types.base.BaseType
    _soap_tag = 'filters'

class taniumpy.object_types.group.Group
    Bases: taniumpy.object_types.base.BaseType
    _soap_tag = 'group'

class taniumpy.object_types.group_list.GroupList
    Bases: taniumpy.object_types.base.BaseType
    _soap_tag = 'groups'

class taniumpy.object_types.metadata_item.MetadataItem
    Bases: taniumpy.object_types.base.BaseType
    _soap_tag = 'item'

class taniumpy.object_types.metadata_list.MetadataList
    Bases: taniumpy.object_types.base.BaseType
    _soap_tag = 'metadata'

class taniumpy.object_types.object_list.ObjectList
    Bases: taniumpy.object_types.base.BaseType
    _soap_tag = 'object_list'

class taniumpy.object_types.options.Options
    Bases: taniumpy.object_types.base.BaseType
    _soap_tag = 'options'

class taniumpy.object_types.package_file.PackageFile
    Bases: taniumpy.object_types.base.BaseType
    _soap_tag = 'file'

class taniumpy.object_types.package_file_list.PackageFileList
    Bases: taniumpy.object_types.base.BaseType
    _soap_tag = 'package_files'

class taniumpy.object_types.package_file_status.PackageFileStatus
    Bases: taniumpy.object_types.base.BaseType
    _soap_tag = 'status'

class taniumpy.object_types.package_file_status_list.PackageFileStatusList
    Bases: taniumpy.object_types.base.BaseType
```

```

    _soap_tag = 'file_status'
class taniumpy.object_types.package_file_template.PackageFileTemplate
    Bases: taniumpy.object_types.base.BaseType
    _soap_tag = 'file_template'
class taniumpy.object_types.package_file_template_list.PackageFileTemplateList
    Bases: taniumpy.object_types.base.BaseType
    _soap_tag = 'file_templates'
class taniumpy.object_types.package_spec.PackageSpec
    Bases: taniumpy.object_types.base.BaseType
    _soap_tag = 'package_spec'
class taniumpy.object_types.package_spec_list.PackageSpecList
    Bases: taniumpy.object_types.base.BaseType
    _soap_tag = 'package_specs'
class taniumpy.object_types.parameter.Parameter
    Bases: taniumpy.object_types.base.BaseType
    _soap_tag = 'parameter'
class taniumpy.object_types.parameter_list.ParameterList
    Bases: taniumpy.object_types.base.BaseType
    _soap_tag = 'parameters'
class taniumpy.object_types.parse_job.ParseJob
    Bases: taniumpy.object_types.base.BaseType
    _soap_tag = 'parse_job'
class taniumpy.object_types.parse_job_list.ParseJobList
    Bases: taniumpy.object_types.base.BaseType
    _soap_tag = 'parse_jobs'
class taniumpy.object_types.parse_result.ParseResult
    Bases: taniumpy.object_types.base.BaseType
    _soap_tag = 'parse_result'
class taniumpy.object_types.parse_result_group.ParseResultGroup
    Bases: taniumpy.object_types.base.BaseType
    _soap_tag = 'parse_result_group'
class taniumpy.object_types.parse_result_group_list.ParseResultGroupList
    Bases: taniumpy.object_types.base.BaseType
    _soap_tag = 'parse_result_groups'
class taniumpy.object_types.parse_result_list.ParseResultList
    Bases: taniumpy.object_types.base.BaseType
    _soap_tag = 'parse_results'
class taniumpy.object_types.permission_list.PermissionList
    Bases: taniumpy.object_types.base.BaseType
    _soap_tag = 'permissions'

```

```
class taniumpy.object_types.plugin.Plugin
    Bases: taniumpy.object_types.base.BaseType
    _soap_tag = 'plugin'

class taniumpy.object_types.plugin_argument.PluginArgument
    Bases: taniumpy.object_types.base.BaseType
    _soap_tag = 'argument'

class taniumpy.object_types.plugin_argument_list.PluginArgumentList
    Bases: taniumpy.object_types.base.BaseType
    _soap_tag = 'arguments'

class taniumpy.object_types.plugin_command_list.PluginCommandList
    Bases: taniumpy.object_types.base.BaseType
    _soap_tag = 'commands'

class taniumpy.object_types.plugin_list.PluginList
    Bases: taniumpy.object_types.base.BaseType
    _soap_tag = 'plugins'

class taniumpy.object_types.plugin_schedule.PluginSchedule
    Bases: taniumpy.object_types.base.BaseType
    _soap_tag = 'plugin_schedule'

class taniumpy.object_types.plugin_schedule_list.PluginScheduleList
    Bases: taniumpy.object_types.base.BaseType
    _soap_tag = 'plugin_schedules'

class taniumpy.object_types.plugin_sql.PluginSql
    Bases: taniumpy.object_types.base.BaseType
    _soap_tag = 'sql_response'

class taniumpy.object_types.plugin_sql_column.PluginSqlColumn
    Bases: taniumpy.object_types.base.BaseType
    _soap_tag = 'columns'

class taniumpy.object_types.plugin_sql_result.PluginSqlResult
    Bases: taniumpy.object_types.base.BaseType
    _soap_tag = 'result_row'

class taniumpy.object_types.question.Question
    Bases: taniumpy.object_types.base.BaseType
    _soap_tag = 'question'

class taniumpy.object_types.question_list.QuestionList
    Bases: taniumpy.object_types.base.BaseType
    _soap_tag = 'questions'

class taniumpy.object_types.question_list_info.QuestionListInfo
    Bases: taniumpy.object_types.base.BaseType
    _soap_tag = 'info'
```

```

class taniumpy.object_types.result_info.ResultInfo
    Bases: object

    Wrap the result of GetResultInfo

    classmethod fromSOAPElement (el)
        Deserialize a ResultInfo from a result_info SOAPElement

        Assumes all properties are integer values (true today)

class taniumpy.object_types.result_set.ResultSet
    Bases: object

    Wrap the result of GetResultData

    classmethod fromSOAPElement (el)
        Deserialize a ResultSet from a result_set SOAPElement

    static to_json (jsonable, **kwargs)
        Convert to a json string.

        jsonable must be a ResultSet instance

    to_jsonable (**kwargs)

    static write_csv (fd, val, **kwargs)

class taniumpy.object_types.row.Row (columns)
    Bases: object

    A row in a result set.

    Values are stored in column order, also accessible by key using []

    classmethod fromSOAPElement (el, columns)

class taniumpy.object_types.saved_action.SavedAction
    Bases: taniumpy.object_types.base.BaseType

    _soap_tag = 'saved_action'

class taniumpy.object_types.saved_action_approval.SavedActionApproval
    Bases: taniumpy.object_types.base.BaseType

    _soap_tag = 'saved_action_approval'

class taniumpy.object_types.saved_action_list.SavedActionList
    Bases: taniumpy.object_types.base.BaseType

    _soap_tag = 'saved_actions'

class taniumpy.object_types.saved_action_policy.SavedActionPolicy
    Bases: taniumpy.object_types.base.BaseType

    _soap_tag = 'policy'

class taniumpy.object_types.saved_action_row_id_list.SavedActionRowIdList
    Bases: taniumpy.object_types.base.BaseType

    _soap_tag = 'row_ids'

class taniumpy.object_types.saved_question.SavedQuestion
    Bases: taniumpy.object_types.base.BaseType

    _soap_tag = 'saved_question'

```

```
class taniumpy.object_types.saved_question_list.SavedQuestionList
    Bases: taniumpy.object_types.base.BaseType
    _soap_tag = 'saved_questions'

class taniumpy.object_types.select.Select
    Bases: taniumpy.object_types.base.BaseType
    _soap_tag = 'select'

class taniumpy.object_types.select_list.SelectList
    Bases: taniumpy.object_types.base.BaseType
    _soap_tag = 'selects'

class taniumpy.object_types.sensor.Sensor
    Bases: taniumpy.object_types.base.BaseType
    _soap_tag = 'sensor'

class taniumpy.object_types.sensor_list.SensorList
    Bases: taniumpy.object_types.base.BaseType
    _soap_tag = 'sensors'

class taniumpy.object_types.sensor_query.SensorQuery
    Bases: taniumpy.object_types.base.BaseType
    _soap_tag = 'query'

class taniumpy.object_types.sensor_query_list.SensorQueryList
    Bases: taniumpy.object_types.base.BaseType
    _soap_tag = 'queries'

class taniumpy.object_types.string_hint_list.StringHintList
    Bases: taniumpy.object_types.base.BaseType
    _soap_tag = 'string_hints'

class taniumpy.object_types.sensor_subcolumn.SensorSubcolumn
    Bases: taniumpy.object_types.base.BaseType
    _soap_tag = 'subcolumn'

class taniumpy.object_types.sensor_subcolumn_list.SensorSubcolumnList
    Bases: taniumpy.object_types.base.BaseType
    _soap_tag = 'subcolumns'

class taniumpy.object_types.soap_error.SoapError
    Bases: taniumpy.object_types.base.BaseType
    _soap_tag = 'soap_error'

class taniumpy.object_types.system_setting.SystemSetting
    Bases: taniumpy.object_types.base.BaseType
    _soap_tag = 'system_setting'

class taniumpy.object_types.system_setting_list.SystemSettingList
    Bases: taniumpy.object_types.base.BaseType
    _soap_tag = 'system_settings'

class taniumpy.object_types.system_status_aggregate.SystemStatusAggregate
    Bases: taniumpy.object_types.base.BaseType
```

```

    _soap_tag = 'aggregate'
class taniumpy.object_types.system_status_list.SystemStatusList
    Bases: taniumpy.object_types.base.BaseType
    _soap_tag = 'system_status'
class taniumpy.object_types.upload_file.UploadFile
    Bases: taniumpy.object_types.base.BaseType
    _soap_tag = 'upload_file'
class taniumpy.object_types.upload_file_list.UploadFileList
    Bases: taniumpy.object_types.base.BaseType
    _soap_tag = 'file_parts'
class taniumpy.object_types.upload_file_status.UploadFileStatus
    Bases: taniumpy.object_types.base.BaseType
    _soap_tag = 'upload_file_status'
class taniumpy.object_types.user.User
    Bases: taniumpy.object_types.base.BaseType
    _soap_tag = 'user'
class taniumpy.object_types.user_list.UserList
    Bases: taniumpy.object_types.base.BaseType
    _soap_tag = 'users'
class taniumpy.object_types.user_role.UserRole
    Bases: taniumpy.object_types.base.BaseType
    _soap_tag = 'role'
class taniumpy.object_types.user_role_list.UserRoleList
    Bases: taniumpy.object_types.base.BaseType
    _soap_tag = 'roles'
class taniumpy.object_types.version_aggregate.VersionAggregate
    Bases: taniumpy.object_types.base.BaseType
    _soap_tag = 'version'
class taniumpy.object_types.version_aggregate_list.VersionAggregateList
    Bases: taniumpy.object_types.base.BaseType
    _soap_tag = 'versions'
class taniumpy.object_types.white_listed_url.WhiteListedUrl
    Bases: taniumpy.object_types.base.BaseType
    _soap_tag = 'white_listed_url'
class taniumpy.object_types.white_listed_url_list.WhiteListedUrlList
    Bases: taniumpy.object_types.base.BaseType
    _soap_tag = 'white_listed_urls'
class taniumpy.object_types.xml_error.XmlError
    Bases: taniumpy.object_types.base.BaseType
    _soap_tag = 'error'

```

## 1.4 xmldict module

Makes working with XML feel like you are working with JSON

```
xmldict.parse(xml_input, encoding=None, expat=module 'xml.parsers.expat' from
               'Library/Python/2.7/site-packages/_xmlplus/parsers/expat.pyc',
               process_namespaces=False, namespace_separator=':', **kwargs)
```

Parse the given XML input and convert it into a dictionary.

*xml\_input* can either be a *string* or a file-like object.

If *xml\_attribs* is *True*, element attributes are put in the dictionary among regular child elements, using *@* as a prefix to avoid collisions. If set to *False*, they are just ignored.

Simple example:

```
>>> import xmldict
>>> doc = xmldict.parse("""
... <a prop="x">
...   <b>1</b>
...   <b>2</b>
... </a>
... """)
>>> doc['a']['@prop']
u'x'
>>> doc['a']['b']
[u'1', u'2']
```

If *item\_depth* is 0, the function returns a dictionary for the root element (default behavior). Otherwise, it calls *item\_callback* every time an item at the specified depth is found and returns *None* in the end (streaming mode).

The callback function receives two parameters: the *path* from the document root to the item (name-attrs pairs), and the *item* (dict). If the callback's return value is false-ish, parsing will be stopped with the *ParsingInterrupted* exception.

Streaming example:

```
>>> def handle(path, item):
...     print 'path:%s item:%s' % (path, item)
...     return True
...
>>> xmldict.parse("""
... <a prop="x">
...   <b>1</b>
...   <b>2</b>
... </a>""", item_depth=2, item_callback=handle)
path:[(u'a', {u'prop': u'x'})], (u'b', None)] item:1
path:[(u'a', {u'prop': u'x'})], (u'b', None)] item:2
```

The optional argument *postprocessor* is a function that takes *path*, *key* and *value* as positional arguments and returns a new (*key*, *value*) pair where both *key* and *value* may have changed. Usage example:

```
>>> def postprocessor(path, key, value):
...     try:
...         return key + ':int', int(value)
...     except (ValueError, TypeError):
...         return key, value
>>> xmldict.parse('<a><b>1</b><b>2</b><b>x</b></a>',
...               postprocessor=postprocessor)
OrderedDict([(u'a', OrderedDict([(u'b:int', [1, 2]), (u'b', u'x')]))])
```



You can pass an alternate version of *expat* (such as *defusedexpat*) by using the *expat* parameter. E.g:

```
>>> import defusedexpat
>>> xmldict.parse('<a>hello</a>', expat=defusedexpat.pyexpat)
OrderedDict([(u'a', u'hello')])
```

`xmldict.unparse(input_dict, output=None, encoding='utf-8', full_document=True, **kwargs)`

Emit an XML document for the given *input\_dict* (reverse of *parse*).

The resulting XML document is returned as a string, but if *output* (a file-like object) is specified, it is written there instead.

Dictionary keys prefixed with *attr\_prefix* (default='@') are interpreted as XML node attributes, whereas keys equal to *cdata\_key* (default='#text') are treated as character data.

The *pretty* parameter (default='False') enables pretty-printing. In this mode, lines are terminated with 'n' and indented with 't', but this can be customized with the *newl* and *indent* parameters.

## 1.5 ddt module

`ddt.data(*values)`

Method decorator to add to your test methods.

Should be added to methods of instances of `unittest.TestCase`.

`ddt.ddt(cls)`

Class decorator for subclasses of `unittest.TestCase`.

Apply this decorator to the test case class, and then decorate test methods with `@data`.

For each method decorated with `@data`, this will effectively create as many methods as data items are passed as parameters to `@data`.

The names of the test methods follow the pattern `original_test_name_{ordinal}_{data}`. `ordinal` is the position of the data argument, starting with 1.

For data we use a string representation of the data value converted into a valid python identifier. If `data.__name__` exists, we use that instead.

For each method decorated with `@file_data('test_data.json')`, the decorator will try to load the `test_data.json` file located relative to the python file containing the method that is decorated. It will, for each `test_name` key create as many methods in the list of values from the `data` key.

`ddt.file_data(value)`

Method decorator to add to your test methods.

Should be added to methods of instances of `unittest.TestCase`.

`value` should be a path relative to the directory of the file containing the decorated `unittest.TestCase`. The file should contain JSON encoded data, that can either be a list or a dict.

In case of a list, each value in the list will correspond to one test case, and the value will be concatenated to the test method name.

In case of a dict, keys will be used as suffixes to the name of the test case, and values will be fed as test data.

`ddt.is_hash_randomized()`

`ddt.mk_test_name(name, value, index=0)`

Generate a new name for a test case.

It will take the original test name and append an ordinal index and a string representation of the value, and convert the result into a valid python identifier by replacing extraneous characters with `_`.

If hash randomization is enabled (a feature available since 2.7.3/3.2.3 and enabled by default since 3.3) and a “non-trivial” value is passed this will omit the name argument by default. Set `PYTHONHASHSEED` to a fixed value before running tests in these cases to get the names back consistently or use the `__name__` attribute on data values.

A “trivial” value is a plain scalar, or a tuple or list consisting only of trivial values.

`ddt.unpack(func)`  
Method decorator to add unpack feature.

## 1.6 threaded\_http module

Simple HTTP server for testing purposes

```
class threaded_http.CustomHTTPHandler(request, client_address, server)
    Bases: BaseHTTPServer.BaseHTTPRequestHandler

    ENABLE_LOGGING = True

    do_GET()

    do_POST()

    log_message(format, *args)

class threaded_http.ThreadedHTTPServer(server_address, RequestHandlerClass,
                                         bind_and_activate=True)
    Bases: SocketServer.ThreadingMixIn, BaseHTTPServer.HTTPServer

    Handle requests in a separate thread.

threaded_http.threaded_http(host='localhost', port=4443, verbosity=2)
    establishes an HTTP server on host:port in a thread
```

## 1.7 requests package

### 1.7.1 requests HTTP library

Requests is an HTTP library, written in Python, for human beings. Basic GET usage:

```
>>> import requests
>>> r = requests.get('https://www.python.org')
>>> r.status_code
200
>>> 'Python is a programming language' in r.content
True
```

... or POST:

```
>>> payload = dict(key1='value1', key2='value2')
>>> r = requests.post('http://httpbin.org/post', data=payload)
>>> print(r.text)
{
    ...
    "form": {
```

```
"key2": "value2",  
  "key1": "value1"  
},  
...  
}
```

The other HTTP methods are supported - see *requests.api*. Full documentation is at <<http://python-requests.org>>.

**copyright**

3. 2015 by Kenneth Reitz.

**license** Apache 2.0, see LICENSE for more details.



## INDICES AND TABLES

- `genindex`
- `modindex`
- `search`



**d**

ddt, 589

**p**

pytan, 3

pytan.binsupport, 72

pytan.constants, 57

pytan.exceptions, 33

pytan.handler, 3

pytan.pollers, 52

pytan.sessions, 35

pytan.utils, 59

pytan.xml\_clean, 81

**r**

requests, 590

**t**

taniumpy, 579

taniumpy.object\_types, 579

taniumpy.object\_types.action, 579

taniumpy.object\_types.action\_list, 579

taniumpy.object\_types.action\_list\_info, 579

taniumpy.object\_types.action\_stop, 579

taniumpy.object\_types.action\_stop\_list, 579

taniumpy.object\_types.all\_objects, 579

taniumpy.object\_types.archived\_question, 579

taniumpy.object\_types.archived\_question\_list, 579

taniumpy.object\_types.audit\_data, 580

taniumpy.object\_types.base, 580

taniumpy.object\_types.cache\_filter, 581

taniumpy.object\_types.cache\_filter\_list, 581

taniumpy.object\_types.cache\_info, 581

taniumpy.object\_types.client\_count, 581

taniumpy.object\_types.client\_status, 581

taniumpy.object\_types.column, 581

taniumpy.object\_types.column\_set, 581

taniumpy.object\_types.computer\_group, 581

taniumpy.object\_types.computer\_group\_list, 581

taniumpy.object\_types.computer\_group\_spec, 581

taniumpy.object\_types.computer\_spec\_list, 581

taniumpy.object\_types.error\_list, 582

taniumpy.object\_types.filter, 582

taniumpy.object\_types.filter\_list, 582

taniumpy.object\_types.group, 582

taniumpy.object\_types.group\_list, 582

taniumpy.object\_types.metadata\_item, 582

taniumpy.object\_types.metadata\_list, 582

taniumpy.object\_types.object\_list, 582

taniumpy.object\_types.object\_list\_types, 582

taniumpy.object\_types.options, 582

taniumpy.object\_types.package\_file, 582

taniumpy.object\_types.package\_file\_list, 582

taniumpy.object\_types.package\_file\_status, 582

taniumpy.object\_types.package\_file\_status\_list, 582

taniumpy.object\_types.package\_file\_template, 583

taniumpy.object\_types.package\_file\_template\_list, 583

taniumpy.object\_types.package\_spec, 583

taniumpy.object\_types.package\_spec\_list, 583

taniumpy.object\_types.parameter, 583

taniumpy.object\_types.parameter\_list, 583

taniumpy.object\_types.parse\_job, 583

taniumpy.object\_types.parse\_job\_list, 583

taniumpy.object\_types.parse\_result, 583

taniumpy.object\_types.parse\_result\_group, 583

`taniumpy.object_types.parse_result_group`, 583  
`taniumpy.object_types.parse_result_list`, 583  
`taniumpy.object_types.permission_list`, 583  
`taniumpy.object_types.plugin`, 583  
`taniumpy.object_types.plugin_argument`, 584  
`taniumpy.object_types.plugin_argument_list`, 584  
`taniumpy.object_types.plugin_command_list`, 584  
`taniumpy.object_types.plugin_list`, 584  
`taniumpy.object_types.plugin_schedule`, 584  
`taniumpy.object_types.plugin_schedule_list`, 584  
`taniumpy.object_types.plugin_sql`, 584  
`taniumpy.object_types.plugin_sql_column`, 584  
`taniumpy.object_types.plugin_sql_result`, 584  
`taniumpy.object_types.question`, 584  
`taniumpy.object_types.question_list`, 584  
`taniumpy.object_types.question_list_info`, 584  
`taniumpy.object_types.result_info`, 584  
`taniumpy.object_types.result_set`, 585  
`taniumpy.object_types.row`, 585  
`taniumpy.object_types.saved_action`, 585  
`taniumpy.object_types.saved_action_approval`, 585  
`taniumpy.object_types.saved_action_list`, 585  
`taniumpy.object_types.saved_action_policy`, 585  
`taniumpy.object_types.saved_action_row_id`, 585  
`taniumpy.object_types.saved_question`, 585  
`taniumpy.object_types.saved_question_list`, 585  
`taniumpy.object_types.select`, 586  
`taniumpy.object_types.select_list`, 586  
`taniumpy.object_types.sensor`, 586  
`taniumpy.object_types.sensor_list`, 586  
`taniumpy.object_types.sensor_query`, 586  
`taniumpy.object_types.sensor_query_list`, 586  
`taniumpy.object_types.sensor_subcolumn`, 586  
`taniumpy.object_types.sensor_subcolumn_list`, 586  
`taniumpy.object_types.sensor_types`, 586  
`taniumpy.object_types.soap_error`, 586  
`taniumpy.object_types.string_hint_list`, 586  
`taniumpy.object_types.system_setting`, 586  
`taniumpy.object_types.system_setting_list`, 586  
`taniumpy.object_types.system_status_aggregate`, 586  
`taniumpy.object_types.system_status_list`, 587  
`taniumpy.object_types.upload_file`, 587  
`taniumpy.object_types.upload_file_list`, 587  
`taniumpy.object_types.upload_file_status`, 587  
`taniumpy.object_types.user`, 587  
`taniumpy.object_types.user_list`, 587  
`taniumpy.object_types.user_role`, 587  
`taniumpy.object_types.user_role_list`, 587  
`taniumpy.object_types.version_aggregate`, 587  
`taniumpy.object_types.version_aggregate_list`, 587  
`taniumpy.object_types.white_listed_url`, 587  
`taniumpy.object_types.white_listed_url_list`, 587  
`taniumpy.object_types.xml_error`, 587  
`test_pytan_invalid_server_tests`, 91  
`test_pytan_unit`, 82  
`test_pytan_valid_server_tests`, 87  
`threaded_http`, 590

X

omitted, 588



## Symbols

- `__author__` (in module `pytan`), 3
- `__copyright__` (in module `pytan`), 3
- `__license__` (in module `pytan`), 3
- `__version__` (in module `pytan`), 3
- `_add()` (`pytan.handler.Handler` method), 6
- `_ask_manual()` (`pytan.handler.Handler` method), 7
- `_build_body()` (`pytan.sessions.Session` method), 36
- `_check_auth()` (`pytan.sessions.Session` method), 37
- `_check_sse_crash_prevention()` (`pytan.handler.Handler` method), 9
- `_check_sse_empty_rs()` (`pytan.handler.Handler` method), 9
- `_check_sse_format_support()` (`pytan.handler.Handler` method), 9
- `_check_sse_timing()` (`pytan.handler.Handler` method), 9
- `_check_sse_version()` (`pytan.handler.Handler` method), 9
- `_clean_headers()` (`pytan.sessions.Session` method), 37
- `_create_add_object_body()` (`pytan.sessions.Session` method), 37
- `_create_delete_object_body()` (`pytan.sessions.Session` method), 37
- `_create_get_object_body()` (`pytan.sessions.Session` method), 37
- `_create_get_result_data_body()` (`pytan.sessions.Session` method), 38
- `_create_get_result_info_body()` (`pytan.sessions.Session` method), 38
- `_create_run_plugin_object_body()` (`pytan.sessions.Session` method), 38
- `_create_update_object_body()` (`pytan.sessions.Session` method), 38
- `_deploy_action()` (`pytan.handler.Handler` method), 9
- `_derive_attribute()` (`pytan.pollers.QuestionPoller` method), 55
- `_derive_expiration()` (`pytan.pollers.QuestionPoller` method), 55
- `_derive_object_info()` (`pytan.pollers.ActionPoller` method), 52
- `_derive_object_info()` (`pytan.pollers.QuestionPoller` method), 55
- `_derive_package_spec()` (`pytan.pollers.ActionPoller` method), 52
- `_derive_result_map()` (`pytan.pollers.ActionPoller` method), 52
- `_derive_status()` (`pytan.pollers.ActionPoller` method), 53
- `_derive_stopped_flag()` (`pytan.pollers.ActionPoller` method), 53
- `_derive_target_group()` (`pytan.pollers.ActionPoller` method), 53
- `_derive_verify_enabled()` (`pytan.pollers.ActionPoller` method), 53
- `_export_class_BaseType()` (`pytan.handler.Handler` method), 11
- `_export_class_ResultSet()` (`pytan.handler.Handler` method), 12
- `_export_format_csv()` (`pytan.handler.Handler` method), 12
- `_export_format_json()` (`pytan.handler.Handler` method), 12
- `_export_format_xml()` (`pytan.handler.Handler` method), 12
- `_extract_resultxml()` (`pytan.sessions.Session` method), 38
- `_find()` (`pytan.handler.Handler` method), 12
- `_find_stat_target()` (`pytan.sessions.Session` method), 39
- `_fix_group()` (`pytan.pollers.ActionPoller` method), 53
- `_flatten_server_info()` (`pytan.sessions.Session` method), 39
- `_from_json()` (`taniumpy.object_types.base.BaseType` class method), 580
- `_full_url()` (`pytan.sessions.Session` method), 39
- `_get_multi()` (`pytan.handler.Handler` method), 12
- `_get_package_def()` (`pytan.handler.Handler` method), 13
- `_get_percentage()` (`pytan.sessions.Session` method), 39
- `_get_response()` (`pytan.sessions.Session` method), 40
- `_get_sensor_defs()` (`pytan.handler.Handler` method), 13
- `_get_single()` (`pytan.handler.Handler` method), 13
- `_http_get()` (`pytan.sessions.Session` method), 40
- `_http_post()` (`pytan.sessions.Session` method), 42
- `_invalid_server_version()` (`pytan.sessions.Session` method), 43
- `_parse_versioning()` (`pytan.sessions.Session` method), 43
- `_post_init()` (`pytan.pollers.ActionPoller` method), 53
- `_post_init()` (`pytan.pollers.QuestionPoller` method), 55

`_post_init()` (pytan.pollers.SSEPoller method), 57

`_refetch_obj()` (pytan.pollers.QuestionPoller method), 55

`_regex_body_for_element()` (pytan.sessions.Session method), 43

`_replace_auth()` (pytan.sessions.Session method), 44

`_resolve_sse_format()` (pytan.handler.Handler method), 13

`_resolve_stat_target()` (pytan.sessions.Session method), 44

`_single_find()` (pytan.handler.Handler method), 13

`_soap_tag` (taniumpy.object\_types.action.Action attribute), 579

`_soap_tag` (taniumpy.object\_types.action\_list.ActionList attribute), 579

`_soap_tag` (taniumpy.object\_types.action\_list\_info.ActionListInfo attribute), 579

`_soap_tag` (taniumpy.object\_types.action\_stop.ActionStop attribute), 579

`_soap_tag` (taniumpy.object\_types.action\_stop\_list.ActionStopList attribute), 579

`_soap_tag` (taniumpy.object\_types.archived\_question.ArchivedQuestion attribute), 579

`_soap_tag` (taniumpy.object\_types.archived\_question\_list.ArchivedQuestionList attribute), 579

`_soap_tag` (taniumpy.object\_types.audit\_data.AuditData attribute), 580

`_soap_tag` (taniumpy.object\_types.base.BaseType attribute), 580

`_soap_tag` (taniumpy.object\_types.cache\_filter.CacheFilter attribute), 581

`_soap_tag` (taniumpy.object\_types.cache\_filter\_list.CacheFilterList attribute), 581

`_soap_tag` (taniumpy.object\_types.cache\_info.CacheInfo attribute), 581

`_soap_tag` (taniumpy.object\_types.client\_count.ClientCount attribute), 581

`_soap_tag` (taniumpy.object\_types.client\_status.ClientStatus attribute), 581

`_soap_tag` (taniumpy.object\_types.computer\_group.ComputerGroup attribute), 581

`_soap_tag` (taniumpy.object\_types.computer\_group\_list.ComputerGroupList attribute), 581

`_soap_tag` (taniumpy.object\_types.computer\_group\_spec.ComputerGroupSpec attribute), 581

`_soap_tag` (taniumpy.object\_types.computer\_group\_spec\_list.ComputerGroupSpecList attribute), 582

`_soap_tag` (taniumpy.object\_types.error\_list.ErrorList attribute), 582

`_soap_tag` (taniumpy.object\_types.filter.Filter attribute), 582

`_soap_tag` (taniumpy.object\_types.filter\_list.FilterList attribute), 582

`_soap_tag` (taniumpy.object\_types.group.Group attribute), 582

`_soap_tag` (taniumpy.object\_types.group\_list.GroupList attribute), 582

`_soap_tag` (taniumpy.object\_types.metadata\_item.MetadataItem attribute), 582

`_soap_tag` (taniumpy.object\_types.metadata\_list.MetadataList attribute), 582

`_soap_tag` (taniumpy.object\_types.object\_list.ObjectList attribute), 582

`_soap_tag` (taniumpy.object\_types.options.Options attribute), 582

`_soap_tag` (taniumpy.object\_types.package\_file.PackageFile attribute), 582

`_soap_tag` (taniumpy.object\_types.package\_file\_list.PackageFileList attribute), 582

`_soap_tag` (taniumpy.object\_types.package\_file\_status.PackageFileStatus attribute), 582

`_soap_tag` (taniumpy.object\_types.package\_file\_status\_list.PackageFileStatusList attribute), 582

`_soap_tag` (taniumpy.object\_types.package\_file\_template.PackageFileTemplate attribute), 583

`_soap_tag` (taniumpy.object\_types.package\_file\_template\_list.PackageFileTemplateList attribute), 583

`_soap_tag` (taniumpy.object\_types.package\_spec.PackageSpec attribute), 583

`_soap_tag` (taniumpy.object\_types.package\_spec\_list.PackageSpecList attribute), 583

`_soap_tag` (taniumpy.object\_types.parameter.Parameter attribute), 583

`_soap_tag` (taniumpy.object\_types.parameter\_list.ParameterList attribute), 583

`_soap_tag` (taniumpy.object\_types.parse\_job.ParseJob attribute), 583

`_soap_tag` (taniumpy.object\_types.parse\_job\_list.ParseJobList attribute), 583

`_soap_tag` (taniumpy.object\_types.parse\_result.ParseResult attribute), 583

`_soap_tag` (taniumpy.object\_types.parse\_result\_group.ParseResultGroup attribute), 583

`_soap_tag` (taniumpy.object\_types.parse\_result\_group\_list.ParseResultGroupList attribute), 583

`_soap_tag` (taniumpy.object\_types.parse\_result\_list.ParseResultList attribute), 583

`_soap_tag` (taniumpy.object\_types.permission\_list.PermissionList attribute), 583

`_soap_tag` (taniumpy.object\_types.plugin.Plugin attribute), 584

`_soap_tag` (taniumpy.object\_types.plugin\_argument.PluginArgument attribute), 584

`_soap_tag` (taniumpy.object\_types.plugin\_argument\_list.PluginArgumentList attribute), 584

`_soap_tag` (taniumpy.object\_types.plugin\_command\_list.PluginCommandList attribute), 584

`_soap_tag` (taniumpy.object\_types.plugin\_list.PluginList attribute), 584

\_soap\_tag (taniumpy.object\_types.plugin\_schedule.PluginSchedule attribute), 584  
 \_soap\_tag (taniumpy.object\_types.plugin\_schedule\_list.PluginScheduleList attribute), 584  
 \_soap\_tag (taniumpy.object\_types.plugin\_sql.PluginSql attribute), 584  
 \_soap\_tag (taniumpy.object\_types.plugin\_sql\_column.PluginSqlColumn attribute), 584  
 \_soap\_tag (taniumpy.object\_types.plugin\_sql\_result.PluginSqlResult attribute), 584  
 \_soap\_tag (taniumpy.object\_types.question.Question attribute), 584  
 \_soap\_tag (taniumpy.object\_types.question\_list.QuestionList attribute), 584  
 \_soap\_tag (taniumpy.object\_types.question\_list\_info.QuestionListInfo attribute), 584  
 \_soap\_tag (taniumpy.object\_types.saved\_action.SavedAction attribute), 585  
 \_soap\_tag (taniumpy.object\_types.saved\_action\_approval.SavedActionApproval attribute), 585  
 \_soap\_tag (taniumpy.object\_types.saved\_action\_list.SavedActionList attribute), 585  
 \_soap\_tag (taniumpy.object\_types.saved\_action\_policy.SavedActionPolicy attribute), 585  
 \_soap\_tag (taniumpy.object\_types.saved\_action\_row\_id\_list.SavedActionRowIdList attribute), 585  
 \_soap\_tag (taniumpy.object\_types.saved\_question.SavedQuestion attribute), 585  
 \_soap\_tag (taniumpy.object\_types.saved\_question\_list.SavedQuestionList attribute), 586  
 \_soap\_tag (taniumpy.object\_types.select.Select attribute), 586  
 \_soap\_tag (taniumpy.object\_types.select\_list.SelectList attribute), 586  
 \_soap\_tag (taniumpy.object\_types.sensor.Sensor attribute), 586  
 \_soap\_tag (taniumpy.object\_types.sensor\_list.SensorList attribute), 586  
 \_soap\_tag (taniumpy.object\_types.sensor\_query.SensorQuery attribute), 586  
 \_soap\_tag (taniumpy.object\_types.sensor\_query\_list.SensorQueryList attribute), 586  
 \_soap\_tag (taniumpy.object\_types.sensor\_subcolumn.SensorSubcolumn attribute), 586  
 \_soap\_tag (taniumpy.object\_types.sensor\_subcolumn\_list.SensorSubcolumnList attribute), 586  
 \_soap\_tag (taniumpy.object\_types.soap\_error.SoapError attribute), 586  
 \_soap\_tag (taniumpy.object\_types.string\_hint\_list.StringHintList attribute), 586  
 \_soap\_tag (taniumpy.object\_types.system\_setting.SystemSetting attribute), 586  
 \_soap\_tag (taniumpy.object\_types.system\_setting\_list.SystemSettingList attribute), 586  
 \_soap\_tag (taniumpy.object\_types.system\_status\_aggregate.SystemStatusAggregate attribute), 586  
 \_soap\_tag (taniumpy.object\_types.system\_status\_list.SystemStatusList attribute), 587  
 \_soap\_tag (taniumpy.object\_types.upload\_file.UploadFile attribute), 587  
 \_soap\_tag (taniumpy.object\_types.upload\_file\_list.UploadFileList attribute), 587  
 \_soap\_tag (taniumpy.object\_types.upload\_file\_status.UploadFileStatus attribute), 587  
 \_soap\_tag (taniumpy.object\_types.user.User attribute), 587  
 \_soap\_tag (taniumpy.object\_types.user\_list.UserList attribute), 587  
 \_soap\_tag (taniumpy.object\_types.user\_role.UserRole attribute), 587  
 \_soap\_tag (taniumpy.object\_types.user\_role\_list.UserRoleList attribute), 587  
 \_soap\_tag (taniumpy.object\_types.version\_aggregate.VersionAggregate attribute), 587  
 \_soap\_tag (taniumpy.object\_types.version\_aggregate\_list.VersionAggregateList attribute), 587  
 \_soap\_tag (taniumpy.object\_types.white\_listed\_url.WhiteListedUrl attribute), 587  
 \_soap\_tag (taniumpy.object\_types.white\_listed\_url\_list.WhiteListedUrlList attribute), 587  
 \_soap\_tag (taniumpy.object\_types.xml\_error.XmlError attribute), 587  
 \_stats\_loop() (pytan.sessions.Session method), 44  
 \_stop (pytan.pollers.QuestionPoller attribute), 55  
 \_version\_support\_check() (pytan.handler.Handler method), 14  
 A  
 Action (class in taniumpy.object\_types.action), 579  
 ACTION\_DONE\_KEY (pytan.pollers.ActionPoller attribute), 52  
 ActionList (class in taniumpy.object\_types.action\_list), 579  
 ActionListInfo (class in taniumpy.object\_types.action\_list\_info), 579  
 ActionPoller (class in pytan.pollers), 52  
 ActionStop (class in taniumpy.object\_types.action\_stop), 579  
 ActionStopList (class in taniumpy.object\_types.action\_stop\_list), 579  
 add() (pytan.sessions.Session method), 44  
 add\_task\_report\_argparser() (in module pytan.binsupport), 73  
 add\_file\_log() (in module pytan.binsupport), 73  
 add\_get\_object\_report\_argparser() (in module pytan.binsupport), 73

- `add_report_file_options()` (in module `pytan.binsupport`), 73
- `ALL_REQUESTS_RESPONSES` (`pytan.sessions.Session` attribute), 35
- `append()` (`taniumpy.object_types.base.BaseType` method), 580
- `apply_options_obj()` (in module `pytan.utils`), 59
- `approve_saved_action()` (`pytan.handler.Handler` method), 14
- `ArchivedQuestion` (class in `taniumpy.object_types.archived_question`), 579
- `ArchivedQuestionList` (class in `taniumpy.object_types.archived_question_list`), 579
- `ask()` (`pytan.handler.Handler` method), 14
- `ask_manual()` (`pytan.handler.Handler` method), 14
- `ask_parsed()` (`pytan.handler.Handler` method), 17
- `ask_saved()` (`pytan.handler.Handler` method), 19
- `AuditData` (class in `taniumpy.object_types.audit_data`), 580
- `AUTH_CONNECT_TIMEOUT_SEC` (`pytan.sessions.Session` attribute), 35
- `AUTH_FAIL_CODES` (`pytan.sessions.Session` attribute), 35
- `AUTH_RES` (`pytan.sessions.Session` attribute), 35
- `AUTH_RESPONSE_TIMEOUT_SEC` (`pytan.sessions.Session` attribute), 35
- `authenticate()` (`pytan.sessions.Session` method), 44
- `AuthorizationError`, 33
- B**
- `BAD_RESPONSE_CMD_PRUNES` (`pytan.sessions.Session` attribute), 35
- `BAD_SERVER_VERSIONS` (`pytan.sessions.Session` attribute), 35
- `BadResponseError`, 33
- `BaseType` (class in `taniumpy.object_types.base`), 580
- `build_group_obj()` (in module `pytan.utils`), 60
- `build_manual_q()` (in module `pytan.utils`), 60
- `build_metadatalist_obj()` (in module `pytan.utils`), 60
- `build_param_obj()` (in module `pytan.utils`), 60
- `build_param_objlist()` (in module `pytan.utils`), 61
- `build_selectlist_obj()` (in module `pytan.utils`), 61
- C**
- `CacheFilter` (class in `taniumpy.object_types.cache_filter`), 581
- `CacheFilterList` (class in `taniumpy.object_types.cache_filter_list`), 581
- `CacheInfo` (class in `taniumpy.object_types.cache_info`), 581
- `calc_percent()` (in module `pytan.utils`), 61
- `change_console_format()` (in module `pytan.utils`), 62
- `check_dictkey()` (in module `pytan.utils`), 62
- `check_for_help()` (in module `pytan.utils`), 62
- `chew_csv()` (in module `test_pytan_valid_server_tests`), 91
- `chk_def_key()` (in module `pytan.utils`), 62
- `clean_kwargs()` (in module `pytan.utils`), 62
- `ClientCount` (class in `taniumpy.object_types.client_count`), 581
- `ClientStatus` (class in `taniumpy.object_types.client_status`), 581
- `Column` (class in `taniumpy.object_types.column`), 581
- `ColumnSet` (class in `taniumpy.object_types.column_set`), 581
- `COMPLETE_PCT_DEFAULT` (`pytan.pollers.ActionPoller` attribute), 52
- `COMPLETE_PCT_DEFAULT` (`pytan.pollers.QuestionPoller` attribute), 54
- `ComputerGroup` (class in `taniumpy.object_types.computer_group`), 581
- `ComputerGroupList` (class in `taniumpy.object_types.computer_group_list`), 581
- `ComputerGroupSpec` (class in `taniumpy.object_types.computer_group_spec`), 581
- `ComputerSpecList` (class in `taniumpy.object_types.computer_spec_list`), 581
- `copy_obj()` (in module `pytan.utils`), 63
- `copy_package_obj_for_action()` (in module `pytan.utils`), 63
- `create_dashboard()` (`pytan.handler.Handler` method), 20
- `create_from_json()` (`pytan.handler.Handler` method), 20
- `create_group()` (`pytan.handler.Handler` method), 21
- `create_package()` (`pytan.handler.Handler` method), 21
- `create_report_file()` (`pytan.handler.Handler` method), 23
- `create_sensor()` (`pytan.handler.Handler` method), 23
- `create_user()` (`pytan.handler.Handler` method), 23
- `create_whitelisted_url()` (`pytan.handler.Handler` method), 24
- `csvdictwriter()` (in module `pytan.binsupport`), 73
- `CustomArgFormat` (class in `pytan.binsupport`), 72
- `CustomArgParse` (class in `pytan.binsupport`), 72
- `CustomHTTPHandler` (class in `threaded_http`), 590
- D**
- `data()` (in module `ddt`), 589
- `datetime_to_timestr()` (in module `pytan.utils`), 63
- `ddt` (module), 589
- `ddt()` (in module `ddt`), 589
- `DEBUG_FORMAT` (in module `pytan.constants`), 57
- `debug_list()` (in module `pytan.binsupport`), 73
- `debug_obj()` (in module `pytan.binsupport`), 73
- `DEFAULT_REPLACEMENT` (in module `pytan.xml_clean`), 81
- `DefinitionParserError`, 33

dehumanize\_package() (in module pytan.utils), 63  
 dehumanize\_question\_filters() (in module pytan.utils), 64  
 dehumanize\_question\_options() (in module pytan.utils), 64  
 dehumanize\_sensors() (in module pytan.utils), 64  
 delete() (pytan.handler.Handler method), 24  
 delete() (pytan.sessions.Session method), 46  
 delete\_dashboard() (pytan.handler.Handler method), 25  
 deploy\_action() (pytan.handler.Handler method), 25  
 derive\_param\_default() (in module pytan.utils), 64  
 disable\_stats\_loop() (pytan.sessions.Session method), 46  
 do\_GET() (threaded\_http.CustomHTTPHandler method), 590  
 do\_POST() (threaded\_http.CustomHTTPHandler method), 590

## E

ELEMENT\_RE\_TXT (pytan.sessions.Session attribute), 35  
 emit() (pytan.utils.SplitStreamHandler method), 59  
 empty\_obj() (in module pytan.utils), 64  
 ENABLE\_LOGGING (threaded\_http.CustomHTTPHandler attribute), 590  
 enable\_stats\_loop() (pytan.sessions.Session method), 46  
 error() (pytan.binsupport.CustomArgParse method), 72  
 ErrorList (class in taniumpy.object\_types.error\_list), 582  
 eval\_timing() (in module pytan.utils), 65  
 EXPIRATION\_ATTR (pytan.pollers.ActionPoller attribute), 52  
 EXPIRATION\_ATTR (pytan.pollers.QuestionPoller attribute), 54  
 EXPIRY\_FALLBACK\_SECS (pytan.pollers.QuestionPoller attribute), 54  
 explode\_json() (taniumpy.object\_types.base.BaseType method), 580  
 export\_id (pytan.pollers.SSEPoller attribute), 57  
 EXPORT\_MAPS (in module pytan.constants), 57  
 export\_obj() (pytan.handler.Handler method), 27  
 export\_to\_report\_file() (pytan.handler.Handler method), 28  
 extract\_filter() (in module pytan.utils), 65  
 extract\_options() (in module pytan.utils), 65  
 extract\_params() (in module pytan.utils), 65  
 extract\_selector() (in module pytan.utils), 65

## F

file\_data() (in module ddt), 589  
 Filter (class in taniumpy.object\_types.filter), 582  
 filter\_filename() (in module pytan.binsupport), 73  
 FILTER\_MAPS (in module pytan.constants), 58  
 FILTER\_RE (in module pytan.constants), 58  
 filter\_sensors() (in module pytan.binsupport), 73  
 filter\_sourced\_sensors() (in module pytan.binsupport), 73  
 FilterList (class in taniumpy.object\_types.filter\_list), 582

find() (pytan.sessions.Session method), 46  
 finished\_eq\_passed\_loop() (pytan.pollers.ActionPoller method), 53  
 flatten\_jsonable() (taniumpy.object\_types.base.BaseType method), 580  
 from\_jsonable() (taniumpy.object\_types.base.BaseType static method), 580  
 fromSOAPBody() (taniumpy.object\_types.base.BaseType class method), 580  
 fromSOAPElement() (taniumpy.object\_types.base.BaseType class method), 580  
 fromSOAPElement() (taniumpy.object\_types.column.Column class method), 581  
 fromSOAPElement() (taniumpy.object\_types.column\_set.ColumnSet class method), 581  
 fromSOAPElement() (taniumpy.object\_types.result\_info.ResultInfo class method), 585  
 fromSOAPElement() (taniumpy.object\_types.result\_set.ResultSet class method), 585  
 fromSOAPElement() (taniumpy.object\_types.row.Row class method), 585  
 func\_timing() (in module pytan.utils), 65

## G

get() (pytan.handler.Handler method), 30  
 get\_all() (pytan.handler.Handler method), 30  
 get\_all\_headers() (in module pytan.binsupport), 73  
 get\_all\_loggers() (in module pytan.utils), 65  
 get\_dashboards() (pytan.handler.Handler method), 30  
 get\_dict\_list\_len() (in module pytan.utils), 66  
 get\_filter\_obj() (in module pytan.utils), 66  
 get\_grp\_opts() (in module pytan.binsupport), 73  
 get\_kwargs\_int() (in module pytan.utils), 66  
 get\_now() (in module pytan.utils), 66  
 GET\_OBJ\_MAP (in module pytan.constants), 58  
 get\_obj\_map() (in module pytan.utils), 66  
 get\_obj\_params() (in module pytan.utils), 67  
 get\_percentage() (in module pytan.utils), 67  
 get\_q\_obj\_map() (in module pytan.utils), 67  
 get\_result\_data() (pytan.handler.Handler method), 31  
 get\_result\_data() (pytan.pollers.QuestionPoller method), 55  
 get\_result\_data() (pytan.sessions.Session method), 47  
 get\_result\_data\_sse() (pytan.handler.Handler method), 31  
 get\_result\_data\_sse() (pytan.sessions.Session method), 47  
 get\_result\_info() (pytan.handler.Handler method), 32



`get_result_info()` (pytan.pollers.QuestionPoller method), 55

`get_result_info()` (pytan.sessions.Session method), 47

`get_server_info()` (pytan.sessions.Session method), 47

`get_server_stats()` (pytan.sessions.Session method), 48

`get_server_version()` (pytan.handler.Handler method), 32

`get_server_version()` (pytan.sessions.Session method), 48

`get_sse_data()` (pytan.pollers.SSEPoller method), 57

`get_sse_status()` (pytan.pollers.SSEPoller method), 57

`get_taniumpy_obj()` (in module pytan.utils), 67

Group (class in `taniumpy.object_types.group`), 582

GroupList (class in `taniumpy.object_types.group_list`), 582

## H

Handler (class in `pytan.handler`), 3

`handler` (pytan.pollers.QuestionPoller attribute), 55

HandlerError, 33

HistoryConsole (class in `pytan.binsupport`), 72

`host` (pytan.sessions.Session attribute), 48

HTTP\_AUTH\_RETRY (pytan.sessions.Session attribute), 35

HTTP\_DEBUG (pytan.sessions.Session attribute), 35

`http_get()` (pytan.sessions.Session method), 48

`http_post()` (pytan.sessions.Session method), 49

HTTP\_RETRY\_COUNT (pytan.sessions.Session attribute), 35

HttpError, 33

`human_time()` (in module `pytan.utils`), 67

HumanParserError, 34

## I

IncorrectTypeException, 581

INFO\_CONNECT\_TIMEOUT\_SEC (pytan.sessions.Session attribute), 35

INFO\_FORMAT (in module `pytan.constants`), 58

INFO\_RES (pytan.sessions.Session attribute), 35

INFO\_RESPONSE\_TIMEOUT\_SEC (pytan.sessions.Session attribute), 35

`init_history()` (pytan.binsupport.HistoryConsole method), 72

`input_prompts()` (in module `pytan.binsupport`), 73

`introspect()` (in module `pytan.binsupport`), 73

INVALID\_UNICODE\_RAW\_RE (in module `pytan.xml_clean`), 81

INVALID\_UNICODE\_RE (in module `pytan.xml_clean`), 81

InvalidServerTests (class in `test_pytan_invalid_server_tests`), 91

`is_auth` (pytan.sessions.Session attribute), 50

`is_dict()` (in module `pytan.utils`), 67

`is_hash_randomized()` (in module `ddt`), 589

`is_list()` (in module `pytan.utils`), 67

`is_num()` (in module `pytan.utils`), 67

`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`), 58

`log_message()` (threaded\_http.CustomHTTPHandler method), 590

`log_session_communication()` (in module `pytan.utils`), 68

`logout()` (pytan.sessions.Session method), 50

## M

`map_filter()` (in module `pytan.utils`), 68

`map_option()` (in module `pytan.utils`), 68

`map_options()` (in module `pytan.utils`), 69

MetadataItem (class in `taniumpy.object_types.metadata_item`), 582

MetadataList (class in `taniumpy.object_types.metadata_list`), 582

`mk_test_name()` (in module `ddt`), 589

## N

NotFoundError, 34

## O

`obj` (pytan.pollers.QuestionPoller attribute), 55

OBJECT\_TYPE (pytan.pollers.ActionPoller attribute), 52

OBJECT\_TYPE (pytan.pollers.QuestionPoller attribute), 54

ObjectList (class in `taniumpy.object_types.object_list`), 582

OPTION\_MAPS (in module `pytan.constants`), 58

OPTION\_RE (in module `pytan.constants`), 59

Options (class in `taniumpy.object_types.options`), 582

OVERRIDE\_TIMEOUT\_SECS\_DEFAULT (pytan.pollers.QuestionPoller attribute), 55

## P

PackageFile (class in `taniumpy.object_types.package_file`), 582

PackageFileList (class in `taniumpy.object_types.package_file_list`), 582

PackageFileStatus (class in `taniumpy.object_types.package_file_status`), 582

- PackageFileStatusList (class in taniumpy.object\_types.package\_file\_status\_list), 582
- PackageFileTemplate (class in taniumpy.object\_types.package\_file\_template), 583
- PackageFileTemplateList (class in taniumpy.object\_types.package\_file\_template\_list), 583
- PackageSpec (class in taniumpy.object\_types.package\_spec), 583
- PackageSpecList (class in taniumpy.object\_types.package\_spec\_list), 583
- PARAM\_DELIM (in module pytan.constants), 59
- PARAM\_KEY\_SPLIT (in module pytan.constants), 59
- PARAM\_RE (in module pytan.constants), 59
- PARAM\_SPLIT\_RE (in module pytan.constants), 59
- Parameter (class in taniumpy.object\_types.parameter), 583
- ParameterList (class in taniumpy.object\_types.parameter\_list), 583
- parse() (in module xmltodict), 588
- parse\_defs() (in module pytan.utils), 69
- parse\_query() (pytan.handler.Handler method), 32
- parse\_sensor\_platforms() (in module pytan.binsupport), 73
- ParseJob (class in taniumpy.object\_types.parse\_job), 583
- ParseJobList (class in taniumpy.object\_types.parse\_job\_list), 583
- ParseResult (class in taniumpy.object\_types.parse\_result), 583
- ParseResultGroup (class in taniumpy.object\_types.parse\_result\_group), 583
- ParseResultGroupList (class in taniumpy.object\_types.parse\_result\_group\_list), 583
- ParseResultList (class in taniumpy.object\_types.parse\_result\_list), 583
- passed\_eq\_est\_total\_loop() (pytan.pollers.QuestionPoller method), 55
- PermissionList (class in taniumpy.object\_types.permission\_list), 583
- PickerError, 34
- platform\_is\_6\_2() (pytan.sessions.Session method), 51
- Plugin (class in taniumpy.object\_types.plugin), 583
- plugin\_zip() (in module pytan.utils), 69
- PluginArgument (class in taniumpy.object\_types.plugin\_argument), 584
- PluginArgumentList (class in taniumpy.object\_types.plugin\_argument\_list), 584
- PluginCommandList (class in taniumpy.object\_types.plugin\_command\_list), 584
- PluginList (class in taniumpy.object\_types.plugin\_list), 584
- PluginSchedule (class in taniumpy.object\_types.plugin\_schedule), 584
- PluginScheduleList (class in taniumpy.object\_types.plugin\_schedule\_list), 584
- PluginSql (class in taniumpy.object\_types.plugin\_sql), 584
- PluginSqlColumn (class in taniumpy.object\_types.plugin\_sql\_column), 584
- PluginSqlResult (class in taniumpy.object\_types.plugin\_sql\_result), 584
- POLLING\_SECS\_DEFAULT (pytan.pollers.QuestionPoller attribute), 55
- POLLING\_SECS\_DEFAULT (pytan.pollers.SSEPoller attribute), 57
- PollingError, 34
- port (pytan.sessions.Session attribute), 51
- port\_check() (in module pytan.utils), 69
- print\_help() (pytan.binsupport.CustomArgParse method), 72
- print\_log\_levels() (in module pytan.utils), 70
- print\_obj() (in module pytan.binsupport), 73
- process\_ask\_manual\_args() (in module pytan.binsupport), 74
- process\_ask\_parsed\_args() (in module pytan.binsupport), 74
- process\_ask\_saved\_args() (in module pytan.binsupport), 74
- process\_create\_group\_args() (in module pytan.binsupport), 74
- process\_create\_json\_object\_args() (in module pytan.binsupport), 75
- process\_create\_package\_args() (in module pytan.binsupport), 75
- process\_create\_sensor\_args() (in module pytan.binsupport), 75
- process\_create\_user\_args() (in module pytan.binsupport), 75
- process\_create\_whitelisted\_url\_args() (in module pytan.binsupport), 76
- process\_delete\_object\_args() (in module pytan.binsupport), 76
- process\_deploy\_action\_args() (in module pytan.binsupport), 76
- process\_get\_object\_args() (in module pytan.binsupport), 77
- process\_get\_results\_args() (in module pytan.binsupport), 77
- process\_handler\_args() (in module pytan.binsupport), 77
- process\_print\_sensors\_args() (in module py-

tan.binsupport), 77  
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), 78  
pytan (module), 3  
pytan.binsupport (module), 72  
pytan.constants (module), 57  
pytan.exceptions (module), 33  
pytan.handler (module), 3  
pytan.pollers (module), 52  
pytan.sessions (module), 35  
pytan.utils (module), 59  
pytan.xml\_clean (module), 81  
PytanHelp, 34

## Q

Q\_OBJ\_MAP (in module pytan.constants), 59  
Question (class in taniumpy.object\_types.question), 584  
QuestionList (class in taniumpy.object\_types.question\_list), 584  
QuestionListInfo (class in taniumpy.object\_types.question\_list\_info), 584  
QuestionPoller (class in pytan.pollers), 54

## R

RECORD\_ALL\_REQUESTS (pytan.sessions.Session attribute), 36  
remove\_file\_log() (in module pytan.binsupport), 79  
remove\_logging\_handler() (in module pytan.utils), 70  
replace\_invalid\_unicode() (in module pytan.xml\_clean), 81  
replace\_restricted\_unicode() (in module pytan.xml\_clean), 81  
REQ\_KWARGS (in module pytan.constants), 59  
REQUEST\_BODY\_BASE (pytan.sessions.Session attribute), 36  
requests (module), 590  
REQUESTS\_SESSION (pytan.sessions.Session attribute), 36  
RESTRICTED\_UNICODE\_RAW\_RE (in module pytan.xml\_clean), 81  
RESTRICTED\_UNICODE\_RE (in module pytan.xml\_clean), 81  
result\_info (pytan.pollers.QuestionPoller attribute), 56  
ResultInfo (class in taniumpy.object\_types.result\_info), 584  
ResultSet (class in taniumpy.object\_types.result\_set), 585  
Row (class in taniumpy.object\_types.row), 585  
run() (pytan.pollers.ActionPoller method), 53  
run() (pytan.pollers.QuestionPoller method), 56

run() (pytan.pollers.SSEPoller method), 57  
run\_callback() (pytan.pollers.QuestionPoller method), 56  
run\_plugin() (pytan.handler.Handler method), 33  
run\_plugin() (pytan.sessions.Session method), 51  
RunFalse, 34  
RUNNING\_STATUSES (pytan.pollers.ActionPoller attribute), 52

## S

save() (pytan.sessions.Session method), 51  
save\_history() (pytan.binsupport.HistoryConsole static method), 73  
SavedAction (class in taniumpy.object\_types.saved\_action), 585  
SavedActionApproval (class in taniumpy.object\_types.saved\_action\_approval), 585  
SavedActionList (class in taniumpy.object\_types.saved\_action\_list), 585  
SavedActionPolicy (class in taniumpy.object\_types.saved\_action\_policy), 585  
SavedActionRowIdList (class in taniumpy.object\_types.saved\_action\_row\_id\_list), 585  
SavedQuestion (class in taniumpy.object\_types.saved\_question), 585  
SavedQuestionList (class in taniumpy.object\_types.saved\_question\_list), 585  
seconds\_from\_now() (in module pytan.utils), 70  
seen\_eq\_passed\_loop() (pytan.pollers.ActionPoller method), 54  
Select (class in taniumpy.object\_types.select), 586  
SelectList (class in taniumpy.object\_types.select\_list), 586  
SELECTORS (in module pytan.constants), 59  
Sensor (class in taniumpy.object\_types.sensor), 586  
SENSOR\_TYPE\_MAP (in module pytan.constants), 59  
SensorList (class in taniumpy.object\_types.sensor\_list), 586  
SensorQuery (class in taniumpy.object\_types.sensor\_query), 586  
SensorQueryList (class in taniumpy.object\_types.sensor\_query\_list), 586  
SensorSubcolumn (class in taniumpy.object\_types.sensor\_subcolumn), 586  
SensorSubcolumnList (class in taniumpy.object\_types.sensor\_subcolumn\_list), 586  
server\_version (pytan.sessions.Session attribute), 51  
server\_version\_dict (pytan.sessions.Session attribute), 51  
ServerParseError, 34  
ServerSideExportError, 34



- Session (class in pytan.sessions), 35  
 session\_id (pytan.sessions.Session attribute), 51  
 set\_all\_loglevels() (in module pytan.utils), 70  
 set\_complect\_pct() (pytan.pollers.QuestionPoller method), 56  
 set\_log\_levels() (in module pytan.utils), 70  
 setup\_ask\_manual\_argparser() (in module pytan.binsupport), 79  
 setup\_ask\_parsed\_argparser() (in module pytan.binsupport), 79  
 setup\_ask\_saved\_argparser() (in module pytan.binsupport), 79  
 setup\_console\_logging() (in module pytan.utils), 70  
 setup\_create\_group\_argparser() (in module pytan.binsupport), 79  
 setup\_create\_json\_object\_argparser() (in module pytan.binsupport), 79  
 setup\_create\_package\_argparser() (in module pytan.binsupport), 79  
 setup\_create\_sensor\_argparser() (in module pytan.binsupport), 79  
 setup\_create\_user\_argparser() (in module pytan.binsupport), 79  
 setup\_create\_whitelisted\_url\_argparser() (in module pytan.binsupport), 79  
 setup\_delete\_object\_argparser() (in module pytan.binsupport), 79  
 setup\_deploy\_action\_argparser() (in module pytan.binsupport), 79  
 setup\_get\_object\_argparser() (in module pytan.binsupport), 80  
 setup\_get\_results\_argparser() (in module pytan.binsupport), 80  
 setup\_logging() (pytan.pollers.QuestionPoller method), 56  
 setup\_logging() (pytan.sessions.Session method), 51  
 setup\_parent\_parser() (in module pytan.binsupport), 80  
 setup\_parser() (in module pytan.binsupport), 80  
 setup\_print\_sensors\_argparser() (in module pytan.binsupport), 80  
 setup\_print\_server\_info\_argparser() (in module pytan.binsupport), 80  
 setup\_pytan\_shell\_argparser() (in module pytan.binsupport), 80  
 setup\_stop\_action\_argparser() (in module pytan.binsupport), 80  
 setup\_test() (test\_pytan\_valid\_server\_tests.ValidServerTests method), 87  
 setup\_tsat\_argparser() (in module pytan.binsupport), 80  
 setUpClass() (test\_pytan\_invalid\_server\_tests.InvalidServerTests class method), 91  
 setUpClass() (test\_pytan\_unit.TestManualBuildObjectUtils class method), 85  
 setUpClass() (test\_pytan\_valid\_server\_tests.ValidServerTests class method), 87  
 shrink\_obj() (in module pytan.utils), 70  
 SOAP\_CONNECT\_TIMEOUT\_SEC (pytan.sessions.Session attribute), 36  
 SOAP\_REQUEST\_HEADERS (pytan.sessions.Session attribute), 36  
 SOAP\_RES (pytan.sessions.Session attribute), 36  
 SOAP\_RESPONSE\_TIMEOUT\_SEC (pytan.sessions.Session attribute), 36  
 SoapError (class in taniumpy.object\_types.soap\_error), 586  
 spew() (in module pytan.utils), 70  
 spew() (in module test\_pytan\_invalid\_server\_tests), 91  
 spew() (in module test\_pytan\_valid\_server\_tests), 91  
 SplitStreamHandler (class in pytan.utils), 59  
 SSE\_CRASH\_MAP (in module pytan.constants), 59  
 SSE\_FORMAT\_MAP (in module pytan.constants), 59  
 SSE\_RESTRICT\_MAP (in module pytan.constants), 59  
 sse\_status\_has\_completed\_loop() (pytan.pollers.SSEPoller method), 57  
 SSEPoller (class in pytan.pollers), 56  
 STATS\_LOOP\_ENABLED (pytan.sessions.Session attribute), 36  
 STATS\_LOOP\_SLEEP\_SEC (pytan.sessions.Session attribute), 36  
 STATS\_LOOP\_TARGETS (pytan.sessions.Session attribute), 36  
 stop() (pytan.pollers.QuestionPoller method), 56  
 stop\_action() (pytan.handler.Handler method), 33  
 STR\_ATTRS (pytan.pollers.QuestionPoller attribute), 55  
 STR\_ATTRS (pytan.pollers.SSEPoller attribute), 57  
 StringHintList (class in taniumpy.object\_types.string\_hint\_list), 586  
 SystemSetting (class in taniumpy.object\_types.system\_setting), 586  
 SystemSettingList (class in taniumpy.object\_types.system\_setting\_list), 586  
 SystemStatusAggregate (class in taniumpy.object\_types.system\_status\_aggregate), 586  
 SystemStatusList (class in taniumpy.object\_types.system\_status\_list), 587
- ## T
- taniumpy (module), 579  
 taniumpy.object\_types (module), 579  
 taniumpy.object\_types.action (module), 579  
 taniumpy.object\_types.action\_list (module), 579  
 taniumpy.object\_types.action\_list\_info (module), 579  
 taniumpy.object\_types.action\_stop (module), 579  
 taniumpy.object\_types.action\_stop\_list (module), 579  
 taniumpy.object\_types.all\_objects (module), 579  
 taniumpy.object\_types.archived\_question (module), 579

`taniumpy.object_types.archived_question_list` (module), 579

`taniumpy.object_types.audit_data` (module), 580

`taniumpy.object_types.base` (module), 580

`taniumpy.object_types.cache_filter` (module), 581

`taniumpy.object_types.cache_filter_list` (module), 581

`taniumpy.object_types.cache_info` (module), 581

`taniumpy.object_types.client_count` (module), 581

`taniumpy.object_types.client_status` (module), 581

`taniumpy.object_types.column` (module), 581

`taniumpy.object_types.column_set` (module), 581

`taniumpy.object_types.computer_group` (module), 581

`taniumpy.object_types.computer_group_list` (module), 581

`taniumpy.object_types.computer_group_spec` (module), 581

`taniumpy.object_types.computer_spec_list` (module), 581

`taniumpy.object_types.error_list` (module), 582

`taniumpy.object_types.filter` (module), 582

`taniumpy.object_types.filter_list` (module), 582

`taniumpy.object_types.group` (module), 582

`taniumpy.object_types.group_list` (module), 582

`taniumpy.object_types.metadata_item` (module), 582

`taniumpy.object_types.metadata_list` (module), 582

`taniumpy.object_types.object_list` (module), 582

`taniumpy.object_types.object_list_types` (module), 582

`taniumpy.object_types.options` (module), 582

`taniumpy.object_types.package_file` (module), 582

`taniumpy.object_types.package_file_list` (module), 582

`taniumpy.object_types.package_file_status` (module), 582

`taniumpy.object_types.package_file_status_list` (module), 582

`taniumpy.object_types.package_file_template` (module), 583

`taniumpy.object_types.package_file_template_list` (module), 583

`taniumpy.object_types.package_spec` (module), 583

`taniumpy.object_types.package_spec_list` (module), 583

`taniumpy.object_types.parameter` (module), 583

`taniumpy.object_types.parameter_list` (module), 583

`taniumpy.object_types.parse_job` (module), 583

`taniumpy.object_types.parse_job_list` (module), 583

`taniumpy.object_types.parse_result` (module), 583

`taniumpy.object_types.parse_result_group` (module), 583

`taniumpy.object_types.parse_result_group_list` (module), 583

`taniumpy.object_types.parse_result_list` (module), 583

`taniumpy.object_types.permission_list` (module), 583

`taniumpy.object_types.plugin` (module), 583

`taniumpy.object_types.plugin_argument` (module), 584

`taniumpy.object_types.plugin_argument_list` (module), 584

`taniumpy.object_types.plugin_command_list` (module), 584

`taniumpy.object_types.plugin_list` (module), 584

`taniumpy.object_types.plugin_schedule` (module), 584

`taniumpy.object_types.plugin_schedule_list` (module), 584

`taniumpy.object_types.plugin_sql` (module), 584

`taniumpy.object_types.plugin_sql_column` (module), 584

`taniumpy.object_types.plugin_sql_result` (module), 584

`taniumpy.object_types.question` (module), 584

`taniumpy.object_types.question_list` (module), 584

`taniumpy.object_types.question_list_info` (module), 584

`taniumpy.object_types.result_info` (module), 584

`taniumpy.object_types.result_set` (module), 585

`taniumpy.object_types.row` (module), 585

`taniumpy.object_types.saved_action` (module), 585

`taniumpy.object_types.saved_action_approval` (module), 585

`taniumpy.object_types.saved_action_list` (module), 585

`taniumpy.object_types.saved_action_policy` (module), 585

`taniumpy.object_types.saved_action_row_id_list` (module), 585

`taniumpy.object_types.saved_question` (module), 585

`taniumpy.object_types.saved_question_list` (module), 585

`taniumpy.object_types.select` (module), 586

`taniumpy.object_types.select_list` (module), 586

`taniumpy.object_types.sensor` (module), 586

`taniumpy.object_types.sensor_list` (module), 586

`taniumpy.object_types.sensor_query` (module), 586

`taniumpy.object_types.sensor_query_list` (module), 586

`taniumpy.object_types.sensor_subcolumn` (module), 586

`taniumpy.object_types.sensor_subcolumn_list` (module), 586

`taniumpy.object_types.sensor_types` (module), 586

`taniumpy.object_types.soap_error` (module), 586

`taniumpy.object_types.string_hint_list` (module), 586

`taniumpy.object_types.system_setting` (module), 586

`taniumpy.object_types.system_setting_list` (module), 586

`taniumpy.object_types.system_status_aggregate` (module), 586

`taniumpy.object_types.system_status_list` (module), 587

`taniumpy.object_types.upload_file` (module), 587

`taniumpy.object_types.upload_file_list` (module), 587

`taniumpy.object_types.upload_file_status` (module), 587

`taniumpy.object_types.user` (module), 587

`taniumpy.object_types.user_list` (module), 587

`taniumpy.object_types.user_role` (module), 587

`taniumpy.object_types.user_role_list` (module), 587

`taniumpy.object_types.version_aggregate` (module), 587

`taniumpy.object_types.version_aggregate_list` (module), 587

`taniumpy.object_types.white_listed_url` (module), 587

`taniumpy.object_types.white_listed_url_list` (module), 587

`taniumpy.object_types.xml_error` (module), 587

tearDownClass() (test_pytan_valid_server_tests.ValidServerTests class method), 87	(test_pytan_unit.TestDehumanizeExtractionUtils method), 82
test_app_port() (in module pytan.utils), 71	test_extract_filter_valid() (test_pytan_unit.TestDehumanizeExtractionUtils method), 83
test_bad_chars_basetype_control() (test_pytan_unit.TestDeserializeBadXML method), 84	test_extract_filter_valid_all() (test_pytan_unit.TestDehumanizeExtractionUtils method), 83
test_bad_chars_resultset_latin1() (test_pytan_unit.TestDeserializeBadXML method), 84	test_extract_options_invalid_option() (test_pytan_unit.TestDehumanizeExtractionUtils method), 83
test_bad_chars_resultset_surrogate() (test_pytan_unit.TestDeserializeBadXML method), 84	test_extract_options_many() (test_pytan_unit.TestDehumanizeExtractionUtils method), 83
test_build_group_obj() (test_pytan_unit.TestManualBuildObjectUtils method), 85	test_extract_options_missing_value_max_data_age() (test_pytan_unit.TestDehumanizeExtractionUtils method), 83
test_build_manual_q() (test_pytan_unit.TestManualBuildObjectUtils method), 85	test_extract_options_missing_value_value_type() (test_pytan_unit.TestDehumanizeExtractionUtils method), 83
test_build_selectlist_obj_invalid_filter() (test_pytan_unit.TestManualBuildObjectUtils method), 85	test_extract_options_nooptions() (test_pytan_unit.TestDehumanizeExtractionUtils method), 83
test_build_selectlist_obj_missing_value() (test_pytan_unit.TestManualBuildObjectUtils method), 85	test_extract_options_single() (test_pytan_unit.TestDehumanizeExtractionUtils method), 83
test_build_selectlist_obj_noparamssensorobj_noparams() (test_pytan_unit.TestManualBuildObjectUtils method), 85	test_extract_params() (test_pytan_unit.TestDehumanizeExtractionUtils method), 83
test_build_selectlist_obj_noparamssensorobj_withparams() (test_pytan_unit.TestManualBuildObjectUtils method), 85	test_extract_params_missing_seperator() (test_pytan_unit.TestDehumanizeExtractionUtils method), 83
test_build_selectlist_obj_withparamssensorobj_noparams() (test_pytan_unit.TestManualBuildObjectUtils method), 85	test_extract_params_multiparams() (test_pytan_unit.TestDehumanizeExtractionUtils method), 83
test_build_selectlist_obj_withparamssensorobj_withparams() (test_pytan_unit.TestManualBuildObjectUtils method), 85	test_extract_params_noparams() (test_pytan_unit.TestDehumanizeExtractionUtils method), 83
test_empty_args_dict() (test_pytan_unit.TestDehumanizeSensorUtils method), 83	test_extract_selector() (test_pytan_unit.TestDehumanizeExtractionUtils method), 83
test_empty_args_list() (test_pytan_unit.TestDehumanizeSensorUtils method), 84	test_extract_selector_use_name_if_noselector() (test_pytan_unit.TestDehumanizeExtractionUtils method), 83
test_empty_args_str() (test_pytan_unit.TestDehumanizeSensorUtils method), 84	test_filter_list() (test_pytan_unit.TestDehumanizeExtractionUtils method), 83
test_empty_filterlist() (test_pytan_unit.TestDehumanizeQuestionFilterUtils method), 83	test_get_obj_map() (test_pytan_unit.TestGenericUtils method), 84
test_empty_filterstr() (test_pytan_unit.TestDehumanizeQuestionFilterUtils method), 83	test_get_obj_map_only_if() (test_pytan_unit.TestGenericUtils method), 84
test_empty_obj() (test_pytan_unit.TestGenericUtils method), 84	test_optionlist() (test_pytan_unit.TestManualPackageDefValidateUtils method), 85
test_empty_optionlist() (test_pytan_unit.TestDehumanizeQuestionFilterUtils method), 83	test_invalid1() (test_pytan_unit.TestManualQuestionFilterDefValidateUtils method), 86
test_empty_optionstr() (test_pytan_unit.TestDehumanizeQuestionFilterUtils method), 83	test_invalid1() (test_pytan_unit.TestManualSensorDefValidateUtils method), 87
test_extract_filter_invalid() (test_pytan_unit.TestDehumanizeExtractionUtils method), 82	
test_extract_filter_nofilter()	

test\_invalid2() (test\_pytan\_unit.TestManualPackageDefValidateUtils (test\_pytan\_valid\_server\_tests.ValidServerTests method), 85 method), 87

test\_invalid2() (test\_pytan\_unit.TestManualSensorDefValidateUtils (test\_pytan\_valid\_server\_tests.ValidServerTests method), 87 method), 87

test\_invalid3() (test\_pytan\_unit.TestManualSensorDefValidateUtils (test\_pytan\_valid\_server\_tests.ValidServerTests method), 87 method), 87

test\_invalid4() (test\_pytan\_unit.TestManualSensorDefValidateUtils (test\_pytan\_valid\_server\_tests.ValidServerTests method), 87 method), 88

test\_invalid\_connect\_1\_bad\_username() (test\_pytan\_invalid\_server\_tests.InvalidServerTests method), 91 test\_invalid\_export\_basetype\_3\_invalid\_export\_basetype\_csv\_bad\_sort\_type() (test\_pytan\_valid\_server\_tests.ValidServerTests method), 88

test\_invalid\_connect\_2\_bad\_host\_and\_non\_ssl\_port() (test\_pytan\_invalid\_server\_tests.InvalidServerTests method), 91 test\_invalid\_export\_basetype\_4\_invalid\_export\_basetype\_xml\_bad\_minimal() (test\_pytan\_valid\_server\_tests.ValidServerTests method), 88

test\_invalid\_connect\_3\_bad\_password() (test\_pytan\_invalid\_server\_tests.InvalidServerTests method), 91 test\_invalid\_export\_basetype\_5\_invalid\_export\_basetype\_json\_bad\_include() (test\_pytan\_valid\_server\_tests.ValidServerTests method), 88

test\_invalid\_connect\_4\_bad\_host\_and\_bad\_port() (test\_pytan\_invalid\_server\_tests.InvalidServerTests method), 91 test\_invalid\_export\_basetype\_6\_invalid\_export\_basetype\_json\_bad\_explode() (test\_pytan\_valid\_server\_tests.ValidServerTests method), 88

test\_invalid\_create\_object\_1\_invalid\_create\_sensor() (test\_pytan\_valid\_server\_tests.ValidServerTests method), 87 test\_invalid\_export\_basetype\_7\_invalid\_export\_basetype\_bad\_format() (test\_pytan\_valid\_server\_tests.ValidServerTests method), 88

test\_invalid\_create\_object\_from\_json\_1\_invalid\_create\_savedata\_invalid\_export\_resultset\_1\_invalid\_export\_resultset\_csv\_bad\_sort\_sub() (test\_pytan\_valid\_server\_tests.ValidServerTests method), 87 test\_invalid\_export\_resultset\_1\_invalid\_export\_resultset\_csv\_bad\_sort\_sub() (test\_pytan\_valid\_server\_tests.ValidServerTests method), 88

test\_invalid\_create\_object\_from\_json\_2\_invalid\_create\_client\_invalid\_export\_resultset\_2\_invalid\_export\_resultset\_csv\_bad\_sort\_type() (test\_pytan\_valid\_server\_tests.ValidServerTests method), 87 test\_invalid\_export\_resultset\_2\_invalid\_export\_resultset\_csv\_bad\_sort\_type() (test\_pytan\_valid\_server\_tests.ValidServerTests method), 88

test\_invalid\_create\_object\_from\_json\_3\_invalid\_create\_user\_invalid\_export\_resultset\_3\_invalid\_export\_resultset\_csv\_bad\_expand() (test\_pytan\_valid\_server\_tests.ValidServerTests method), 87 test\_invalid\_export\_resultset\_3\_invalid\_export\_resultset\_csv\_bad\_expand() (test\_pytan\_valid\_server\_tests.ValidServerTests method), 88

test\_invalid\_create\_object\_from\_json\_4\_invalid\_create\_settings\_invalid\_export\_resultset\_4\_invalid\_export\_resultset\_csv\_bad\_sensors() (test\_pytan\_valid\_server\_tests.ValidServerTests method), 87 test\_invalid\_export\_resultset\_4\_invalid\_export\_resultset\_csv\_bad\_sensors() (test\_pytan\_valid\_server\_tests.ValidServerTests method), 88

test\_invalid\_deploy\_action\_1\_invalid\_deploy\_action\_run\_false() (test\_pytan\_valid\_server\_tests.ValidServerTests method), 87 test\_invalid\_export\_resultset\_5\_invalid\_export\_resultset\_bad\_format() (test\_pytan\_valid\_server\_tests.ValidServerTests method), 88

test\_invalid\_deploy\_action\_2\_invalid\_deploy\_action\_package\_help() (test\_pytan\_valid\_server\_tests.ValidServerTests method), 87 test\_invalid\_filter1() (test\_pytan\_unit.TestDehumanizeQuestionFilterUtils method), 83

test\_invalid\_deploy\_action\_3\_invalid\_deploy\_action\_package() (test\_pytan\_valid\_server\_tests.ValidServerTests method), 87 test\_invalid\_filter2() (test\_pytan\_unit.TestDehumanizeQuestionFilterUtils method), 83

test\_invalid\_deploy\_action\_4\_invalid\_deploy\_action\_options\_help() (test\_pytan\_valid\_server\_tests.ValidServerTests method), 87 test\_invalid\_filter3() (test\_pytan\_unit.TestDehumanizeQuestionFilterUtils method), 83

test\_invalid\_deploy\_action\_5\_invalid\_deploy\_action\_empty\_package() (test\_pytan\_valid\_server\_tests.ValidServerTests method), 87 test\_invalid\_get\_object\_1\_invalid\_get\_action\_single\_by\_name() (test\_pytan\_valid\_server\_tests.ValidServerTests method), 88

test\_invalid\_deploy\_action\_6\_invalid\_deploy\_action\_filters\_help() (test\_pytan\_valid\_server\_tests.ValidServerTests method), 87 test\_invalid\_get\_object\_2\_invalid\_get\_question\_by\_name() (test\_pytan\_valid\_server\_tests.ValidServerTests method), 88

test\_invalid\_deploy\_action\_7\_invalid\_deploy\_action\_missing\_parameters() (test\_pytan\_valid\_server\_tests.ValidServerTests method), 87 test\_invalid\_option1() (test\_pytan\_unit.TestDehumanizeQuestionOptionUtils method), 83

test\_invalid\_deploy\_action\_7\_invalid\_deploy\_action\_missing\_parameters() (test\_pytan\_valid\_server\_tests.ValidServerTests method), 87 test\_invalid\_option2() (test\_pytan\_unit.TestDehumanizeQuestionOptionUtils method), 83

test_invalid_port()	(test_pytan_unit.TestGenericUtils method), 84	(test_pytan_unit.TestGenericUtils method), 85
test_invalid_question_1_invalid_ask_manual_question_sensor_help()	(test_pytan_valid_server_tests.ValidServerTests method), 88	test_invalid_json_file_invalid_json()
test_invalid_question_2_invalid_ask_manual_question_bad_option()	(test_pytan_valid_server_tests.ValidServerTests method), 88	(test_pytan_unit.TestGenericUtils method), 85
test_invalid_question_3_invalid_ask_manual_question_filter_help()	(test_pytan_valid_server_tests.ValidServerTests method), 88	test_multi_filter_list()
test_invalid_question_4_invalid_ask_manual_question_bad_option()	(test_pytan_valid_server_tests.ValidServerTests method), 88	(test_pytan_unit.TestDehumanizeQuestionFilterUtils method), 83
test_invalid_question_5_invalid_ask_manual_question_missing_option()	(test_pytan_valid_server_tests.ValidServerTests method), 88	test_multi_list_complex()
test_invalid_question_6_invalid_ask_manual_question_option_help()	(test_pytan_valid_server_tests.ValidServerTests method), 88	(test_pytan_unit.TestDehumanizeSensorUtils method), 84
test_invalid_question_7_invalid_ask_manual_question_too_many_parameters()	(test_pytan_valid_server_tests.ValidServerTests method), 88	test_option_list_many()
test_invalid_question_8_invalid_ask_manual_question_bad_sensorname()	(test_pytan_valid_server_tests.ValidServerTests method), 88	(test_pytan_unit.TestDehumanizeQuestionOptionUtils method), 83
test_is_dict()	(test_pytan_unit.TestGenericUtils method), 84	test_option_list_multi()
test_is_list()	(test_pytan_unit.TestGenericUtils method), 84	(test_pytan_unit.TestDehumanizeQuestionOptionUtils method), 83
test_is_not_dict()	(test_pytan_unit.TestGenericUtils method), 84	test_option_list_split()
test_is_not_list()	(test_pytan_unit.TestGenericUtils method), 84	(test_pytan_unit.TestDehumanizeQuestionOptionUtils method), 83
test_is_not_num()	(test_pytan_unit.TestGenericUtils method), 84	test_option_str()
test_is_not_str()	(test_pytan_unit.TestGenericUtils method), 85	(test_pytan_unit.TestDehumanizeQuestionOptionUtils method), 83
test_is_num()	(test_pytan_unit.TestGenericUtils method), 85	test_parse_complex()
test_is_str()	(test_pytan_unit.TestGenericUtils method), 85	(test_pytan_unit.TestManualSensorDefParseUtils method), 86
test_jsonify()	(test_pytan_unit.TestGenericUtils method), 85	test_parse_dict_check_keys()
test_load_param_file_invalid_file()	(test_pytan_unit.TestGenericUtils method), 85	(test_pytan_unit.TestManualSensorDefParseUtils method), 86
test_load_param_file_invalid_json()	(test_pytan_unit.TestGenericUtils method), 85	test_parse_dict_id()
test_load_param_file_valid()	(test_pytan_unit.TestGenericUtils method), 85	(test_pytan_unit.TestManualSensorDefParseUtils method), 86
test_load_taniumpy_file_invalid_file()		test_parse_dict_name()
		(test_pytan_unit.TestManualSensorDefParseUtils method), 86
		test_parse_emptydict()
		(test_pytan_unit.TestManualQuestionFilterDefParseUtils method), 85
		test_parse_emptydict()
		(test_pytan_unit.TestManualQuestionOptionDefParseUtils method), 86
		test_parse_emptydict()
		(test_pytan_unit.TestManualSensorDefParseUtils method), 86
		test_parse_emptylist()
		(test_pytan_unit.TestManualQuestionFilterDefParseUtils method), 85
		test_parse_emptylist()
		(test_pytan_unit.TestManualQuestionOptionDefParseUtils method), 86
		test_parse_emptylist()
		(test_pytan_unit.TestManualSensorDefParseUtils method), 86
		test_parse_emptystr()
		(test_pytan_unit.TestManualQuestionFilterDefParseUtils method), 85
		test_parse_emptystr()
		(test_pytan_unit.TestManualQuestionOptionDefParseUtils method), 86
		test_parse_emptystr()
		(test_pytan_unit.TestManualSensorDefParseUtils method), 86
		test_parse_list()
		(test_pytan_unit.TestManualQuestionOptionDefParseUtils method), 86
		test_parse_multi_filter()
		(test_pytan_unit.TestManualQuestionFilterDefParseUtils method), 86
		test_parse_noargs()
		(test_pytan_unit.TestManualQuestionFilterDefParseUtils method), 86
		test_parse_noargs()
		(test_pytan_unit.TestManualQuestionOptionDefParseUtils method), 86
		test_parse_noargs()
		(test_pytan_unit.TestManualSensorDefParseUtils method), 86



[test\\_parse\\_none\(\) \(test\\_pytan\\_unit.TestManualQuestionFilterDefParseUtils method\), 86](#)  
[test\\_parse\\_none\(\) \(test\\_pytan\\_unit.TestManualQuestionOptionDefParseUtils method\), 86](#)  
[test\\_parse\\_none\(\) \(test\\_pytan\\_unit.TestManualSensorDefParseUtils method\), 86](#)  
[test\\_parse\\_options\\_dict\(\) \(test\\_pytan\\_unit.TestManualQuestionOptionDefParseUtils method\), 86](#)  
[test\\_parse\\_single\\_filter\(\) \(test\\_pytan\\_unit.TestManualQuestionFilterDefParseUtils method\), 86](#)  
[test\\_parse\\_str\(\) \(test\\_pytan\\_unit.TestManualQuestionFilterDefParseUtils method\), 86](#)  
[test\\_parse\\_str\(\) \(test\\_pytan\\_unit.TestManualQuestionOptionDefParseUtils method\), 86](#)  
[test\\_parse\\_str1\(\) \(test\\_pytan\\_unit.TestManualSensorDefParseUtils method\), 87](#)  
[test\\_pytan\\_invalid\\_server\\_tests \(module\), 91](#)  
[test\\_pytan\\_unit \(module\), 82](#)  
[test\\_pytan\\_valid\\_server\\_tests \(module\), 87](#)  
[test\\_single\\_filter\\_list\(\) \(test\\_pytan\\_unit.TestDehumanizeQuestionFilterUtils method\), 83](#)  
[test\\_single\\_filter\\_str\(\) \(test\\_pytan\\_unit.TestDehumanizeQuestionFilterUtils method\), 83](#)  
[test\\_single\\_str\(\) \(test\\_pytan\\_unit.TestDehumanizeSensorUtils method\), 84](#)  
[test\\_single\\_str\\_complex1\(\) \(test\\_pytan\\_unit.TestDehumanizeSensorUtils method\), 84](#)  
[test\\_single\\_str\\_complex2\(\) \(test\\_pytan\\_unit.TestDehumanizeSensorUtils method\), 84](#)  
[test\\_single\\_str\\_with\\_filter\(\) \(test\\_pytan\\_unit.TestDehumanizeSensorUtils method\), 84](#)  
[test\\_valid1\(\) \(test\\_pytan\\_unit.TestManualPackageDefValidateUtils method\), 85](#)  
[test\\_valid1\(\) \(test\\_pytan\\_unit.TestManualQuestionFilterDefValidateUtils method\), 86](#)  
[test\\_valid1\(\) \(test\\_pytan\\_unit.TestManualSensorDefValidateUtils method\), 87](#)  
[test\\_valid2\(\) \(test\\_pytan\\_unit.TestManualPackageDefValidateUtils method\), 85](#)  
[test\\_valid2\(\) \(test\\_pytan\\_unit.TestManualQuestionFilterDefValidateUtils method\), 86](#)  
[test\\_valid2\(\) \(test\\_pytan\\_unit.TestManualSensorDefValidateUtils method\), 87](#)  
[test\\_valid3\(\) \(test\\_pytan\\_unit.TestManualSensorDefValidateUtils method\), 87](#)  
[test\\_valid4\(\) \(test\\_pytan\\_unit.TestManualSensorDefValidateUtils method\), 87](#)  
[test\\_valid\\_create\\_object\\_1\\_create\\_user\(\) \(test\\_pytan\\_valid\\_server\\_tests.ValidServerTests method\), 88](#)  
[test\\_valid\\_create\\_object\\_2\\_create\\_package\(\) \(test\\_pytan\\_valid\\_server\\_tests.ValidServerTests method\), 88](#)  
[test\\_valid\\_create\\_object\\_3\\_create\\_group\(\) \(test\\_pytan\\_valid\\_server\\_tests.ValidServerTests method\), 88](#)  
[test\\_valid\\_create\\_object\\_4\\_create\\_whitelisted\\_url\(\) \(test\\_pytan\\_valid\\_server\\_tests.ValidServerTests method\), 88](#)  
[test\\_valid\\_create\\_object\\_from\\_json\\_1\\_create\\_package\\_from\\_json\(\) \(test\\_pytan\\_valid\\_server\\_tests.ValidServerTests method\), 88](#)  
[test\\_valid\\_create\\_object\\_from\\_json\\_2\\_create\\_user\\_from\\_json\(\) \(test\\_pytan\\_valid\\_server\\_tests.ValidServerTests method\), 88](#)  
[test\\_valid\\_create\\_object\\_from\\_json\\_3\\_create\\_saved\\_question\\_from\\_json\(\) \(test\\_pytan\\_valid\\_server\\_tests.ValidServerTests method\), 88](#)  
[test\\_valid\\_create\\_object\\_from\\_json\\_4\\_create\\_action\\_from\\_json\(\) \(test\\_pytan\\_valid\\_server\\_tests.ValidServerTests method\), 88](#)  
[test\\_valid\\_create\\_object\\_from\\_json\\_5\\_create\\_sensor\\_from\\_json\(\) \(test\\_pytan\\_valid\\_server\\_tests.ValidServerTests method\), 88](#)  
[test\\_valid\\_create\\_object\\_from\\_json\\_6\\_create\\_question\\_from\\_json\(\) \(test\\_pytan\\_valid\\_server\\_tests.ValidServerTests method\), 88](#)  
[test\\_valid\\_create\\_object\\_from\\_json\\_7\\_create\\_whitelisted\\_url\\_from\\_json\(\) \(test\\_pytan\\_valid\\_server\\_tests.ValidServerTests method\), 88](#)  
[test\\_valid\\_create\\_object\\_from\\_json\\_8\\_create\\_group\\_from\\_json\(\) \(test\\_pytan\\_valid\\_server\\_tests.ValidServerTests method\), 88](#)  
[test\\_valid\\_deploy\\_action\\_1\\_deploy\\_action\\_simple\\_against\\_windows\\_comp \(test\\_pytan\\_valid\\_server\\_tests.ValidServerTests method\), 88](#)  
[test\\_valid\\_deploy\\_action\\_2\\_deploy\\_action\\_simple\\_without\\_results\(\) \(test\\_pytan\\_valid\\_server\\_tests.ValidServerTests method\), 88](#)  
[test\\_valid\\_deploy\\_action\\_3\\_deploy\\_action\\_with\\_params\\_against\\_windows\\_comp \(test\\_pytan\\_valid\\_server\\_tests.ValidServerTests method\), 88](#)  
[test\\_valid\\_deploy\\_action\\_4\\_deploy\\_action\\_simple\(\) \(test\\_pytan\\_valid\\_server\\_tests.ValidServerTests method\), 88](#)  
[test\\_valid\\_export\\_basetype\\_10\\_export\\_basetype\\_xml\\_default\\_options\(\) \(test\\_pytan\\_valid\\_server\\_tests.ValidServerTests method\), 89](#)  
[test\\_valid\\_export\\_basetype\\_11\\_export\\_basetype\\_csv\\_with\\_explode\\_true\(\) \(test\\_pytan\\_valid\\_server\\_tests.ValidServerTests method\), 89](#)  
[test\\_valid\\_export\\_basetype\\_12\\_export\\_basetype\\_json\\_explode\\_false\(\) \(test\\_pytan\\_valid\\_server\\_tests.ValidServerTests method\), 89](#)

---

```

test_valid_export_basetype_13_export_basetype_json_type_false() test_valid_export_resultset_4_export_resultset_csv_expand_false()
    (test_pytan_valid_server_tests.ValidServerTests (test_pytan_valid_server_tests.ValidServerTests
    method), 89 method), 89
test_valid_export_basetype_14_export_basetype_json_defaults_optional() test_valid_export_resultset_5_export_resultset_csv_sort_empty()
    (test_pytan_valid_server_tests.ValidServerTests (test_pytan_valid_server_tests.ValidServerTests
    method), 89 method), 89
test_valid_export_basetype_1_export_basetype_csv_with_sort_list() test_valid_export_resultset_6_export_resultset_csv_sort_true()
    (test_pytan_valid_server_tests.ValidServerTests (test_pytan_valid_server_tests.ValidServerTests
    method), 89 method), 89
test_valid_export_basetype_2_export_basetype_csv_with_explicit_filters() test_valid_export_resultset_7_export_resultset_csv_sort_list()
    (test_pytan_valid_server_tests.ValidServerTests (test_pytan_valid_server_tests.ValidServerTests
    method), 89 method), 89
test_valid_export_basetype_3_export_basetype_json_type_true() test_valid_export_resultset_8_export_resultset_csv_sensor_false()
    (test_pytan_valid_server_tests.ValidServerTests (test_pytan_valid_server_tests.ValidServerTests
    method), 89 method), 89
test_valid_export_basetype_4_export_basetype_xml_minimal() test_valid_export_resultset_9_export_resultset_csv_expand_true()
    (test_pytan_valid_server_tests.ValidServerTests (test_pytan_valid_server_tests.ValidServerTests
    method), 89 method), 89
test_valid_export_basetype_5_export_basetype_xml_minimal() test_valid_get_object_10_get_all_saved_questions()
    (test_pytan_valid_server_tests.ValidServerTests (test_pytan_valid_server_tests.ValidServerTests
    method), 89 method), 89
test_valid_export_basetype_6_export_basetype_csv_with_sort_enabled() test_valid_get_object_11_get_user_by_name()
    (test_pytan_valid_server_tests.ValidServerTests (test_pytan_valid_server_tests.ValidServerTests
    method), 89 method), 89
test_valid_export_basetype_7_export_basetype_csv_defaults_optional() test_valid_get_object_12_get_all_userroleless()
    (test_pytan_valid_server_tests.ValidServerTests (test_pytan_valid_server_tests.ValidServerTests
    method), 89 method), 89
test_valid_export_basetype_8_export_basetype_json_explode_true() test_valid_get_object_13_get_all_questions()
    (test_pytan_valid_server_tests.ValidServerTests (test_pytan_valid_server_tests.ValidServerTests
    method), 89 method), 89
test_valid_export_basetype_9_export_basetype_csv_with_sort_true() test_valid_get_object_14_get_sensor_by_id()
    (test_pytan_valid_server_tests.ValidServerTests (test_pytan_valid_server_tests.ValidServerTests
    method), 89 method), 89
test_valid_export_resultset_10_export_resultset_csv_defaults_optional() test_valid_get_object_15_get_all_groups()
    (test_pytan_valid_server_tests.ValidServerTests (test_pytan_valid_server_tests.ValidServerTests
    method), 89 method), 89
test_valid_export_resultset_11_export_resultset_csv_type_true() test_valid_get_object_16_get_all_sensors()
    (test_pytan_valid_server_tests.ValidServerTests (test_pytan_valid_server_tests.ValidServerTests
    method), 89 method), 89
test_valid_export_resultset_12_export_resultset_csv_all_options() test_valid_get_object_17_get_sensor_by_mixed()
    (test_pytan_valid_server_tests.ValidServerTests (test_pytan_valid_server_tests.ValidServerTests
    method), 89 method), 89
test_valid_export_resultset_13_export_resultset_csv_sort_false() test_valid_get_object_18_get_whitelisted_url_by_id()
    (test_pytan_valid_server_tests.ValidServerTests (test_pytan_valid_server_tests.ValidServerTests
    method), 89 method), 89
test_valid_export_resultset_1_export_resultset_json() test_valid_get_object_19_get_group_by_name()
    (test_pytan_valid_server_tests.ValidServerTests (test_pytan_valid_server_tests.ValidServerTests
    method), 89 method), 90
test_valid_export_resultset_2_export_resultset_csv_sensor_true() test_valid_get_object_1_get_all_users()
    (test_pytan_valid_server_tests.ValidServerTests (test_pytan_valid_server_tests.ValidServerTests
    method), 89 method), 90
test_valid_export_resultset_3_export_resultset_csv_type_false() test_valid_get_object_20_get_all_whitelisted_urls()
    (test_pytan_valid_server_tests.ValidServerTests (test_pytan_valid_server_tests.ValidServerTests
    method), 89 method), 90

```

<code>test_valid_get_object_21_get_sensor_by_hash()</code> (test_pytan_valid_server_tests.ValidServerTests method), 90	<code>test_valid_question_10_ask_manual_question_sensor_with_filter()</code> (test_pytan_valid_server_tests.ValidServerTests method), 90
<code>test_valid_get_object_22_get_package_by_name()</code> (test_pytan_valid_server_tests.ValidServerTests method), 90	<code>test_valid_question_11_ask_manual_question_multiple_sensors_identified()</code> (test_pytan_valid_server_tests.ValidServerTests method), 90
<code>test_valid_get_object_23_get_all_clients()</code> (test_pytan_valid_server_tests.ValidServerTests method), 90	<code>test_valid_question_12_ask_manual_question_sensor_with_parameters_and_filters()</code> (test_pytan_valid_server_tests.ValidServerTests method), 90
<code>test_valid_get_object_24_get_sensor_by_names()</code> (test_pytan_valid_server_tests.ValidServerTests method), 90	<code>test_valid_question_13_ask_manual_question_sensor_with_filter_and_3_options()</code> (test_pytan_valid_server_tests.ValidServerTests method), 90
<code>test_valid_get_object_25_get_all_packages()</code> (test_pytan_valid_server_tests.ValidServerTests method), 90	<code>test_valid_question_14_ask_manual_question_complex_query2()</code> (test_pytan_valid_server_tests.ValidServerTests method), 90
<code>test_valid_get_object_26_get_saved_question_by_name()</code> (test_pytan_valid_server_tests.ValidServerTests method), 90	<code>test_valid_question_15_ask_manual_question_complex_query1()</code> (test_pytan_valid_server_tests.ValidServerTests method), 90
<code>test_valid_get_object_27_get_all_actions()</code> (test_pytan_valid_server_tests.ValidServerTests method), 90	<code>test_valid_question_1_ask_manual_question_sensor_with_parameters_and_filters()</code> (test_pytan_valid_server_tests.ValidServerTests method), 90
<code>test_valid_get_object_28_get_user_by_id()</code> (test_pytan_valid_server_tests.ValidServerTests method), 90	<code>test_valid_question_2_ask_manual_question_multiple_sensors_with_parameters()</code> (test_pytan_valid_server_tests.ValidServerTests method), 90
<code>test_valid_get_object_29_get_sensor_by_name()</code> (test_pytan_valid_server_tests.ValidServerTests method), 90	<code>test_valid_question_3_ask_manual_question_simple_multiple_sensors()</code> (test_pytan_valid_server_tests.ValidServerTests method), 90
<code>test_valid_get_object_2_get_action_by_id()</code> (test_pytan_valid_server_tests.ValidServerTests method), 90	<code>test_valid_question_4_ask_manual_question_sensor_without_parameters_and_filters()</code> (test_pytan_valid_server_tests.ValidServerTests method), 90
<code>test_valid_get_object_30_get_saved_action_by_name()</code> (test_pytan_valid_server_tests.ValidServerTests method), 90	<code>test_valid_question_5_ask_manual_question_sensor_with_filter_and_2_options()</code> (test_pytan_valid_server_tests.ValidServerTests method), 90
<code>test_valid_get_object_3_get_question_by_id()</code> (test_pytan_valid_server_tests.ValidServerTests method), 90	<code>test_valid_question_6_ask_manual_question_sensor_with_parameters_and_filters()</code> (test_pytan_valid_server_tests.ValidServerTests method), 90
<code>test_valid_get_object_4_get_saved_question_by_names()</code> (test_pytan_valid_server_tests.ValidServerTests method), 90	<code>test_valid_question_7_ask_manual_question_sensor_complex()</code> (test_pytan_valid_server_tests.ValidServerTests method), 90
<code>test_valid_get_object_5_get_userrole_by_id()</code> (test_pytan_valid_server_tests.ValidServerTests method), 90	<code>test_valid_question_8_ask_manual_question_sensor_with_parameters_and_filters()</code> (test_pytan_valid_server_tests.ValidServerTests method), 90
<code>test_valid_get_object_6_get_all_saved_actions()</code> (test_pytan_valid_server_tests.ValidServerTests method), 90	<code>test_valid_question_9_ask_manual_question_simple_single_sensor()</code> (test_pytan_valid_server_tests.ValidServerTests method), 90
<code>test_valid_get_object_7_get_leader_clients()</code> (test_pytan_valid_server_tests.ValidServerTests method), 90	<code>test_valid_saved_question_1_ask_saved_question_refresh_data()</code> (test_pytan_valid_server_tests.ValidServerTests method), 91
<code>test_valid_get_object_8_get_all_settings()</code> (test_pytan_valid_server_tests.ValidServerTests method), 90	<code>test_valid_saved_question_2_ask_saved_question_by_name()</code> (test_pytan_valid_server_tests.ValidServerTests method), 91
<code>test_valid_get_object_9_get_setting_by_name()</code> (test_pytan_valid_server_tests.ValidServerTests method), 90	<code>test_valid_saved_question_3_ask_saved_question_by_name_in_list()</code> (test_pytan_valid_server_tests.ValidServerTests method), 91



- test\_valid\_simple\_list() (test\_pytan\_unit.TestDehumanizeSensorUtils static method), 585
- test\_valid\_simple\_str\_hash\_selector() (test\_pytan\_unit.TestDehumanizeSensorUtils method), 84
- test\_valid\_simple\_str\_id\_selector() (test\_pytan\_unit.TestDehumanizeSensorUtils method), 84
- test\_valid\_simple\_str\_name\_selector() (test\_pytan\_unit.TestDehumanizeSensorUtils method), 84
- test\_version\_higher() (test\_pytan\_unit.TestGenericUtils method), 85
- test\_version\_lower() (test\_pytan\_unit.TestGenericUtils method), 85
- TestDehumanizeExtractionUtils (class in test\_pytan\_unit), 82
- TestDehumanizeQuestionFilterUtils (class in test\_pytan\_unit), 83
- TestDehumanizeQuestionOptionUtils (class in test\_pytan\_unit), 83
- TestDehumanizeSensorUtils (class in test\_pytan\_unit), 83
- TestDeserializeBadXML (class in test\_pytan\_unit), 84
- TestGenericUtils (class in test\_pytan\_unit), 84
- TestManualBuildObjectUtils (class in test\_pytan\_unit), 85
- TestManualPackageDefValidateUtils (class in test\_pytan\_unit), 85
- TestManualQuestionFilterDefParseUtils (class in test\_pytan\_unit), 85
- TestManualQuestionFilterDefValidateUtils (class in test\_pytan\_unit), 86
- TestManualQuestionOptionDefParseUtils (class in test\_pytan\_unit), 86
- TestManualSensorDefParseUtils (class in test\_pytan\_unit), 86
- TestManualSensorDefValidateUtils (class in test\_pytan\_unit), 87
- threaded\_http (module), 590
- threaded\_http() (in module threaded\_http), 590
- ThreadedHTTPServer (class in threaded\_http), 590
- TIME\_FORMAT (in module pytan.constants), 59
- TIMEOUT\_SECS\_DEFAULT (pytan.pollers.SSEPoller attribute), 57
- TimeoutException, 34
- timestr\_to\_datetime() (in module pytan.utils), 71
- to\_flat\_dict() (taniumpy.object\_types.base.BaseType method), 580
- to\_flat\_dict\_explode\_json() (taniumpy.object\_types.base.BaseType method), 580
- to\_json() (taniumpy.object\_types.base.BaseType static method), 580
- to\_json() (taniumpy.object\_types.result\_set.ResultSet static method), 580
- to\_jsonable() (taniumpy.object\_types.base.BaseType method), 580
- to\_jsonable() (taniumpy.object\_types.result\_set.ResultSet static method), 585
- toSOAPBody() (taniumpy.object\_types.base.BaseType method), 580
- toSOAPElement() (taniumpy.object\_types.base.BaseType method), 580
- ## U
- unpack() (in module ddt), 590
- unparse() (in module xmldict), 589
- UnsupportedVersionError, 34
- UploadFile (class in taniumpy.object\_types.upload\_file), 587
- UploadFileList (class in taniumpy.object\_types.upload\_file\_list), 587
- UploadFileStatus (class in taniumpy.object\_types.upload\_file\_status), 587
- User (class in taniumpy.object\_types.user), 587
- UserList (class in taniumpy.object\_types.user\_list), 587
- UserRole (class in taniumpy.object\_types.user\_role), 587
- UserRoleList (class in taniumpy.object\_types.user\_role\_list), 587
- ## V
- val\_package\_def() (in module pytan.utils), 71
- val\_q\_filter\_defs() (in module pytan.utils), 71
- val\_sensor\_defs() (in module pytan.utils), 71
- ValidServerTests (class in test\_pytan\_valid\_server\_tests), 87
- version\_check() (in module pytan.binsupport), 80
- VersionAggregate (class in taniumpy.object\_types.version\_aggregate), 587
- VersionAggregateList (class in taniumpy.object\_types.version\_aggregate\_list), 587
- VersionMismatchError, 34
- VersionParseError, 34
- ## W
- WhiteListedUrl (class in taniumpy.object\_types.white\_listed\_url), 587
- WhiteListedUrlList (class in taniumpy.object\_types.white\_listed\_url\_list), 587
- write\_csv() (taniumpy.object\_types.base.BaseType static method), 580
- write\_csv() (taniumpy.object\_types.result\_set.ResultSet static method), 585

## X

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), [71](#)

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), [33](#)

XmlError (class in taniumpy.object\_types.xml\_error), [587](#)

XMLNS (pytan.sessions.Session attribute), [36](#)

xmldict (module), [588](#)