Creopyson Documentation

Release 0.7.6

Benjamin C.

Feb 26, 2023

Contents:

CHAPTER 1

Creopyson

Creopyson is a python library that aim to control PTC's CREO Parametric via JLink via CREOSON.

CREOSON uses JSON Requests to send commands/functions to CREO, JSON Responses are used to communicate the results of your requests.

Creopyson creates a Client to send JSON Requests to CREOSON server.

- Free software: MIT license
- Documentation: https://creopyson.readthedocs.io.

1.1 Features

Creopyson can be used to automate actions in CREO:

- Get BOM
- Manage files, Working directories
- Support Familytables
- Export 3D/2D: pdf3d, pdf, STEP, IGES, JPEG...
- Interact with layers, views
- Read/Write parameters, dimensions
- Support Windchill

Basic usage video:

See documentation for more informations…

1.2 Credits

CREOSON from Simplified Logic, Inc.

This package was created with Cookiecutter and the audreyr/cookiecutter-pypackage project template.

CHAPTER 2

Installation

2.1 Stable release

To install Creopyson, run this command in your terminal:

```
$ pip install creopyson
```

This is the preferred method to install Creopyson, as it will always install the most recent stable release.

If you don't have pip installed, this Python installation guide can guide you through the process.

2.2 From sources

The sources for Creopyson can be downloaded from the Github repo.

You can either clone the public repository:

```
$ git clone git://github.com/Zepmanbc/creopyson
```

Or download the tarball:

```
$ curl  -OL https://github.com/Zepmanbc/creopyson/tarball/master
```

Once you have a copy of the source, you can install it with:

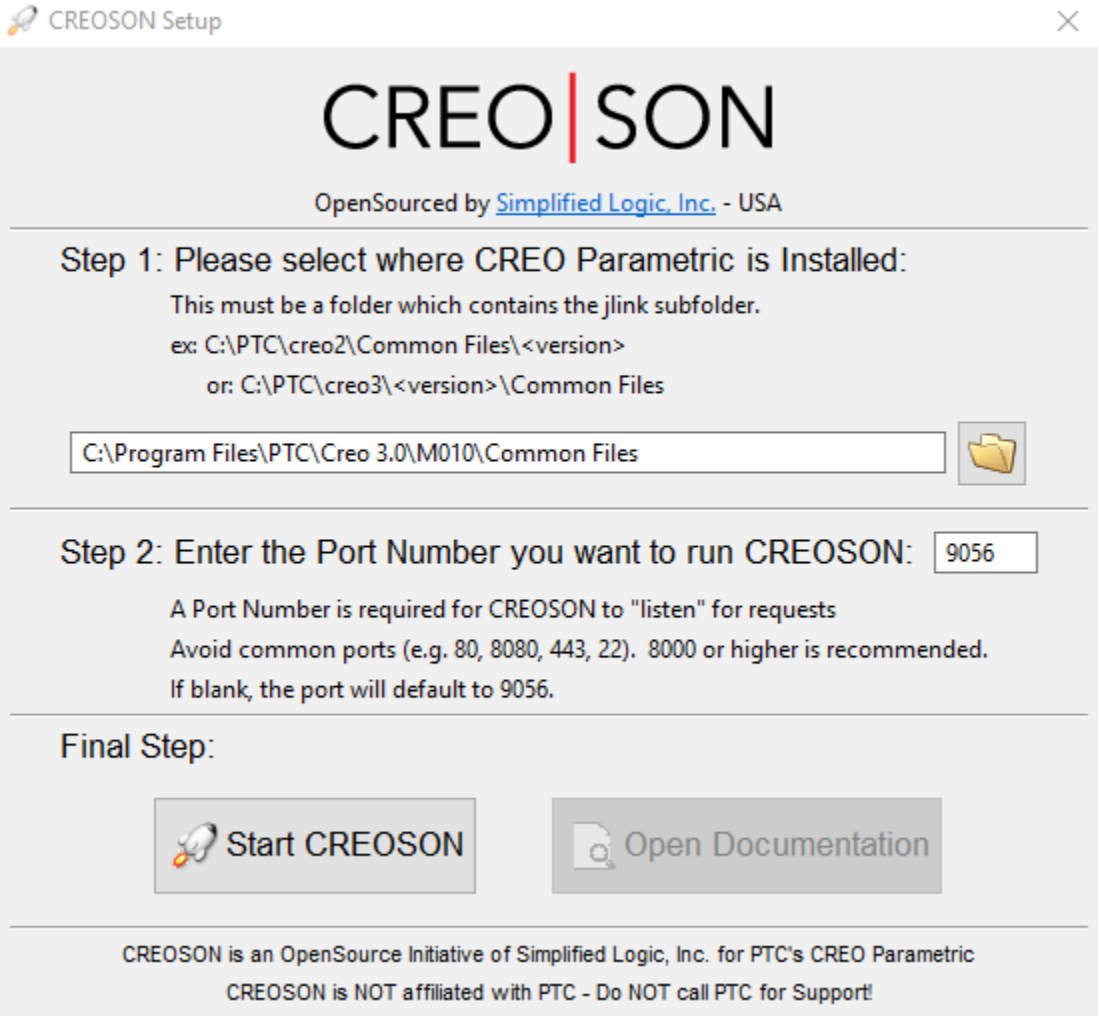
```
$ python setup.py install
```

CHAPTER 3

Usage

3.1 Quickstart

- Download last release of Creoson Server for your system.
- Run CreosonSetup and configure it with your Creo's version.



If you want to launch Creo with Creoson, please create a *nitro_proe_remote.bat* file.

You can copy *C:\Program Files\PTC\Creo x.x\Mxxx\Parametric\bin\parametric.bat* and rename it *nitro_proe_remote.bat* anywhere you want.

To use Creopyson in a project:

```
import creopyson
```

Create a Client object and create a connection with Creoson:

```
c = creopyson.Client()
c.connect()
```

Verify is Creo is running:

```
c.is_creo_running() # Return a boolean.
```

Launch Creo:

```
c.start_creo("path to nitro_proe_remote.bat")
```

Basic usage:

```
current_directory = c.creo_pwd()  # return current working directory.
listfiles = c.creo_list_files() # return a list in the working directory.
listdirs = c.creo_list_dirs() # return a list of folders in the working directory.
c.creo_cd("new_folder")  # change working directory.
c.file_exists("my_file.prt")  # verify if `my_file.prt` exists.
c.file_open("my_file.prt", display=True)  # Open `my_file.prt` in Creo.
c.dimension_set("my_file.prt", "diamm", 180)  # Modify `diamm` dimension.
c.file_regenerate("my_file.prt")  # Regenerate file, raise `Warning` if regeneration␣
↪fails.
```

3.2 Creo 7 Users

If you are using Creo 7 you must declare it once per session to prevent errors on deprecated features:

```
c.creo_set_creo_version(7)
```

3.3 «Vanilla» Creoson usage

mostly for debugging:

```
import creopyson
c = creopyson.Client()
c.connect()

# Here you define command/function
# data is a dictionnary with data part of the JSON request
# Please refer to Creoson documentation
command = "file"
function = "open"
data ={"file":"my_file.prt", "display": True}
result = c._creoson_post(command, function, data)
```

result would be the *data* part of Creoson's response:

```
{'dirname': 'C:/your/working/path/', 'files': ['my_file.prt'], 'revision': 1}
```

3.4 Logging basic usage

If you want see what are the requests to Creoson you should activate logging this way:

```
import logging
logging.basicConfig(level=logging.DEBUG)
# Hide urllib3 logging
logging.getLogger("urllib3").setLevel(logging.WARNING)
```

(continues on next page)

(continued from previous page)

```
import creopyson

c = creopyson.Client()
c.connect()

c.file_open("my_file.prt", display=True)
```

The result in you console would be something like this:

```
DEBUG:creopyson.connection:request: {'sessionId': '', 'command': 'connection',
↪'function': 'connect', 'data': None}
DEBUG:creopyson.connection:response: {'status': {'error': False}, 'sessionId': '-
↪8685569143476874454'}
DEBUG:creopyson.connection:request: {'sessionId': '-8685569143476874454', 'command':
↪'file', 'function': 'open', 'data': {'display': True, 'activate': True, 'file': 'my_
↪file.prt'}}
DEBUG:creopyson.connection:response: {'status': {'error': False}, 'data': {'revision
↪': 1, 'files': ['MY_FILE.prt'], 'dirname': 'C:/your/working/path/'}}
```

CHAPTER 4

creopyson

4.1 creopyson package

4.1.1 Submodules

4.1.2 creopyson.bom module

Bom module.

creopyson.bom.**get_paths**(*client*, *file_=None*, *paths=None*, *skeletons=None*, *top_level=None*, *get_transforms=None*, *exclude_inactive=None*, *get_simpreps=None*)

Get a hierarchy of components within an assembly.

Even if you do not set exclude_inactive to true, the function will still exclude any components with a status of INACTIVE or UNREGENERATED.

Args:

client (obj): creopyson Client.

***file_* (string, optional):** file name, if not set, active model is used.

paths (boolean, optional): Whether to return component paths for each component (default" : False)

skeletons (boolean, optional): Whether to include skeleton components (default" : False)

top_level (boolean, optional): Whether to return only the top-level components in the assembly. (default" : False)

get_transforms (boolean, optional): Whether to return the 3D transform matrix for each component. (default" : False)

exclude_inactive (boolean, optional): Whether to exclude components which do not have an ACTIVE status. (default" : False)

get_simpreps (boolean, optionnal): Whether to return the Simplified Rep data for each component. (default" : False)

Returns:

Dict:

file (string): Assembly file name

generic (string): Generic name for the assembly

children (object:BomChild): The hierarchy of component data, starting with the top-level assembly.

has_simprep (boolean): Whether the assembly has a Simplified Rep.

in *children* there is the *seq_path* which indicates the children level, ex: *root.3.2*.

4.1.3 creopyson.connection module

Connection module.

class creopyson.connection.**Client**(*ip_adress='localhost'*, *port=9056*)
Bases: object

Creates Client object.

bom_get_paths(*file_=None*, *paths=None*, *skeletons=None*, *top_level=None*, *get_transforms=None*, *exclude_inactive=None*, *get_simpreps=None*)
Get a hierarchy of components within an assembly.

Even if you do not set exclude_inactive to true, the function will still exclude any components with a status of INACTIVE or UNREGENERATED.

Args:

client (obj): creopyson Client.

***file_* (string, optional):** file name, if not set, active model is used.

paths (boolean, optional): Whether to return component paths for each component (default" : False)

skeletons (boolean, optional): Whether to include skeleton components (default" : False)

top_level (boolean, optional): Whether to return only the top-level components in the assembly. (default" : False)

get_transforms (boolean, optional): Whether to return the 3D transform matrix for each component. (default" : False)

exclude_inactive (boolean, optional): Whether to exclude components which do not have an ACTIVE status. (default" : False)

get_simpreps (boolean, optionnal): Whether to return the Simplified Rep data for each component. (default" : False)

Returns:

Dict:

file (string): Assembly file name

generic (string): Generic name for the assembly

children (object:BomChild): The hierarchy of component data, starting with the top-level assembly.

has_simprep (boolean): Whether the assembly has a Simplified Rep.

in *children* there is the *seq_path* which indicates the children level, ex: *root.3.2*.

connect()
Connect to CREOSON.

Define 'sessionId'. Exit if server not found.

creo_cd(*dirname*)
Change Creo's working directory.

You can use absolute path or relative: "C:\My Workdir" "..\Other_directory"

Args: client (obj): creopyson Client. dirname (str): New directory name.

Returns: (str): Name of new working directory.

creo_delete_files(*filename=None*, *dirname=None*)
Delete files from a directory working directory.

Args:

client (obj): creopyson Client.

filename (str or list:str, optional): File name filter or list of file names. if blank all files will be deleted. (wildcards_allowed: True). Defaults to None.

dirname (str, optional): Directory name. Defaults is Creo's current working directory.

Returns: (list:str): List of deleted files.

creo_get_config(*name*)
Get the value of a Creo config option.

Args: client (obj): creopyson Client. name (str): Option name.

Raises: Warning: error message from creoson.

Returns:

(list:str): List of option values (some options can have multiple values).

creo_get_std_color(*color_type*)
Get one of Creo's standard colors.

Args:

client (obj): creopyson Client.

color_type (str): Color type. Valid values: letter, highlight, drawing, background, half_tone, edge_highlight, dimmed, error, warning, sheetmetal, curve, presel_highlight, selected, secondary_selected, preview, secondary_preview, datum, quilt.

Returns:

(dict): red (int): Red value (0-255) green (int): Green value (0-255) blue (int): Blue value (0-255)

creo_list_dirs(*dirname=None*)
List subdirectories of Creo's current working directory.

Args:

client (obj): creoopyson Client.

dirname (str, optional): Directory name filter (wildcards_allowed: True). Defaults: All subdirectories are listed.

Returns: (list:str): List of subdirectories

creo_list_files(*filename=None*)
List files in Creo's current working directory.

Args:

client (obj): creopyson Client.

filename (str, optional): File name filter (wildcards_allowed: True). Defaults: all files are listed.

Returns: (list:int): List of files.

creo_mkdir(*dirname*)
Create a new directory.

Args: client (obj): creopyson Client. dirname (str): New directory name.

Returns: (str): Full name of new working directory.

creo_pwd()
Return Creo's current working directory.

Args: client (obj): creopyson Client.

Returns: (str): Full name of working directory.

creo_rmdir(*dirname*)
Delete a directory.

Args: client (obj): creopyson Client. dirname (str): Directory name to delete.

Returns: None.

creo_set_config(*name*, *value*, *ignore_errors=None*)
Set a Creo config option.

Args:

client (obj): creopyson Client.

name (str): Option name.

value (str): New option value.

ignore_errors (boolean, optional): Whether to ignore errors that might occur when setting the config option. Defaults is False.

Returns: None.

creo_set_creo_version(*version*)
Set the version of Creo you are running.

This function only needs to be called once per creoson session. This function must be called if you are doing certain functions in Creo 7 or later due to deprecated config options. At this time this function only supports 7, 8 and 9..

This is needed for functions: familytable_replace file_assemble file_regenerate feature_delete feature_resume feature_suppress

Args:

client (obj): creopyson Client.

version (int): Creo version.

Returns: None.

creo_set_std_color(*color_type*, *red*, *green*, *blue*)
Set one of Creo's standard colors.

Args:

client (obj): creopyson Client.

color_type (str): Color type. Valid values: letter, highlight, drawing, background, half_tone, edge_highlight, dimmed, error, warning, sheetmetal, curve, presel_highlight, selected, secondary_selected, preview, secondary_preview, datum, quilt.

red (int): Red value (0-255).

green (int): Green value (0-255).

blue (int): Blue value (0-255).

Returns: None.

dimension_copy(*name*, *to_name*, *file_=None*, *to_file=None*)
Copy dimension to another in the same model or another model.

Args:

client (obj): creopyson Client.

name (str): Dimension name to copy.

to_name (str): Destination dimension; th dimension must already exist.

***file_* (str, optional):** Model name. Defaults is current active model.

to_file (str, optional): Destination model. Defaults is the source model.

Returns: None

dimension_list(*name=None*, *file_=None*, *dim_type=None*, *encoded=None*, *select=False*)
Get a list of dimensions from a model.

If select is true, then the current selection in Creo will be cleared even if no items are found.

Args:

client (obj): creopyson Client.

name (str|list:str, optional): Dimension name; if empty then all dimensions are listed.

***file_* (str, optional):** Model name. Defaults is current active model.

dim_type (str, optional): Dimension type filter. Defaults is *no filter*. Valid values: linear, radial, diameter, angular.

encoded (boolean, optional): Whether to return the values Base64-encoded. Defaults is False.

select (boolean, optional): If true, the dimensions that are found will be selected in Creo. Defaults is False.

Returns:

(list:dict): List of dimension information.

name (str): Dimension name

value (str|float): Dimension value; if encoded is True it is a str, if encoded is False it is a float.

encoded (boolean): Whether the returned value is Base64-encoded.

dwg_dim (boolean): Whether dimension is a drawing dimension rather than a model dimension.

dimension_list_detail(*name=None*, *file_=None*, *dim_type=None*, *encoded=None*, *select=False*)
Get a list of dimension details from a model.

Values will automatically be returned Base64-encoded if they are strings which contain Creo Symbols or other non-ASCII data. If select is true, then the current selection in Creo will be cleared even if no items are found.

Args:

client (obj): creopyson Client.

name (str|list:str, optional): Dimension name; if empty then all dimensions are listed.

***file_* (str, optional):** Model name. Defaults is current active model.

dim_type (str, optional): Dimension type filter. Defaults is *no filter*. Valid values: linear, radial, diameter, angular.

encoded (boolean, optional): Whether to return the values Base64-encoded. Defaults is False.

select (boolean, optional): If true, the dimensions that are found will be selected in Creo. Defaults is False.

Returns:

(list:dict): List of dimension information.

name (str): Dimension name

value (str|float): Dimension value; if encoded is True it is a str, if encoded is False it is a float.

encoded (boolean): Whether the returned value is Base64-encoded.

sheet (int): Sheet number.

view_name (str): View name.

dim_type (str): Dimension type. Valid values: linear, radial, diameter, angular.

dwg_dim (boolean): Whether dimension is a drawing dimension rather than a model dimension.

text (str): dimension text.

location (dict): Coordonates location. x (float): X coordonate location. y (float): Y coordonate location. z (float): Z coordonate location.

tolerance_type (str): Tolerance type, if not specified not returned. Valid values: plus_minus (TODO complete list).

tol_plus (float): Plus tolerance value. if tolerance_type not specified not returned.

tol_minus (float): Minus tolerance value. if tolerance_type not specified not returned.

dimension_set (*name*, *value*, *file_=None*, *encoded=None*)

Set a dimension value.

One reason to encode values is if the value contains special characters, such as Creo symbols. You may be able to avoid Base64-encoding symbols by using Unicode for the binary characters, for example including \u0001#\u0002 in the value to insert a plus/minus symbol.

Args:

client (obj): creopyson Client.

name (str): Dimension name.

value (str|float): Dimension value.

***file_* (string, optional):** file name, if not set, active model is used.

encoded (boolean, optional): Whether the value is Base64-encoded. Defaults is False.

Raises: Warning: error message from creoson.

Returns: None

dimension_set_text (*name*, *file_=None*, *text=None*, *encoded=False*)
Set dimension text.

Args:

client (obj): creopyson object.

name (str): Dimension name.

***file_* (string, optional):** file name, if not set, active model is used.

text ([type], optional): Dimension text. Defaults to None, sets the dimension's text to @D.

encoded (bool, optional): Whether the text value is Base64-encoded. Defaults to False.

Returns: None

dimension_show (*name*, *file_=None*, *assembly=None*, *path=None*)
Display or hide a dimension in Creo.

Args:

client (obj): creopyson Client.

name (str): Dimension name.

***file_* (str, optional):** Model name. Defaults is current active model.

assembly (str, optional): Assembly name; only used if path is given. Defaults is the currently active model.

path (list:int, optional): Path to occurrence of the model within the assembly; the dimension will only be shown for that occurrence. Defaults: all occurrences of the component are affected.

Returns: None

dimension_user_select (*file_=None*, *maxi=None*)
Prompt user to select one or more dimensions, and return their selections.

client (obj): creopyson Client.

***file_* (str, optional):** Model name. Defaults is current active model.

maxi (int, optional): The maximum number of dimensions that the user can select. Defaults is *1*.

Raises: Warning: error message from creoson.

Returns:

(list:dict): List of selected dimension information

name (str): Dimension name

value (str|float): Dimension value; if encoded is True it is a str, if encoded is False it is a float.

encoded (boolean): Whether the returned value is Base64-encoded.

file (str): File name.

relation_id (int): Relation ID number.

disconnect()
Disconnect from CREOSON.

Empty sessionId.

drawing_add_model(*model*, *drawing=None*)
Add a model to a drawing.

Args:

client (obj): creopyson Client.

model (str): Model name.

drawing (str, optional): Drawing name. Defaults is Current active drawing.

Returns: None

drawing_add_sheet(*position=None*, *drawing=None*)
Add a drawing sheet.

Args:

client (obj): creopyson Client.

position (int, optional): Position to add the sheet. Defaults: Sheet will be added to the end.

drawing (str, optional): Drawing name. Defaults is current active drawing.

Returns: None

drawing_create(*template*, *model=None*, *drawing=None*, *scale=None*, *display=None*, *activate=None*, *new_window=None*)
Create a new drawing from a template.

Args:

client (obj): creopyson Client.

template (str): Template

model (str, optional): Model name. Defaults: Current active model.

drawing (str, optional): New drawing name. Defaults: A name derived from the model's instance name.

scale (float, optional): Drawing scale. Defaults is *1.0*.

display (boolean, optional): Display the drawing after open. Defaults is False.

activate (boolean, optional): Activate the drawing window after open. Defaults is False.

new_window (boolean, optional): Open drawing in a new window. Defaults is False.

Returns: (str): New drawing name.

drawing_create_gen_view(*model_view*, *point*, *drawing=None*, *view=None*, *sheet=None*, *model=None*, *scale=None*, *display_data=None*, *exploded=None*)
Create general view on a drawing.

Args:

client (obj): creopyson Client

model_view (str): Model view to use for the drawing view orientation.

point (dict): Coordinates for the view in Drawing Units.

drawing (str, optional): Drawing name. Defaults: current active drawing.

view (str, optional): New view name. Defaults: the model_view parameter.

sheet (int, optional): Sheet number. Defaults: current active sheet on the drawing.

model (str, optional): Model for the view. Defaults: current active model on the drawing.

scale (float, optional): View scale. Defaults: the sheet's scale.

display_data (dict, optional): Display parameters used to create the view. Defaults: Creo defaults.

exploded (boolean, optional): Whether to create the view as an exploded view. Defaults is False.

Returns: None

drawing_create_proj_view(*parent_view*, *point*, *drawing=None*, *view=None*, *sheet=None*, *display_data=None*, *exploded=None*)

Create projection view on a drawing.

When specifying the view coordinates, you should specify only an X or a Y coordinate to avoid confusion. If you specify both coordinates, it appears Creo may be using whichever has the larger absolute value.

Args:

client (obj): creopyson Client

parent_view (str): Parent view for the projection view.

point (dict): Coordinates for the view, relative to the location of the parent view, in Drawing Units.

drawing (str, optional): Drawing name. Defaults: current active drawing.

view (str, optional): New view name. Defaults: Creo's default name for a new view.

sheet (int, optional): Sheet number. Defaults: current active sheet on the drawing.

display_data (dict, optional): Display parameters used to create the view. Defaults: the display parameters of the parent view.

exploded (boolean, optional): Whether to create the view as an exploded view. Defaults is False.

Returns: None

drawing_create_symbol(*symbol_file*, *point*, *drawing=None*, *replace_values=None*, *sheet=None*)

Add a symbol instance to a drawing.

Args:

client (obj): creopyson Client

symbol_file (str): Name of the symbol file.

point (dict): Coordinates for the symbol in Drawing Units.

drawing (str, optional): Drawing name. Defaults: current active drawing.

replace_values (dict, optional): Object containing replacement values for any variable text in the symbol. Defaults to None.

sheet (int, optional): Sheet number (0 for all sheets). Defaults: the symbol will be added to all sheets.

Returns: None

drawing_delete_models(*model=None*, *drawing=None*, *delete_views=None*)

Delete one or more models from a drawing.

Args:

client (obj): creopyson Client

model (str, optional): Model name (wildcard allowed: True). Defaults: all models will be deleted from the drawing.

drawing (str, optional): Drawing name. Defaults: current active drawing.

delete_views (boolean, optional): Whether to delete drawing views associated with the model. Defaults is False.

Returns: None

drawing_delete_sheet (*sheet*, *drawing=None*)
Delete a drawing sheet.

An error will occur if you try to delete the only sheet in a drawing.

Args:

client (obj): creopyson Client

sheet (int): Sheet number.

drawing (str, optional): Drawing name. Defaults: current active drawing.

Returns: None

drawing_delete_symbol_def (*symbol_file*, *drawing=None*)
Delete a symbol definition and its instances from a drawing.

Args:

client (obj): creopyson Client

symbol_file (str): Name of the symbol file.

drawing (str, optional): Drawing name. Defaults: current active drawing.

Returns: None

drawing_delete_symbol_inst (*symbol_id*, *drawing=None*)
Delete a specific symbol instance from a drawing.

Args:

client (obj): creopyson Client

symbol_id (str): ID of the symbol instance.

drawing (str, optional): Drawing name. Defaults: current active drawing.

Returns: None

drawing_delete_view (*view*, *drawing=None*, *sheet=None*, *del_children=None*)
Delete a drawing view.

Args:

client (obj): creopyson Client

view (str): View name.

drawing (str, optional): Drawing name. Defaults: current active drawing.

sheet (int, optional): Sheet number; if filled in, the view will only be deleted if it is on that sheet. Defaults: Delete the view from any sheet.

del_children ([boolean, optional): Whether to also delete any children of the view. Defaults is False.

Returns: None

drawing_get_cur_model (*drawing=None*)
Get the active model on a drawing.

Args:

client (obj): creopyson Client

drawing (str, optional): Drawing name. Defaults: current active drawing.

Returns: (str): Model name.

drawing_get_cur_sheet (*drawing=None*)
Get the current drawing sheet.

Args:

client (obj): creopyson Client

drawing (str, optional): Drawing name. Defaults: current active drawing.

Returns: (int): Sheet number.

drawing_get_num_sheets (*drawing=None*)
Get the number of sheets on a drawing.

Args:

client (obj): creopyson Client

drawing (str, optional): Drawing name. Defaults: current active drawing.

Returns: (int): Number of sheets.

drawing_get_sheet_format (*sheet*, *drawing=None*)
Get the drawing format file of drawing sheet.

Args:

client (obj): creopyson Client.

sheet (int): Sheet number.

drawing (str, optional): Drawing name. Defaults: current active drawing.

Returns:

(dict):

file(str): Format file name, may be null if there is no current format.

full_name(str): Format full name.

common_name(str): Format common name.

drawing_get_sheet_scale (*sheet*, *drawing=None*, *model=None*)
Get the scale of a drawing sheet.

Args:

client (obj): creopyson Client

sheet (int): Sheet number.

drawing (str, optional): Drawing name. Defaults: current active drawing.

model (str, optional): Drawing model used to calculate the scale. Defaults: the active model on the drawing.

Returns: (float): Sheet scale.

drawing_get_sheet_size(*sheet*, *drawing=None*)
Get the size of a drawing sheet.

Args:

client (obj): creopyson Client

sheet (int): Sheet number.

drawing (str, optional): Drawing name. Defaults: current active drawing.

Returns: (str): Sheet size.

drawing_get_view_loc(*view*, *drawing=None*)
Get the location of a drawing view.

Args:

client (obj): creopyson Client

view (str): View name.

drawing (str, optional): Drawing name. Defaults: current active drawing.

Returns:

(dict): x (float): X-coordinate of the view y (float): Y-coordinate of the view z (float): Z-coordinate of the view

drawing_get_view_scale(*view*, *drawing=None*)
Get the scale of a drawing view.

Args:

client (obj): creopyson Client

view (str): View name.

drawing (str, optional): Drawing name. Defaults: current active drawing.

Raises: Warning: error message from creoson.

Returns: (float): View scale.

drawing_get_view_sheet(*view*, *drawing=None*)
Get the sheet number that contains a drawing view.

Args:

client (obj): creopyson Client

view (str): View name.

drawing (str, optional): Drawing name. Defaults: current active drawing.

Returns: (int): Sheet number.

drawing_is_symbol_def_loaded(*symbol_file*, *drawing=None*)
Check whether a symbol definition file is loaded into Creo.

Args:

client (obj): creopyson Client

symbol_file (str): Name of the symbol file.

drawing (str, optional): Drawing name. Defaults: current active drawing.

Returns: (boolean): Whether the symbol definition is loaded into Creo.

drawing_list_models (*model=None*, *drawing=None*)
List the models contained in a drawing.

Args:

client (obj): creopyson Client

model (str, optional): Model name filter (wildcards allowed: True). Defaults: no filter.

drawing (str, optional): Drawing name. Defaults: current active drawing.

Returns: (list:str): List of model names in the drawing.

drawing_list_symbols (*drawing=None*, *symbol_file=None*, *sheet=None*)
List symbols contained on a drawing.

Args:

client (obj): creopyson Client

drawing (str, optional): Drawing name. Defaults: current active drawing.

symbol_file (str, optional): Symbol file name filter. Defaults: no filter.

sheet (int, optional): Sheet number (0 for all sheets). Defaults: The symbol will be added to all sheets.

Returns:

(list:dict):

List of symbols in the drawing. id (int): Symbol ID. symbol_name (str): Symbol name. sheet (int): Sheet number.

drawing_list_view_details (*view=None*, *drawing=None*)
List the views contained in a drawing, with more details.

Args:

client (obj): creopyson Client

view (str, optional): View name filter (wildcards allowed: True). Defaults: no filter.

drawing (str, optional): Drawing name. Defaults: current active drawing.

Returns:

(list:dict):

List of views in the drawing

name (str): View name.

sheet (int): Sheet number.

location (dict):

Coordonates x (float): X-coordinate of the view y (float): Y-coordinate of the view z (float): Z-coordinate of the view

text_height (float): Text Heigh in Drawing Units.

view_model (str): View model name.

simp_rep (str): View simplified rep name.

drawing_list_views (*view=None*, *drawing=None*)
List the views contained in a drawing.

Args:

client (obj): creopyson Client

view (str, optional): View name filter (wildcards allowed: True). Defaults: no filter.

drawing (str, optional): Drawing name. Defaults: current active drawing.

Returns: (list:str): List of views in the drawing.

drawing_load_symbol_def (*symbol_file*, *symbol_dir=None*, *drawing=None*)
Load a Creo symbol definition file into Creo from disk.

Args:

client (obj): creopyson Client

symbol_file (str): Name of the symbol file.

symbol_dir (str, optional): Directory containing the symbol file; if relative, assumed to be relative to Creo's current working directory. Defaults: Creo's current working directory.

drawing (str, optional): Drawing name. Defaults: current active drawing.

Returns:

(dict): Symbol definition. id (int): ID of the loaded symbol. name (str): Symbol Name of the loaded symbol.

drawing_regenerate (*drawing=None*)
Regenerate a drawing.

Args:

client (obj): creopyson Client

drawing (str, optional): Drawing name. Defaults: current active drawing.

Returns: None

drawing_regenerate_sheet (*sheet=None*, *drawing=None*)
Regenerate a sheet on a drawing.

Args:

client (obj): creopyson Client

sheet (int, optional): Sheet number (0 for all sheets). Defaults: all sheets will be regenerated.

drawing (str, optional): Drawing name. Defaults: current active drawing.

Returns: None

drawing_rename_view (*view*, *new_view*, *drawing=None*)
Rename a drawing view.

Args:

client (obj): creopyson Client

view (str): Old view name.

new_view (str): New view name.

drawing (str, optional): Drawing name. Defaults: current active drawing.

Returns: None

drawing_scale_sheet(*sheet*, *scale*, *drawing=None*, *model=None*)
Set the scale of a drawing sheet.

Args:

client (obj): creopyson Client

sheet (int): Sheet number.

scale (float): View scale.

drawing (str, optional): Drawing name. Defaults: current active drawing.

model (str, optional): Drawing model to scale. Defaults: tThe active model on the drawing.

Returns: None

drawing_scale_view(*scale*, *view=None*, *drawing=None*)
Set the scale of one or more drawing views.

Args:

client (obj): creopyson Client

scale (float): View scale.

view (str, optional): View name (wildcards allowed: True). Defaults: all views will be scaled.

drawing (str, optional): Drawing name. Defaults: current active drawing.

Returns:

(dict)

succes_views (list): List of view which were successfully scaled.

failed_views (list): List of view which failed to scale.

drawing_select_sheet(*sheet*, *drawing=None*)
Make a drawing sheet the current sheet.

Args:

client (obj): creopyson Client

sheet (int): Sheet number.

drawing (str, optional): Drawing name. Defaults: current active drawing.

Returns: None

drawing_set_cur_model(*model*, *drawing=None*)
Set the active model on a drawing.

Args:

client (obj): creopyson Client

model (str): Model name.

drawing (str, optional): Drawing name. Defaults: current active drawing.

Returns: None

drawing_set_sheet_format(*sheet*, *file_format*, *dirname=None*, *drawing=None*)
Set the drawing format file of a drawing sheet.

Args:

client (obj): creopyson Client.

sheet (int): Sheet number.

file_format (str): Format file name.

dirname (str, optional): Directory name containing the file format. Defaults to None is current working directory.

drawing (str, optional): Drawing name. Defaults to None is current active drawing.

Returns: None

drawing_set_view_loc(*view*, *point*, *drawing=None*)
Set the location of a drawing view.

Args:

client (obj): creopyson Client

view (str): View name.

point (dict): Coordinates for the view in Drawing Units

drawing (str, optional): Drawing name. Defaults: current active drawing.

Returns: None

drawing_view_bound_box(*view*, *drawing=None*)
Get the 2D bounding box for a drawing view.

Args:

client (obj): creopyson Client

view (str): View name.

drawing (str, optional): Drawing name. Defaults: current active drawing.

Returns:

(dict): xmin (float): Minimum X-coordinate of drawing view. xmax (float): Maximum X-coordinate of drawing view. ymin (float): Minimum Y-coordinate of drawing view. ymax (float): Maximum Y-coordinate of drawing view.

familytable_add_inst(*instance*, *file_=None*)
Add a new instance to a family table.

Creates a family table if one does not exist.

Args:

client (obj): creopyson Client.

instance (str): New instance name.

***file_* (str, optional):** File name. Defaults is currently active model.

Returns: None

familytable_create_inst(*instance*, *file_=None*)
Create a new model from a family table row.

Args:

client (obj): creopyson Client.

instance (str): Instance name.

***file_* (str, optional):** File name. Defaults is currently active model.

Returns: (str): New model name.

familytable_delete(*file_=None*)
Delete an entire family table.

Args:

client (obj): creopyson Client.

***file_* (str, optional):** File name (wildcards allowed: True). Defaults is currently active model.

Returns: None

familytable_delete_inst(*instance*, *file_=None*)
Delete an instance from a family table.

Args:

client (obj): creopyson Client.

instance (str): Instance name.

***file_* (str, optional):** File name. Defaults is currently active model.

Returns: None

familytable_exists(*instance*, *file_=None*)
Check whether an instance exists in a family table.

Args:

client (obj): creopyson Client.

instance (str): Instance name.

***file_* (str, optional):** File name. Defaults is currently active model.

Returns:

(boolean): Whether the instance exists in the model's family table; returns false if there is no family table in the model.

familytable_get_cell(*instance*, *colid*, *file_=None*)
Get one cell of a family table.

Args:

client (obj): creopyson Client.

instance (str): Instance name.

colid (str): Colimn ID.

***file_* (str, optional):** File name. Defaults is currently active model.

Returns:

(dict):

instance (str): Family Table instance name.

colid (str): Column ID.

value (depends on datatype): Cell value.

datatype (str): Data type.

coltype (str): Column Type; a string corresponding to the Creo column type.

familytable_get_header (*file_=None*)
Get the header of a family table.

Args:

client (obj): creopyson Client.

***file_* (str, optional):** File name. Defaults is currently active model.

Returns:

(list:dict):

colid (str): Column ID.

value (depends on date type): Cell value.

datatype (str): Data type. Valid values: STRING, DOUBLE, INTEGER, BOOL, NOTE.

coltype (str): Column Type; a string corresponding to the Creo column type.

familytable_get_parents (*file_=None*)
Get the parent instances of a model in a nested family table.

Args:

client (obj): creopyson Client.

***file_* (str, optional):** File name. Defaults is currently active model.

Returns:

(list:str): List of parent instance names, starting with the immediate parent and working back.

familytable_get_row (*instance*, *file_=None*)
Get one row of a family table.

Args:

client (obj): creopyson Client.

instance (str): Instance name.

***file_* (str, optional):** File name. Defaults is currently active model.

Returns:

(dict):

colid (str): Column ID.

value (depends on datatype): Cell value.

datatype (str): Data type.

coltype (str): Column Type; a string corresponding to the Creo column type.

familytable_list (*file_=None*, *instance=None*)
List the instance names in a family table.

Args:

client (obj): creopyson Client.

instance (str, optional): Instance name filter (wildcards allowed: True). Defaults is all instances.

***file_* (str, optional):** File name. Defaults is currently active model.

Returns: (list:str): List of matching instance names

familytable_list_tree (*file_=None*, *erase=None*)
Get a hierarchical structure of a nested family table.

Args:

client (obj): creopyson Client.

***file_* (str, optional):** File name. Defaults is currently active model.

erase (boolean, optional): Erase model and non-displayed models afterwards. Defaults is *False*.

Returns:

(list:str):

List of child instances

total (int): Count of all child instances including their decendants.

children (list:dict):

name (str): Name of the family table instance.

total (int): Count of all child instances including their decendants.

children (list:dict): List of child instances.

familytable_replace (*cur_model*, *new_inst*, *file_=None*, *cur_inst=None*, *path=None*)
Replace a model in an assembly with another inst in the same family table.

You must specify either cur_inst or path.

Args:

client (obj): creopyson Client.

cur_model (str): Generic model containing the instances.

new_inst (str): New instance name.

***file_* (str, optional):** File name (usually an assembly). Defaults is currently active model.

cur_inst (str): Instance name to replace. Defaults to None.

path (list:int, optional): Path to component to replace. Defaults to None.

Returns: None

familytable_set_cell (*instance*, *colid*, *value*, *file_=None*)
Set the value of one cell of a family table.

Args:

client (obj): creopyson Client.

instance (str): Family Table instance name.

colid (str): Column ID.

value (depends on data type): Cell value.

***file_* (str, optional):** File name (usually an assembly). Defaults is currently active model.

Returns: None

feature_delete (*name=None*, *file_=None*, *status=None*, *type_=None*, *clip=None*)
Delete one or more features that match criteria.

Args:

client (obj): creopyson Client.

name (str|list:str, optional): Dimension name, (wildcards allowed: True); if empty then all features are listed.

***file_* (str, optional):** Model name (wildcards allowed: True). Defaults is current active model.

status (str, optional): Feature status pattern (wildcards allowed: True). Defaults: All feature statuses. Valid values: ACTIVE, INACTIVE, FAMILY_TABLE_SUPPRESSED, SIMP_REP_SUPPRESSED, PROGRAM_SUPPRESSED, SUPPRESSED, UNREGENERATED

***type_* (str, optional):** Feature type pattern (wildcards allowed: True). Defaults: All feature types.

clip (boolean, optional): Whether to clip-delete ANY features from this feature through the end of the structure. Defaults is False.

Raises: ValueError: status value is incorrect.

Returns: None

feature_delete_param(*name=None*, *file_=None*, *param=None*)
Delete a feature parameter.

Args:

client (obj): creopyson Client.

name (str, optional): Parameter name (wildcards allowed: True). Defaults: All parameter names.

***file_* (str, optional):** Model name. Defaults is current active model.

param (str, optional): Parameter name (wildcards allowed: True). Defaults: All parameter names.

Returns: None

feature_list(*file_=None*, *name=None*, *status=None*, *type_=None*, *paths=None*, *no_datum=None*, *inc_unnamed=None*, *no_comp=None*)
List feature parameters that match criteria.

Will only list parameters on visible features.

Args:

client (obj): creopyson Client.

***file_* (str, optional):** File name. Defaults is the currently active model.

name (str, optional): Feature name (wildcards allowed: True). Defaults: All features are listed.

status (str, optionnal): Feature status pattern. Defaults: All feature statuses. Valid values: ACTIVE, INACTIVE, FAMILY_TABLE_SUPPRESSED, SIMP_REP_SUPPRESSED, PROGRAM_SUPPRESSED, SUPPRESSED, UNREGENERATED.

***type_* (str, optional):** Feature type patter (wildcards allowed: True). Defaults: All feature types.

paths (boolean, optionnal): Whether feature ID and feature number are returned with the data Default: False.

no_datum (boolean, optional): Whether to exclude datum-type features from the list; these are COORD_SYS, CURVE, DATUM_AXIS, DATUM_PLANE, DATUM_POINT, DATUM_QUILT, and DATUM_SURFACE features. Defaults is False.

inc_unnamed (boolean, optional): Whether to include unnamed features in the list. Defaults is False.

no_comp (boolean, optional): Whether to include component-type features in the list. Defaults is False.

Raises: ValueError: status value is incorrect.

Returns:

(list:dict): List of parameter information.

name (str): Parameter nam.

value (depends on data type): Parameter value.

type (string): Data type. Valid values: STRING, DOUBLE, INTEGER, BOOL, NOTE.

designate (boolean): Value is designated.

encoded (boolean): Value is Base64-encoded.

owner_name (str): Owner Name.

owner_id (int): Owner ID.

owner_type (str): Owner type.

feature_list_group_features (*group_name*, *type_=None*, *file_=None*)

List features in a Creo Group.

Args:

client (obj): creopyson Client.

group_name (str): Group name.

***type_* (str, optional):** Feature type patter (wildcards allowed: True). Defaults: All feature types.

***file_* (str, optional):** File name. Defaults is the currently active model.

Returns: (list:dict): List of feature information

feature_list_params (*file_=None*, *name=None*, *type_=None*, *no_datum=None*, *inc_unnamed=None*, *no_comp=None*, *param=None*, *value=None*, *encoded=None*)

List feature parameters that match criteria.

Args:

client (obj): creopyson Client.

***file_* (str, optional):** File name. Defaults is the currently active model.

name (str|int, optional): str: Feature name (wildcards allowed: True). int: Feature ID. Defaults: All features are listed.

***type_* (str, optional):** Feature type patter (wildcards allowed: True). Defaults: All feature types.

no_datum (boolean, optional): Whether to exclude datum-type features from the list; these are COORD_SYS, CURVE, DATUM_AXIS, DATUM_PLANE, DATUM_POINT, DATUM_QUILT, and DATUM_SURFACE features. Defaults is False.

inc_unnamed (boolean, optional): Whether to include unnamed features in the list. Defaults is False.

no_comp (boolean, optional): Whether to include component-type features in the list. Defaults is False.

param (str|list:str, optional): Parameter name; (wildcards allowed: True) if empty all parameters are listed.

value (str, optional): Parameter value filter (wildcards allowed: True). Defaults is no filter.

encoded (boolean, optional): Whether to return the values Base64-encoded. Defaults is False.

Returns:

(list:dict): List of parameter information.

name (str): Parameter nam.

value (depends on data type): Parameter value.

type (string): Data type. Valid values: STRING, DOUBLE, INTEGER, BOOL, NOTE.

designate (boolean): Value is designated.

encoded (boolean): Value is Base64-encoded.

owner_name (str): Owner Name.

owner_id (int): Owner ID.

owner_type (str): Owner type.

description (str): List of parameter information.

feature_list_pattern_features(*patter_name*, *type_=None*, *file_=None*)

List features in a Creo Pattern.

Args:

client (obj): creopyson Client.

patter_name (str): Pattern name.

***type_* (str, optional):** Feature type patter (wildcards allowed: True). Defaults: All feature types.

***file_* (str, optional):** File name. Defaults is the currently active model.

Returns: (list:dict): List of feature information

feature_list_selected()

List the currently selected features in Creo

Returns:

(list): List of feature informations.

```
[
    { 'file' : model name (str)
      'name' : feature name (str)
      'status' : feature status (str)
      'type' : feature type (str)
      'feat_id' : feature ID (int)
      'feat_number' : feature number (int)
      'path' : feature's component path (list of ints)
    },
]
```

feature_param_exists(*file_=None*, *name=None*, *param=None*)

Check whether parameter(s) exists on a feature.

Args:

client (obj): creopyson Client.

***file_* (str, optional):** File name. Defaults is the currently active model.

name (str, optional): Parameter name (wildcards allowed: True). Defaults: All parameter names.

param (str|list:str, optional): Parameter name; (wildcards allowed: True) if empty all parameters are listed.

Returns: (boolean): Whether the parameter exists on the model

feature_rename(*name*, *new_name*, *file_=None*)
Rename a feature.

Args:

client (obj): creopyson Client.

name (str|int, optional): Feature name (str) or Feature ID (int).

new_name (str): New name for the feature.

***file_* (str, optional):** File name. Defaults is the currently active model.

Returns: None

feature_resume(*file_=None*, *name=None*, *status=None*, *type_=None*, *with_children=None*)
Resume one or more features that match criteria.

Will only resume visible features.

Args:

client (obj): creopyson Client.

***file_* (str, optional):** File name (wildcards allowed: True). Defaults is the currently active model.

name (int|str|list:str, optional): Feature name or Feature ID, (wildcards allowed: True); if empty then all features are resumed. int => Feat_ID str => name list:str => names

status (str, optional): Feature status pattern. Defaults: All feature statuses. Valid values: ACTIVE, INACTIVE, FAMILY_TABLE_SUPPRESSED, SIMP_REP_SUPPRESSED, PROGRAM_SUPPRESSED, SUPPRESSED, UNREGENERATED

***type_* (str, optional):** Feature type pattern (wildcards allowed: True). Defaults: All feature types.

with_children (boolean, optional): Whether to resume any child features of the resumed feature. Defaults is False.

Raises: ValueError: status value is incorrect.

Returns: None

feature_set_param(*param*, *file_=None*, *name=None*, *type_=None*, *value=None*, *encoded=None*, *designate=None*, *description=None*, *no_create=None*)
Set the value of a feature parameter.

Will only set parameters on visible features.

Args:

client (obj): creopyson Client.

param (str): Parameter name.

***file_* (str, optional):** File name (wildcards allowed: True). Defaults is the currently active model.

name (str, optional): Feature name. Defaults: All features are updated.

***type_* (str, optional):** Parameter data type. Defaults is True. Valid values: STRING, DOUBLE, INTEGER, BOOL, NOTE.

value (depends on data type, optional): Parameter value. Defaults: Clears the parameter value if missing.

encoded (boolean, optional): Value is Base64-encoded. Defaults is False.

designate (boolean, optional): Set parameter to be designated/not designated, blank=do not set. Defaults is *blank*.

description (str, optionnal): Parameter description. If missing, leaves the currect description in place.

no_create (boolean, optional): If parameter does not already exist, do not create it. Defaults is False.

Returns: None

feature_suppress (*file_=None*, *name=None*, *status=None*, *type_=None*, *clip=None*, *with_children=None*)

Suppress one or more features that match criteria.

Will only suppress visible features.

Args:

client (obj): creopyson Client.

***file_* (str, optional):** File name (wildcards allowed: True). Defaults is the currently active model.

name (int|str|list:str, optional): Feature name or Feature ID, (wildcards allowed: True); if empty then all features are suppressed. int => Feat_ID str => name list:str => names

status (str, optional): Feature status pattern. Defaults: All feature statuses. Valid values: ACTIVE, INACTIVE, FAMILY_TABLE_SUPPRESSED, SIMP_REP_SUPPRESSED, PROGRAM_SUPPRESSED, SUPPRESSED, UNREGENERATED

***type_* (str, optional):** Feature type pattern (wildcards allowed: True). Defaults: All feature types.

clip (boolean, optional): Whether to clip-suppress ANY features from this feature through the end of the structure. Defaults is True.

with_children (boolean, optional): Whether to suppress any child features of the suppressed feature. Defaults is True.

Raises: ValueError: status value is incorrect.

Returns: None

feature_user_select_csys (*file_=None*, *max_=None*)

Prompt the user to select one or more coordinate systems.

and return their selections.

Args:

client (obj): creopyson Client.

***file_* (str, optional):** File name. Defaults is the currently active model.

***max_* (int, optional):** The maximum number of dimensions that the user can select. Defaults is *1*.

Returns:

(list:dict):

List of feature information.

name (str): Feature name.

type (string): Feature type.

status (str): Feature status. Valid values: ACTIVE, INACTIVE, FAMILY_TABLE_SUPPRESSED, SIMP_REP_SUPPRESSED, PROGRAM_SUPPRESSED, SUPPRESSED, UNREGENERATED.

feat_id (int): Feature ID.

file (str): File name containing the feature.

path (list:int): Component Path to feature (optionnal)

file_assemble(*file_*, *dirname=None*, *generic=None*, *into_asm=None*, *path=None*, *ref_model=None*, *transform=None*, *constraints=None*, *package_assembly=None*, *walk_children=None*, *assemble_to_root=None*, *suppress=None*)

Assemble a component into an assembly.

Args:

client (obj): creopyson Client.

***file_* (str):** File name component.

dirname (str, optional): Diretory name. Defaults is Creo's current working directory.

generic (str, optional): Generic model name (if file name represents an instance). Defaults is generic model name (if file name represents an instance).

into_asm (str, optional): Target assembly. Defaults is currently active model.

path (list:int, optional): Path to a component that the new part will be constrained to. Defaults to None.

ref_model (str, optional): Reference model that the new part will be constrained to; only used if path is not given. If there are multiple of this model in the assembly, the component will be assembled multiple times, once to each occurrence. Defaults to None.

transform (obj:JLTransform, optional): Transform structure for the initial position and orientation of the new component; only used if there are no constraints, or for certain constraint types. Defaults to None.

constraints (obj_array:JLConstraint, optional): Assembly constraints. Defaults to None.

package_assembly (bool, optional): Whether to package the component to the assembly; only used if there are no constraints specified. Defaults is If there are no constraints, then the user will be prompted to constrain the component through the Creo user interface.

walk_children (bool, optional): Whether to walk into subassemblies to find reference models to constrain to. Defaults to None.

assemble_to_root (bool, optional): Whether to always assemble to the root assembly, or assemble to the subassembly containing the reference path/model. Defaults to None.

suppress (bool, optional): Whether to suppress the components immediately after assembling them. Defaults to None.

Returns:

(dict):

dirname (str): Directory name of component.

files (list:str): File name of component.

revision (int): Revision of file that was opened; if more than one file was opened, this field is not returned.

featureid (int): Last Feature ID of component after assembly.

file_backup(*target_dir*, *file_=None*)
Backup a model.

Args:

client (obj): creopyson Client.

target_dir (str): Target directory name.

***file_* (str, optional):** File name. Defaults is currently active model.

Returns: None

file_close_window(*file_=None*)
Close the window containing a model.

Args:

client (obj): creopyson object.

***file_* (str, optional):** File name. Defaults is currently active model.

Returns: None

file_delete_material(*material*, *file_=None*)
Delete a material from a part.

Args:

client (obj): creopyson object

material (str): Material name.

***file_* (str, optional):** File name. (Wildcards allowed: True). Defaults is currently active model.

file_display(*file_*, *activate=None*)
Display a model in a window.

Args:

client (obj): creopyson object.

***file_* (str):** File name

activate (bool, optional): Activate the model after displaying. Defaults is True.

Returns: None

file_erase(*file_=None*, *erase_children=None*)
Erase one or more models from memory.

Args:

client (obj): creopyson Client.

***file_* (str|list:str, optional):** File name or List of file names; (Wildcards allowed: True). if empty all models in memory are erased.

erase_children (bool, optional): Erase children of the models too. Defaults is False.

Returns: None

file_erase_not_displayed()
Erase all non-displayed models from memory.

Args: client (obj): creopyson Client.

Returns: None

file_exists (*file_*)
Check whether a model exists in memory.

Args: client (obj): creopyson Client. *file_* (str): File name.

Returns: (bool): Whether the file is open in Creo.

file_get_accuracy (*file_=None*)
Get a solid's accuracy.

If the model has no accuracy value, this function will return null.

Args:

client (obj): creopyson Client.

***file_* (str, optional):** File name. Defaults is current active model

Returns:

(list):

accuracy (float): Accuracy value.

relative (bool): True = relative; False = absolute accuracy.

file_get_active ()
Get the active model from Creo.

Args: client (obj): creopyson Client.

Returns:

(dict): dirname (str): Directory name of current model. file (str): File name of current model.

file_get_cur_material (*file_=None*)
Get the current material for a part.

Note: This is the same as 'get_cur_material_wildcard' but this function does not allow wildcards on the part name. They are separate functions because the return structures are different. This function is retained for backwards compatibility.

Args:

client (obj): creopyson Client.

***file_* (str, optional):** Part name. Defaults to None is current active model.

Returns:

str: Current material for the part, may be null if there is no current material.

file_get_cur_material_wildcard (*file_=None*, *include_non_matching_parts=False*)
Get the current material for a part or parts.

Note: This is the same as 'get_cur_material' but this function allows wildcards on the part name. They are separate functions because the return structures are different.

Args:

client (obj): creopyson Client.

***file_* (str, optional):** Part name. Defaults to None is current active model.

include_non_matching_parts (bool, optionnal): Whether to include parts that match the part name pattern but don't have a current material. Defaults to False.

Returns:

list: A list of part and current-material pairs.

file_get_fileinfo (*file_=None*)
Open one or more files in memory or from the drive.

Args:

client (obj): creopyson Client.

***file_* (str, optional):** File name. Defaults is currently active model.

Returns:

(dict):

dirname (str): Directory name of the file.

file (str): File name.

revision (int): Revision number of file. (Not availlable if the file is in Windchill)

file_get_length_units (*file_=None*)
Get the current length units for a model.

Args:

client (obj): creopyson Client.

***file_* (str, optional):** File name. Defaults is currently active model.

Returns: (str): Length units.

file_get_mass_units (*file_=None*)
Get the current mass units for a model.

Args:

client (obj): creopyson Client.

***file_* (str, optional):** File name. Defaults is currently active model.

Returns: (str): mass units.

file_get_transform (*asm=None*, *path=None*, *csys=None*)
Get the 3D transform for a component in an assembly.

Args:

client (obj): creopyson Client

asm (str, optional): Assembly name. Defaults is currently active model.

path (list:int, optional): Path to a component in the assembly. Defaults is the transform is calculated for the assembly itself.

csys (str, optional): Coordinate system on the component to calculate the transform for. Defaults is the component's default coordinate system.

Returns: (obj:JLTransform): The 3D transform from the assembly to the component's coordinate system.

file_has_instances (*file_=None*)
Check whether a model has a family table.

Args:

client (obj): creopyson Client.

***file_* (str, optional):** File name. Defaults is currently active model.

Returns: (bool): Whether the file has a family table.

file_is_active (*file_*)
Check whether a model is the active model.

Args: client (obj): creopyson Client. *file_* (str): File name.

Returns: (bool): Whether the file is the currently active model.

file_list (*file_=None*)
Get a list of files in the current Creo session that match patterns.

Args:

client (obj): creopyson Client.

***file_* (str|list:str, optional):** File name or List of file names;

Returns: (list:str) List of files.

file_list_instances (*file_=None*)
List instances in a model's family table.

Args:

client (obj): creopyson Client.

***file_* (str, optional):** File name. Defaults is currently active model.

Returns:

(dict): dirname (str): Directory name of the file. generic (str): Generic name. files (list:str): List of model names in the table.

file_list_materials (*file_=None*, *material=None*)
List materials on a part.

Args:

client (obj): creopyson Client.

***file_* (str, optional):** File name. Defaults is currently active model.

material (str, optional): Material name pattern. Wildcards allowed. Defaults to None is all materials.

Returns: list: List of materials in the part.

file_list_materials_wildcard (*file_=None*, *material=None*, *include_non_matching_parts=False*)
List materials on a part or parts.

Args:

client (obj): creopyson Client.

***file_* (str, optional):** File name. Defaults is currently active model.

material (str, optional): Material name pattern. Wildcards allowed. Defaults to None is all materials.

include_non_matching_parts (bool, optional): Whether to include parts that match the part name pattern but don't have any materials matching the material pattern. Defaults to False.

Returns:

list: A list of part and material pairs. If a part has more than one material, it will have multiple entries in this array.

file_list_simp_reps (*file_=None*, *rep=None*)
List simplified reps in a model.

Args:

client (obj): creopyson Client

***file_* (str, optional):** File name. Defaults is currently active model.

rep (str, optional): Simplified rep name pattern (wildcards_allowed: True). Defaults is all simplified reps.

Returns:

(dict): rep (str): Simplified rep name. reps (list:str): Simplified reps names.

file_load_material_file (*material*, *dirname=None*, *file_=None*)
Load a new material file into a part or parts.

Note: If 'material' has a file extension, it will be removed before the material is loaded.

Args:

client (obj): creopyson Client

material (str): Material name

dirname (str, optional): Directory name containing the material file. Default is Creo's 'pro_material_dir' config setting, or search path, or current working directory

***file_* (str, optional):** File name. Wildcards allowed. Defaults is currently active model.

Returns:

list: List of files impacted.

file_massprops (*file_=None*)
Get mass property information about a model.

Notes: PTC's description of coord_sys_inertia: "The inertia matrix with respect to coordinate frame:(element ij is the integral of x_i x_j over the object)". PTC's description of coord_sys_inertia_tensor: "The inertia tensor with respect to coordinate frame: CoordSysInertiaTensor = trace(CoordSysInertia) * identity - CoordSysInertia".

Args:

client (obj): creopyson Client.

***file_* (str, optional):** File name. Defaults is currently active model.

Returns:

(dict):

volume (float): Model volume.

mass (float): Model mass.

density (float): Model density.

surface_area (float): Model surface area.

ctr_grav_inertia_tensor (object:JLInertia): Model's Inertia Tensor translated to center of gravity.

coord_sys_inertia (object:JLInertia): Model's Inertia Matrix with respect to the coordinate frame.

coord_sys_inertia_tensor (object:JLInertia): Model's Inertia Tensor with respect to the coordinate frame.

ctr_grav (object:JLPoint): Model's center of gravity.

file_open (*file_*, *dirname=None*, *generic=None*, *display=None*, *activate=None*, *new_window=None*, *regen_force=None*)

Open one or more files in memory or from the drive.

note: if you open more than one file, it will only put file in your session you won't be able to display more than one file at once.

Args:

client (obj): creopyson Client.

***file_* (str|list:str):** File name or List of file names;

dirname (str, optional): Directory name. Defaults is Creo's current working directory.

generic (str, optional): Generic model name (if file name represents an instance). Defaults to None.

display (bool, optional): Display the model after opening. Defaults is True.

activate (bool, optional): Activate the model after opening. Defaults is True.

new_window (bool, optional): Open model in a new window. Defaults is False.

regen_force (bool, optional): Force regeneration after opening. Defaults is False.

Returns:

(dict):

dirname (str): Directory name of opened file(s).

files (list:str): File names that were opened.

revision (int): Revision of file that was opened; if more than one file was opened, this field is not returned.

file_open_errors (*file_=None*)

Check whether Creo errors have occurred opening a model.

Args:

client (obj): creopyson Client.

***file_* (str, optional):** File name. Defaults is currently active model.

Returns: (bool): Whether errors exist in Creo.

file_postregen_relations_get (*file_=None*)

Get post-regeneration relations for a model.

Args:

client (obj): creopyson Client.

***file_* (str, optional):** File name. Defaults is currently active model.

Returns: (list:str): Exported relations text, one entry per line.

file_postregen_relations_set (*file_=None*, *relations=None*)

Set post-regeneration relations for a model.

Args:

client (obj): creopyson Client.

***file_* (str, optional):** File name. Defaults is currently active model.

relations (list:str, optional): Relations text to import, one line per entry. Clear the relations if missing.

Retunrs: None

file_refresh (*file_=None*)
Refresh the window containing a model.

Args:

client (obj): creopyson Client.

***file_* (str, optional):** File name. Defaults is currently active model.

Returns: None

file_regenerate (*file_=None*, *display=None*)
Regenerate one or more models.

Args:

client (obj): creopyson Client.

***file_* (str|list:str, optional):** File name or List of file names; Defaults is currently active model

display (bool, optional): Display the model before regenerating. Defaults is False.

Returns: None

file_relations_get (*file_=None*)
Get relations for a model.

Args:

client (obj): creopyson Client.

***file_* (str, optional):** File name. Defaults is currently active model.

Returns: (list:srt): Exported relations text, one entry per line.

file_relations_set (*file_=None*, *relations=None*)
Set relations for a model.

Args:

client (obj): creopyson Client.

***file_* (str, optional):** File name. Defaults is currently active model.

relations (list:str, optional): Relations text to import, one line per entry. Clear the relations if missing.

Returns: None

file_rename (*new_name*, *file_=None*, *onlysession=None*)
Rename a model.

Args:

client (obj): creopyson Client.

new_name (str): New file name.

***file_* (str, optional):** File name. Defaults is currently active model.

onlysession (bool, optional): Modify only in memory, not on disk. Defaults is False.

Returns: (str): The new model name.

file_repaint (*file_=None*)
Repaint the window containing a model.

Args:

client (obj): creopyson Client.

***file_* (str, optional):** File name. Defaults is currently active model.

Returns: None

file_save (*file_=None*)
Save one or more models.

Args:

client (obj): creopyson Client.

***file_* (str|list:str, optional):** File name or List of file names; Defaults is currently active model.

Returns: None

file_set_cur_material (*material*, *file_=None*)
Set the current material for a part or parts.

Args:

client (obj): creopyson Client.

material (str): Material name.

***file_* (str, optional):** Part name. Wildcard allowed. Defaults is currently active model.

Returns:

list: list of impacted files.

file_set_length_units (*units*, *file_=None*, *convert=None*)
Set the current length units for a model.

This will search the model's available Unit Systems for the first one which contains the given length unit.

Args:

client (obj): creopyson Client.

units (str): New length units.

***file_* (str|list:str, optional):** File name or List of file names; Defaults is currently active model.

convert (bool, optional): Whether to convert the model's length values to the new units (True) or leave them the same value (False). Defaults is True.

Returns: None

file_set_mass_units (*units*, *file_=None*, *convert=None*)
Set the mass units for a model.

This will search the model's available Unit Systems for the first one which contains the given mass unit.

Args:

client (obj): creopyson Client.

units (str): New mass units.

***file_* (str|list:str, optional):** File name or List of file names; Defaults is currently active model.

convert (bool, optional): Whether to convert the model's mass values to the new units (True) or leave them the same value (False). Defaults is True.

Returns: None

`geometry_bound_box` (*file_=None*)
Get the bounding box for a model.

Args:

client (obj): creopyson Client.

***file_* (str, optional):** File name. Defaults is current active model.

Returns:

(dict): xmin (float): Minimum X-coordinate of model. xmax (float): Maximum X-coordinate of model. ymin (float): Minimum Y-coordinate of model. ymax (float): Maximum Y-coordinate of model zmin (float): Minimum Z-coordinate of model. zmax (float): Maximum Z-coordinate of model

`geometry_get_edges` (*surface_ids*, *file_=None*)
Get the list of edges for a model for given surfaces.

Args:

client (obj): creopyson Client.

surface_ids (list:int): List of surface IDs.

***file_* (str, optional):** File name. Defaults is current active model

Returns:

(list:dict):

surface_id (int): Surface ID.

traversal (string): Traversal type. Valid values: internal, external.

edglist (list:dict):

Information about an edge.

edgeid (integer): Edge ID.

length (float): Edge length.

start (list:dict):

A 3D coordinate.

x (float): X-coordinate.

y (float): Y-coordinate.

z (float): Z-coordinate.

end (list:dict):

A 3D coordinate.

x (float): X-coordinate.

y (float): Y-coordinate.

z (float): Z-coordinate.

geometry_get_surfaces (*file_=None*)

Get the list of surfaces for a model.

Args:

client (obj): creopyson Client.

***file_* (str, optional):** File name. Defaults is current active model

Returns:

(list:dict):

surface_id (int): Surface ID.

area (float): Surface area

min_extent (list:dict):

A 3D coordinate.

x (float): X-coordinate.

y (float): Y-coordinate.

z (float): Z-coordinate.

max_extent (list:dict):

A 3D coordinate.

x (float): X-coordinate.

y (float): Y-coordinate.

z (float): Z-coordinate.

interface_export_3dpdf (*file_=None*, *filename=None*, *dirname=None*, *height=None*, *width=None*, *dpi=None*, *use_drawing_settings=None*, *sheet_range='all'*)

Export a model to a 3D PDF file.

Args:

client (obj): creopyson Client.

***file_* (str, optional):** Model name. Defaults is current active model.

filename (str, optional): Destination file name. May also contain a path to the file. Defaults is the model name with the appropriate file extension, in Creo's working directory.

dirname (str, optional): Destination directory. Defaults is Creo's current working directory.

height (float, optional): PDF image height. Defaults is Creo default export height.

width (float, optional): PDF image width. Defaults is Creo default export width.

dpi (int, optional): PDF Image DPI. Default is Creo default export DPI.

use_drawing_settings (boolean, optional): Whether to use special settings for exporting drawings. Defaut is False.

sheet_range (string): Range of drawing sheets to export. Default vale is "all". Valid values: "all", "current", range of sheet numbers (ex: "1,3-4")

Returns:

dict: dirname (str): Directory of the output file filename (str): Name of the output file

interface_export_file(*file_type*, *file_=None*, *filename=None*, *dirname=None*, *geom_flags=None*, *advanced=None*)

Export a model to a file.

The geom_flags option only applies to IGES and STEP exports. Setting geom_flags to 'default' will cause it to check the Creo config option 'intf3d_out_default_option' for the setting The advanced option will cause the Export to use settings defined in the appropriate *export_profiles* Creo Config Option for the file type. The advanced option only applies to DXF, IGES and STEP exports. The advanced option will only work with Creo 4 M030 or later.

Args:

client (obj): creopyson Client.

file_type (str): File type. Valid values: "DXF", "IGES", "NEUTRAL", "PV", "STEP", "VRML".

***file_* (str, optional):** Model name. Defaults is current active model.

filename (str, optional): Destination file name. May also contain a path to the file. Defaults is the model name with the appropriate file extension, in Creo's working directory.

dirname (str, optional): Destination directory. Defaults is Creo's current working directory.

geom_flags (str, optional): Geometry type for the export. Defaults is *solids*. Valid values: solids, surfaces, wireframe, wireframe_surfaces, quilts, default.

advanced (Boolean, optional): Whether to use the newer Creo 4 file export function. Defaults is False.

Returns:

dict: dirname (str): Directory of the output file filename (str): Name of the output file

interface_export_image(*file_type*, *file_=None*, *filename=None*, *height=None*, *width=None*, *dpi=None*, *depth=None*)

Export a model to an image file.

Args:

client (obj): creopyson Client.

file_type (str): Image Type. Valid values: BMP, EPS, JPEG, TIFF.

***file_* (str, optional):** Model name. Defaults is current active model.

filename (str, optional): Destination file name. May also contain a path to the file. Defaults is the model name with the appropriate file extension, in Creo's working directory.

height (float, optional): Image height. Defaults is *10.0*.

width (float, optional): Image width. Defaults is *7.5*.

dpi (int, optional): Image DPI. Defaults is *100*. Valid values: 100, 200, 300, 400, 500, 600.

depth (int, optional): Image depth. Defaults is *24*. Valid values: 8, 24.

Returns:

dict: dirname (str): Directory of the output file filename (str): Name of the output file

interface_export_pdf(*file_=None*, *filename=None*, *dirname=None*, *height=None*, *width=None*, *dpi=None*, *use_drawing_settings=None*, *sheet_range='all'*)

Export a model to a PDF file.

When use_drawing_settings is true, the Font Stroke option will be set to Stroke All Fonts, and the Color Depth option will be set to Grayscale.

Args:

client (obj): creopyson Client.

***file_* (str, optional):** Model name. Defaults is current active model.

filename (str, optional): Destination file name. May also contain a path to the file. Defaults is the model name with the appropriate file extension, in Creo's working directory.

dirname (str, optional): Destination directory. Defaults is Creo's current working directory.

height (float, optional): PDF image height. Defaults is Creo default export height.

width (float, optional): PDF image width. Defaults is Creo default export width.

dpi (int, optional): PDF Image DPI. Default is Creo default export DPI.

use_drawing_settings (boolean, optional): Whether to use special settings for exporting drawings. Defaut is False.

sheet_range (string): Range of drawing sheets to export. Default vale is "all". Valid values: "all", "current", range of sheet numbers (ex: "1,3-4")

Returns:

dict: dirname (str): Directory of the output file filename (str): Name of the output file

interface_export_program(*file_=None*)

Export a model's program to a file.

Args:

client (obj): creopyson Client.

***file_* (str, optional):** Model name. Defaults is current active model.

Returns:

dict: dirname (str): Directory of the output file filename (str): Name of the output file

interface_import_file(*filename*, *file_type=None*, *dirname=None*, *new_name=None*, *new_model_type='asm'*)

Import a file as a model.

Note: This function will not automatically display or activate the imported model. If you want that, you should take the file name returned by this function and pass it to file:open. Users of the old import_pv function should start using this function instead.

Args:

client (obj): creopyson Client.

filename (str): Source file name.

file_type (str, optional): File type. Valid values: "IGES", "NEUTRAL", "PV", "STEP". Defaults to None. Will analyse filename extension *.igs*|.iges* => IGES *.stp*|.step* => STEP *.neu* => NEUTRAL *.pvz* => PV

dirname (str, optional): Source directory. Defaults is Creo's current working directory.

new_name (str, optional): New model name. Any extension will be stripped off and replaced with one based on new_model_type. Defaults to None is the name of the file with an extension based on new_model_type..

new_model_type (str, optional): New model type. Valid values: "asm", "prt" Defaults to "asm".

Returns: str: Name of the model imported

interface_import_program(*file_=None*, *filename=None*, *dirname=None*)

Import a program file for a model.

Cannot specify both file and filename parameters.

Args:

client (obj): creopyson Client.

***file_* (str, optional):** Destination Model name. Defaults is currently active model, or the model for the filename parameter if given.

filename (str, optional): Source file name. Default is the model name with the appropriate file extension.

dirname (str, optional): Source directory. Defaults is Creo's current working directory.

Returns: str: Name of the model updated

interface_mapkey (*script*, *delay=0*)

Run a Mapkey script in Creo.

Make sure to remove any *mapkey(continued)* clauses from the script argument. The tilde at the start of a line should occur immediately after the semicolon at the end of the previous line.

Args:

client (obj): creopyson Client.

script (str): The mapkey script to run.

delay (int): Amount of time to wait after starting the mapkey, in milliseconds. Default is 0.

Returns: None

interface_plot (*file_=None*, *dirname=None*, *driver=None*)

Export a model plot.

Args:

client (obj): creopyson Client.

***file_* (str, optional):** Model name. Defaults is currently active model.

dirname (str, optional): Destination directory. Defaults is Creo's current working directory.

driver (str, optional): Driver export. Defaults is *POSTSCRIPT*. Valid values: POSTSCRIPT, JPEG, TIFF.

Returns:

dict: dirname (str): Directory of the output file filename (str): Name of the output file

is_creo_running ()

Check whether Creo is running.

This function tests whether the current connection is still active; if there is no active connection, then it tries to make a new connection to Creo and returns whether the connection succeeds. The sessionId is optional, and ignored.

Raises: Warning: error message from creoson.

Returns: (boolean): True if Creo is running, False instead.

kill_creo ()

Kill primary Creo processes.

This will kill the 'xtop.exe' and 'nmsd.exe' processes by name. The sessionId is optional, and ignored.

Raises: Warning: error message from creoson.

Returns: None

layer_delete (*name=None*, *file_=None*)

Delete one or more layers.

Args:

client (obj): creopyson Client.

name (str, optional): Layer name (wildcards allowed: True). Defaults: All layers will be deleted.

***file_* (str, optional):** File name (wildcards alloawed: True). Defaults is current active model.

Returns: None

layer_exists(*name=None*, *file_=None*)
Check whether layer(s) exists on a model.
Args:

client (obj): creopyson Client.

name (str, optional): Layer name (wildcards allowed: True). Defaults: All layers are listed.

***file_* (str, optional):** File name. Defaults is current active model.
Returns: (boolean): Whether the layer exists on the model.

layer_list(*name=None*, *file_=None*)
List layers that match criteria.
Args:

client (obj): creopyson Client.

name (str, optional): Layer name (wildcards allowed: True). Defaults: All layers are listed.

***file_* (str, optional):** File name. Defaults is current active model.
Returns:

(list:dict):

name (str): Layer name.

status (str): Layer status. Valid values: BLANK, DISPLAY, HIDDEN, NORMAL.

ID (int): Layer ID.

layer_show(*name=None*, *file_=None*, *show_=None*)
how/Hide one or more layers.
Args:

client (obj): creopyson Client.

name (str, optional): Layer name (wildcards allowed: True). Defaults: All layers are listed.

***file_* (str, optional):** File name (wildcards allowed: True). Defaults is current active model.

***show_* (boolean, optional):** Whether to show or hide the layers. Defaults is True (show).
Returns: None

note_copy(*name*, *to_name=None*, *file_=None*, *to_file=None*)
Copy note to another in the same model or another model.
Args:

client (obj): creopyson Client.

name (str): Note name to copy (wildcards allowed: True).

to_name (str): Destination note. Defaults is the source note name

***file_* (str, optional):** Model name (wildcards allowed: True). Defaults is current active model.

to_file (str, optional): Destination model. Defaults is the source model.
Returns: None

note_delete(*name*, *file_=None*)
Delete a model or drawing note.
Args:

client (obj): creopyson Client.

name (str): Note name (wildcards allowed: True).

***file_* (str, optional):** Model name (wildcards allowed: True). Defaults is current active model.

Returns: None

note_exists (*file_=None*, *name=None*)

Check whether note(s) exists on a model.

Args:

client (obj): creopyson Client.

***file_* (str, optional):** Model name. Defaults is current active model.

name (str|list:str, optional): Note name; List of note names. if empty it checks for any note's existence.

Returns: (boolean): Whether the note exists on the model.

note_get (*name*, *file_=None*)

Get the text of a model or drawing note.

Values will automatically be returned Base64-encoded if they are strings which contain Creo Symbols or other non-ASCII data.

Args:

client (obj): creopyson Client.

name (str): Note name.

***file_* (str, optional):** Model name. Defaults is current active model or drawing.

Returns:

(dict):

file (str): File name.

name (str): Note name.

encoded (boolean): Value is Base64-encoded.

url (str): "Note URL, if there is one.

location (obj:JLPoint): Note location in Drawing Units (drawing notes only)

note_list (*file_=None*, *name=None*, *value=None*, *get_expanded=None*, *select=False*)

Get a list of notes from one or more models.

Values will automatically be returned Base64-encoded if they are strings which contain Creo Symbols or other non-ASCII data.

Args:

client (obj): creopyson Client.

***file_* (str, optional):** Model name (wildcards allows: True). Defaults is current active model.

name (str|list:str, optional): Note name; List of note names. if empty all notes are listed.

value (str, optional): Parameter value filter (wildcards allows: True). Defaults is *no filter*.

get_expanded (boolean, optional): Whether to return text with parameter values replaced. Defaults is False.

select (boolean, optional): If true, the notes that are found will be selected in Creo. Defaults is False.

Returns:

(list:dict): name (str): Note name. value (str): Note text with parameters not expanded. value_expanded (str): Note text with parameters expanded. encoded (boolean): Value is Base64-encoded. location (jlpoint): 3D coordinate dict.

note_set(*name*, *file_=None*, *location=None*, *encoded=None*, *value=None*)
Set the text of a model or drawing note.

The location parameter can used to position a new note, or to change the position of an existing note. If the text contains Creo Symbols or other non-ASCII text, you must Base64-encode the value and set encoded to true You may be able to avoid Base64-encoding symbols by using Unicode for the binary characters, for example including \u0001#\u0002 in the value to insert a plus/minus symbol. Embed newlines in the value for line breaks.
Args:

client (obj): creopyson Client.

name (str): Note name (wildcards allowed: True).

***file_* (str, optional):** Model name. Defaults is current active model or drawing.

location (JLPoint): Coordinates for the note placement in Drawing Units. If missing and this is a new note, note will be placed at 0, 0.

encoded (boolean, optional): Whether the value is Base64-encoded. Defaults is False.

value (str, optional): Note text with parameters not expanded. Defaults to None: clears the note if missing.

Returns: None

parameter_copy(*name*, *to_name*, *file_=None*, *to_file=None*, *designate=None*)
Copy parameter to another in the same model or another model.
Args:

client (obj): creopyson Client.

name (str): Parameter name to copy (wildcards allowed: True).

to_name (str): Destination parameter.

***file_* (str, optional):** Model name. Defaults is current active model.

to_file (str, optional): Destination model (wildcards allowed: True). Defaults is the source model.

designate (boolean, optional): Set copied parameter to be designated/not designated, blank=do not set. Defaults is *blank*.

Returns: None

parameter_delete(*name*, *file_=None*)
Delete a parameter.
Args:

client (obj): creopyson Client.

name (str): Parameter name (wildcards allowed: True).

***file_* (str, optional):** Model name. Defaults is current active model.

Returns: None

parameter_exists(*name=None*, *file_=None*)
Check whether parameter(s) exists on a model.
Args:

client (obj): creopyson Client.

name (str|list:str, optional): Parameter name; List of parameter names. if empty it checks for any parameter's existence.

***file_* (str, optional):** Model name. Defaults is current active model.

Returns: (boolean): Whether the parameter exists on the model.

parameter_list(*name=None*, *file_=None*, *encoded=None*, *value=None*)

Get a list of parameters from one or more models.

Args:

client (obj): creopyson Client.

name (str|list:str, optional): Parameter name; List of parameter names. if empty it checks for any parameter's existence.

***file_* (str, optional):** Model name. Defaults is current active model.

encoded (boolean, optional): Whether to return the values Base64-encoded. Defaults is False.

value (str, optional): Parameter value filter. Defaults is *no filter*.

Returns:

(list:dict): name (str): Parameter name. type (str): Parameter type. value (various): Parameter value # TODO designate (boolean): Whether the parameter is designated. description (str): Description. encoded (boolean): Whether the parameter is encoded. owner_name (str): File name.

parameter_set(*name*, *value=None*, *file_=None*, *type_=None*, *encoded=None*, *designate=None*, *description=None*, *no_create=None*)

Set the value of a parameter.

Args:

client (obj): creopyson Client.

name (str): Parameter name (wildcards allowed: True).

value (depends on data type, optional): Parameter value. Defaults to None. Clears the parameter value if missing.

***file_* (str, optional):** Model name. Defaults is current active model.

***type_* (str, optional):** Data type. Defaults is *STRING*. Valid values: STRING, DOUBLE, INTEGER, BOOL, NOTE.

encoded (boolean, optional): Whether the value is Base64-encoded. Defaults is False.

designate (boolean, optional): Set parameter to be designated/not designated, blank=do not set. Defaults is *blank*.

description (str, optional): Parameter description. If missing, leaves the current description in place.

no_create (boolean, optional): If parameter does not already exist, do not create it. Defaults is False.

Returns: None

parameter_set_designated(*name*, *designate*, *file_=None*)

Set the designated state of a parameter.

Args:

client (obj): creopyson Client.

name (str): Parameter name (wildcards allowed: True).

designate (boolean): Set parameter to be designated/not designated.

***file_* (str, optional):** Model name. Defaults is current active model.

Returns: None

server_pwd()

Return the creoson server's execution directory.

Args:

client (obj): creopyson Client.
Raises: Warning: error message from creoson.
Returns: (str): Full name of working directory.

start_creo(*path*, *retries=0*, *use_desktop=False*)
Execute an external .bat file to start Creo.

Then attempts to connect to Creo.

The .bat file is restricted to a specific name to make the function more secure. (nitro_proe_remote.bat) Set retries to 0 to NOT attempt to connect to Creo. The server will pause for 3 seconds before attempting a connection, and will pause for 10 seconds between connection retries. If Creo pops up a message after startup, this function may cause Creo to crash unless retries is set to 0. If use_desktop is set, make sure that your nitro_proe_remote.bat file contains a cd command to change to the directory where you want Creo to start!
Args:

path (string): path to the .bat file (must be full path) will be split in 'start_command' and 'start_dir'

retries (int): Number of retries to make when connecting (default 0)

use_desktop (boolean): Whether to use the desktop to start creo rather than the java runtime. Should only be used if the runtime method doesn't work. Default is False.
Raises: Warning: error message from creoson.
Returns: None

stop_creo()
Disconnect current session from Creo and cause Creo to exit.

NOTE that this will cause Creo to exit cleanly. If there is no current connection to Creo, this function will do nothing.
Raises: Warning: error message from creoson.
Returns: None

view_activate(*name*, *file_=None*)
Activate a model view.
Args:

client (obj): creopyson Client.

name (str): View name.

***file_* (str, optional):** Model name. Defaults is current active model.
Returns: None

view_list(*file_=None*, *name=None*)
List views that match criteria.
Args:

client (obj): creopyson Client.

***file_* (str, optional):** Model name. Defaults is current active model.

name (str, optional): View name (wildcards allowed: True). Defaults is None: all views are listed.
Returns: (list:str): List of view names.

view_list_exploded(*file_=None*, *name=None*)
List views that match criteria and are exploded.
Args:

client (obj): creopyson Client.

***file_* (str, optional):** Model name. Defaults is current active model.

name (str, optional): View name (wildcards allowed: True). Defaults is None: all views are listed.

Returns: (list:str): List of view names.

view_save(*name*, *file_=None*)

Save a model's current orientation as a new view.

Args:

client (obj): creopyson Client.

name (str): View name.

***file_* (str, optional):** Model name. Defaults is current active model.

Returns: None

windchill_authorize(*user*, *password*)

Set user's Windchill login/password.

Args: client (obj): creopyson Client user (str): user name password (str): password

Returns: None

windchill_clear_workspace(*workspace=None*, *filenames=None*)

Clear a workspace on the active server.

Args:

client (obj): creopyson Client

workspace (str, optionnal): Workspace name. Default is current workspace.

filenames (str|list, optionnal): List of files to delete from the workspace. Default: All files are deleted.

Returns: None

windchill_create_workspace(*workspace*, *context_name*)

Create a workspace on the active server.

Args: client (obj): creopyson Client workspace (str): Workspace name context_name (str): Context name

Returns: None

windchill_delete_workspace(*workspace*)

Delete a workspace on the active server.

Args: client (obj): creopyson Client workspace (str): Workspace name

Returns: None

windchill_file_checked_out(*filename*, *workspace=None*)

Check whether a file is checked out in a workspace on the active server.

Args:

client (obj): creopyson Client

filename (str): File name

workspace (str, optionnal): Workspace name. Default is current workspace.

Returns: Boolean: Whether the file is checked out in the workspace.

windchill_get_workspace()

Retrieve the name of the active workspace on the active server.

Args: client (obj): creopyson Client

Returns: str: Active Workspace name.

windchill_list_workspace_files(*workspace=None*, *filename=None*)

Get a list of files in a workspace on the active server.

Args:

client (obj): creopyson Client

workspace (str, optionnal): Workspace name. Default is current workspace.

filename (str, optional): File name or search. Default is all files. ex: **.asm*, *screw_*.prt*

Returns: list: List of files in the workspace correspnding to the data.

windchill_list_workspaces()
Get a list of workspaces the user can access on the active server.
Args: client (obj): creopyson Client
Returns: list: List of workspaces

windchill_server_exists(*server_url*)
Check whether a server exists.
Args: client (obj): creopyson Client server_url (str): server URL or Alias
Returns: Boolean: Whether the server exists

windchill_set_server(*server_url*)
Select a Windchill server.
Args: client (obj): creopyson Client server_url (str): server URL or Alias
Returns: None

windchill_set_workspace(*workspace*)
Select a workspace on the active server.
Args: client (obj): creopyson Client workspace (str): Workspace name
Returns: None

windchill_workspace_exists(*workspace*)
Check whether a workspace exists on the active server.
Args: client (obj): creopyson Client workspace (str): Workspace name
Returns: Boolean: Whether the workspace exists

4.1.4 creopyson.creo module

Creo module.

creopyson.creo.**cd**(*client*, *dirname*)
Change Creo's working directory.

You can use absolute path or relative: "C:\My Workdir" "..\Other_directory"
Args: client (obj): creopyson Client. dirname (str): New directory name.
Returns: (str): Name of new working directory.

creopyson.creo.**delete_files**(*client*, *filename=None*, *dirname=None*)
Delete files from a directory working directory.
Args:
client (obj): creopyson Client.
filename (str or list:str, optional): File name filter or list of file names. if blank all files will be deleted. (wildcards_allowed: True). Defaults to None.
dirname (str, optional): Directory name. Defaults is Creo's current working directory.
Returns: (list:str): List of deleted files.

creopyson.creo.**get_config**(*client*, *name*)
Get the value of a Creo config option.
Args: client (obj): creopyson Client. name (str): Option name.
Raises: Warning: error message from creoson.
Returns:
(list:str): List of option values (some options can have multiple values).

creopyson.creo.**get_std_color**(*client*, *color_type*)

Get one of Creo's standard colors.

Args:

client (obj): creopyson Client.

color_type (str): Color type. Valid values: letter, highlight, drawing, background, half_tone, edge_highlight, dimmed, error, warning, sheetmetal, curve, presel_highlight, selected, secondary_selected, preview, secondary_preview, datum, quilt.

Returns:

(dict): red (int): Red value (0-255) green (int): Green value (0-255) blue (int): Blue value (0-255)

creopyson.creo.**list_dirs**(*client*, *dirname=None*)

List subdirectories of Creo's current working directory.

Args:

client (obj): creoopyson Client.

dirname (str, optional): Directory name filter (wildcards_allowed: True). Defaults: All subdirectories are listed.

Returns: (list:str): List of subdirectories

creopyson.creo.**list_files**(*client*, *filename=None*)

List files in Creo's current working directory.

Args:

client (obj): creopyson Client.

filename (str, optional): File name filter (wildcards_allowed: True). Defaults: all files are listed.

Returns: (list:int): List of files.

creopyson.creo.**mkdir**(*client*, *dirname*)

Create a new directory.

Args: client (obj): creopyson Client. dirname (str): New directory name.

Returns: (str): Full name of new working directory.

creopyson.creo.**pwd**(*client*)

Return Creo's current working directory.

Args: client (obj): creopyson Client.

Returns: (str): Full name of working directory.

creopyson.creo.**rmdir**(*client*, *dirname*)

Delete a directory.

Args: client (obj): creopyson Client. dirname (str): Directory name to delete.

Returns: None.

creopyson.creo.**set_config**(*client*, *name*, *value*, *ignore_errors=None*)

Set a Creo config option.

Args:

client (obj): creopyson Client.

name (str): Option name.

value (str): New option value.

ignore_errors (boolean, optional): Whether to ignore errors that might occur when setting the config option. Defaults is False.

Returns: None.

creopyson.creo.**set_creo_version**(*client*, *version*)

Set the version of Creo you are running.

This function only needs to be called once per creoson session. This function must be called if you are doing certain functions in Creo 7 or later due to deprecated config options. At this time this function only supports 7, 8 and 9..

This is needed for functions: familytable_replace file_assemble file_regenerate feature_delete feature_resume feature_suppress

Args:
client (obj): creopyson Client.
version (int): Creo version.
Returns: None.

creopyson.creo.**set_std_color**(*client*, *color_type*, *red*, *green*, *blue*)
Set one of Creo's standard colors.
Args:
client (obj): creopyson Client.
color_type (str): Color type. Valid values: letter, highlight, drawing, background, half_tone, edge_highlight, dimmed, error, warning, sheetmetal, curve, presel_highlight, selected, secondary_selected, preview, secondary_preview, datum, quilt.
red (int): Red value (0-255).
green (int): Green value (0-255).
blue (int): Blue value (0-255).
Returns: None.

4.1.5 creopyson.dimension module

Dimension module.

creopyson.dimension.**copy**(*client*, *name*, *to_name*, *file_=None*, *to_file=None*)
Copy dimension to another in the same model or another model.
Args:
client (obj): creopyson Client.
name (str): Dimension name to copy.
to_name (str): Destination dimension; th dimension must already exist.
***file_* (str, optional):** Model name. Defaults is current active model.
to_file (str, optional): Destination model. Defaults is the source model.
Returns: None

creopyson.dimension.**list_**(*client*, *name=None*, *file_=None*, *dim_type=None*, *encoded=None*, *select=False*)
Get a list of dimensions from a model.

If select is true, then the current selection in Creo will be cleared even if no items are found.
Args:
client (obj): creopyson Client.
name (str|list:str, optional): Dimension name; if empty then all dimensions are listed.
***file_* (str, optional):** Model name. Defaults is current active model.
dim_type (str, optional): Dimension type filter. Defaults is *no filter*. Valid values: linear, radial, diameter, angular.
encoded (boolean, optional): Whether to return the values Base64-encoded. Defaults is False.
select (boolean, optional): If true, the dimensions that are found will be selected in Creo. Defaults is False.
Returns:
(list:dict): List of dimension information.

name (str): Dimension name

value (str|float): Dimension value; if encoded is True it is a str, if encoded is False it is a float.

encoded (boolean): Whether the returned value is Base64-encoded.

dwg_dim (boolean): Whether dimension is a drawing dimension rather than a model dimension.

creopyson.dimension.**list_detail**(*client*, *name=None*, *file_=None*, *dim_type=None*, *encoded=None*, *select=False*)

Get a list of dimension details from a model.

Values will automatically be returned Base64-encoded if they are strings which contain Creo Symbols or other non-ASCII data. If select is true, then the current selection in Creo will be cleared even if no items are found.

Args:

client (obj): creopyson Client.

name (str|list:str, optional): Dimension name; if empty then all dimensions are listed.

***file_* (str, optional):** Model name. Defaults is current active model.

dim_type (str, optional): Dimension type filter. Defaults is *no filter*. Valid values: linear, radial, diameter, angular.

encoded (boolean, optional): Whether to return the values Base64-encoded. Defaults is False.

select (boolean, optional): If true, the dimensions that are found will be selected in Creo. Defaults is False.

Returns:

(list:dict): List of dimension information.

name (str): Dimension name

value (str|float): Dimension value; if encoded is True it is a str, if encoded is False it is a float.

encoded (boolean): Whether the returned value is Base64-encoded.

sheet (int): Sheet number.

view_name (str): View name.

dim_type (str): Dimension type. Valid values: linear, radial, diameter, angular.

dwg_dim (boolean): Whether dimension is a drawing dimension rather than a model dimension.

text (str): dimension text.

location (dict): Coordonates location. x (float): X coordonate location. y (float): Y coordonate location. z (float): Z coordonate location.

tolerance_type (str): Tolerance type, if not specified not returned. Valid values: plus_minus (TODO complete list).

tol_plus (float): Plus tolerance value. if tolerance_type not specified not returned.

tol_minus (float): Minus tolerance value. if tolerance_type not specified not returned.

creopyson.dimension.**set_**(*client*, *name*, *value*, *file_=None*, *encoded=None*)

Set a dimension value.

One reason to encode values is if the value contains special characters, such as Creo symbols. You may be able to avoid Base64-encoding symbols by using Unicode for the binary characters, for example including \u0001#\u0002 in the value to insert a plus/minus symbol.

Args:

client (obj): creopyson Client.

name (str): Dimension name.

value (str|float): Dimension value.

***file_* (string, optional):** file name, if not set, active model is used.

encoded (boolean, optional): Whether the value is Base64-encoded. Defaults is False.

Raises: Warning: error message from creoson.

Returns: None

creopyson.dimension.**set_text**(*client*, *name*, *file_=None*, *text=None*, *encoded=False*)

Set dimension text.

Args:

client (obj): creopyson object.
name (str): Dimension name.
***file_* (string, optional):** file name, if not set, active model is used.
text ([type], optional): Dimension text. Defaults to None, sets the dimension's text to @D.
encoded (bool, optional): Whether the text value is Base64-encoded. Defaults to False.

Returns: None

`creopyson.dimension.`**`show`**(*client*, *name*, *file_=None*, *assembly=None*, *path=None*)

Display or hide a dimension in Creo.

Args:

client (obj): creopyson Client.
name (str): Dimension name.
***file_* (str, optional):** Model name. Defaults is current active model.
assembly (str, optional): Assembly name; only used if path is given. Defaults is the currently active model.
path (list:int, optional): Path to occurrence of the model within the assembly; the dimension will only be shown for that occurrence. Defaults: all occurrences of the component are affected.

Returns: None

`creopyson.dimension.`**`user_select`**(*client*, *file_=None*, *maxi=None*)

Prompt user to select one or more dimensions, and return their selections.

client (obj): creopyson Client.
***file_* (str, optional):** Model name. Defaults is current active model.
maxi (int, optional): The maximum number of dimensions that the user can select. Defaults is *1*.

Raises: Warning: error message from creoson.

Returns:

(list:dict): List of selected dimension information

name (str): Dimension name

value (str|float): Dimension value; if encoded is True it is a str, if encoded is False it is a float.

encoded (boolean): Whether the returned value is Base64-encoded.

file (str): File name.

relation_id (int): Relation ID number.

4.1.6 creopyson.drawing module

Drawing module.

`creopyson.drawing.`**`add_model`**(*client*, *model*, *drawing=None*)

Add a model to a drawing.

Args:

client (obj): creopyson Client.
model (str): Model name.
drawing (str, optional): Drawing name. Defaults is Current active drawing.

Returns: None

`creopyson.drawing.`**`add_sheet`**(*client*, *position=None*, *drawing=None*)

Add a drawing sheet.

Args:

client (obj): creopyson Client.
position (int, optional): Position to add the sheet. Defaults: Sheet will be added to the end.
drawing (str, optional): Drawing name. Defaults is current active drawing.

Returns: None

creopyson.drawing.**create**(*client*, *template*, *model=None*, *drawing=None*, *scale=None*, *display=None*, *activate=None*, *new_window=None*)

Create a new drawing from a template.

Args:

client (obj): creopyson Client.
template (str): Template
model (str, optional): Model name. Defaults: Current active model.
drawing (str, optional): New drawing name. Defaults: A name derived from the model's instance name.
scale (float, optional): Drawing scale. Defaults is *1.0*.
display (boolean, optional): Display the drawing after open. Defaults is False.
activate (boolean, optional): Activate the drawing window after open. Defaults is False.
new_window (boolean, optional): Open drawing in a new window. Defaults is False.

Returns: (str): New drawing name.

creopyson.drawing.**create_gen_view**(*client*, *model_view*, *point*, *drawing=None*, *view=None*, *sheet=None*, *model=None*, *scale=None*, *display_data=None*, *exploded=None*)

Create general view on a drawing.

Args:

client (obj): creopyson Client
model_view (str): Model view to use for the drawing view orientation.
point (dict): Coordinates for the view in Drawing Units.
drawing (str, optional): Drawing name. Defaults: current active drawing.
view (str, optional): New view name. Defaults: the model_view parameter.
sheet (int, optional): Sheet number. Defaults: current active sheet on the drawing.
model (str, optional): Model for the view. Defaults: current active model on the drawing.
scale (float, optional): View scale. Defaults: the sheet's scale.
display_data (dict, optional): Display parameters used to create the view. Defaults: Creo defaults.
exploded (boolean, optional): Whether to create the view as an exploded view. Defaults is False.

Returns: None

creopyson.drawing.**create_proj_view**(*client*, *parent_view*, *point*, *drawing=None*, *view=None*, *sheet=None*, *display_data=None*, *exploded=None*)

Create projection view on a drawing.

When specifying the view coordinates, you should specify only an X or a Y coordinate to avoid confusion. If you specify both coordinates, it appears Creo may be using whichever has the larger absolute value.

Args:

client (obj): creopyson Client
parent_view (str): Parent view for the projection view.
point (dict): Coordinates for the view, relative to the location of the parent view, in Drawing Units.
drawing (str, optional): Drawing name. Defaults: current active drawing.
view (str, optional): New view name. Defaults: Creo's default name for a new view.
sheet (int, optional): Sheet number. Defaults: current active sheet on the drawing.
display_data (dict, optional): Display parameters used to create the view. Defaults: the display parameters of the parent view.
exploded (boolean, optional): Whether to create the view as an exploded view. Defaults is False.

Returns: None

creopyson.drawing.**create_symbol**(*client*, *symbol_file*, *point*, *drawing=None*, *replace_values=None*, *sheet=None*)

Add a symbol instance to a drawing.

Args:

client (obj): creopyson Client
symbol_file (str): Name of the symbol file.
point (dict): Coordinates for the symbol in Drawing Units.
drawing (str, optional): Drawing name. Defaults: current active drawing.

replace_values (dict, optional): Object containing replacement values for any variable text in the symbol. Defaults to None.
sheet (int, optional): Sheet number (0 for all sheets). Defaults: the symbol will be added to all sheets.
Returns: None

creopyson.drawing.**delete_models**(*client*, *model=None*, *drawing=None*, *delete_views=None*)
Delete one or more models from a drawing.
Args:
client (obj): creopyson Client
model (str, optional): Model name (wildcard allowed: True). Defaults: all models will be deleted from the drawing.
drawing (str, optional): Drawing name. Defaults: current active drawing.
delete_views (boolean, optional): Whether to delete drawing views associated with the model. Defaults is False.
Returns: None

creopyson.drawing.**delete_sheet**(*client*, *sheet*, *drawing=None*)
Delete a drawing sheet.

An error will occur if you try to delete the only sheet in a drawing.
Args:
client (obj): creopyson Client
sheet (int): Sheet number.
drawing (str, optional): Drawing name. Defaults: current active drawing.
Returns: None

creopyson.drawing.**delete_symbol_def**(*client*, *symbol_file*, *drawing=None*)
Delete a symbol definition and its instances from a drawing.
Args:
client (obj): creopyson Client
symbol_file (str): Name of the symbol file.
drawing (str, optional): Drawing name. Defaults: current active drawing.
Returns: None

creopyson.drawing.**delete_symbol_inst**(*client*, *symbol_id*, *drawing=None*)
Delete a specific symbol instance from a drawing.
Args:
client (obj): creopyson Client
symbol_id (str): ID of the symbol instance.
drawing (str, optional): Drawing name. Defaults: current active drawing.
Returns: None

creopyson.drawing.**delete_view**(*client*, *view*, *drawing=None*, *sheet=None*, *del_children=None*)
Delete a drawing view.
Args:
client (obj): creopyson Client
view (str): View name.
drawing (str, optional): Drawing name. Defaults: current active drawing.
sheet (int, optional): Sheet number; if filled in, the view will only be deleted if it is on that sheet. Defaults: Delete the view from any sheet.
del_children ([boolean, optional): Whether to also delete any children of the view. Defaults is False.
Returns: None

creopyson.drawing.**get_cur_model**(*client*, *drawing=None*)
Get the active model on a drawing.
Args:
client (obj): creopyson Client

drawing (str, optional): Drawing name. Defaults: current active drawing.
Returns: (str): Model name.

creopyson.drawing.**get_cur_sheet**(*client*, *drawing=None*)
Get the current drawing sheet.
Args:
client (obj): creopyson Client
drawing (str, optional): Drawing name. Defaults: current active drawing.
Returns: (int): Sheet number.

creopyson.drawing.**get_num_sheets**(*client*, *drawing=None*)
Get the number of sheets on a drawing.
Args:
client (obj): creopyson Client
drawing (str, optional): Drawing name. Defaults: current active drawing.
Returns: (int): Number of sheets.

creopyson.drawing.**get_sheet_format**(*client*, *sheet*, *drawing=None*)
Get the drawing format file of drawing sheet.
Args:
client (obj): creopyson Client.
sheet (int): Sheet number.
drawing (str, optional): Drawing name. Defaults: current active drawing.
Returns:
(dict):

file(str): Format file name, may be null if there is no current format.

full_name(str): Format full name.

common_name(str): Format common name.

creopyson.drawing.**get_sheet_scale**(*client*, *sheet*, *drawing=None*, *model=None*)
Get the scale of a drawing sheet.
Args:
client (obj): creopyson Client
sheet (int): Sheet number.
drawing (str, optional): Drawing name. Defaults: current active drawing.
model (str, optional): Drawing model used to calculate the scale. Defaults: the active model on the drawing.
Returns: (float): Sheet scale.

creopyson.drawing.**get_sheet_size**(*client*, *sheet*, *drawing=None*)
Get the size of a drawing sheet.
Args:
client (obj): creopyson Client
sheet (int): Sheet number.
drawing (str, optional): Drawing name. Defaults: current active drawing.
Returns: (str): Sheet size.

creopyson.drawing.**get_view_loc**(*client*, *view*, *drawing=None*)
Get the location of a drawing view.
Args:
client (obj): creopyson Client
view (str): View name.
drawing (str, optional): Drawing name. Defaults: current active drawing.
Returns:

(dict): x (float): X-coordinate of the view y (float): Y-coordinate of the view z (float): Z-coordinate of the view

creopyson.drawing.**get_view_scale**(*client*, *view*, *drawing=None*)

Get the scale of a drawing view.

Args:

client (obj): creopyson Client
view (str): View name.
drawing (str, optional): Drawing name. Defaults: current active drawing.

Raises: Warning: error message from creoson.

Returns: (float): View scale.

creopyson.drawing.**get_view_sheet**(*client*, *view*, *drawing=None*)

Get the sheet number that contains a drawing view.

Args:

client (obj): creopyson Client
view (str): View name.
drawing (str, optional): Drawing name. Defaults: current active drawing.

Returns: (int): Sheet number.

creopyson.drawing.**is_symbol_def_loaded**(*client*, *symbol_file*, *drawing=None*)

Check whether a symbol definition file is loaded into Creo.

Args:

client (obj): creopyson Client
symbol_file (str): Name of the symbol file.
drawing (str, optional): Drawing name. Defaults: current active drawing.

Returns: (boolean): Whether the symbol definition is loaded into Creo.

creopyson.drawing.**list_models**(*client*, *model=None*, *drawing=None*)

List the models contained in a drawing.

Args:

client (obj): creopyson Client
model (str, optional): Model name filter (wildcards allowed: True). Defaults: no filter.
drawing (str, optional): Drawing name. Defaults: current active drawing.

Returns: (list:str): List of model names in the drawing.

creopyson.drawing.**list_symbols**(*client*, *drawing=None*, *symbol_file=None*, *sheet=None*)

List symbols contained on a drawing.

Args:

client (obj): creopyson Client
drawing (str, optional): Drawing name. Defaults: current active drawing.
symbol_file (str, optional): Symbol file name filter. Defaults: no filter.
sheet (int, optional): Sheet number (0 for all sheets). Defaults: The symbol will be added to all sheets.

Returns:

(list:dict):

List of symbols in the drawing. id (int): Symbol ID. symbol_name (str): Symbol name. sheet (int): Sheet number.

creopyson.drawing.**list_view_details**(*client*, *view=None*, *drawing=None*)

List the views contained in a drawing, with more details.

Args:

client (obj): creopyson Client
view (str, optional): View name filter (wildcards allowed: True). Defaults: no filter.
drawing (str, optional): Drawing name. Defaults: current active drawing.

Returns:

(list:dict):

List of views in the drawing

name (str): View name.

sheet (int): Sheet number.

location (dict):

Coordonates x (float): X-coordinate of the view y (float): Y-coordinate of the view z (float): Z-coordinate of the view

text_height (float): Text Heigh in Drawing Units.

view_model (str): View model name.

simp_rep (str): View simplified rep name.

creopyson.drawing.**list_views**(*client*, *view=None*, *drawing=None*)

List the views contained in a drawing.

Args:

client (obj): creopyson Client
view (str, optional): View name filter (wildcards allowed: True). Defaults: no filter.
drawing (str, optional): Drawing name. Defaults: current active drawing.

Returns: (list:str): List of views in the drawing.

creopyson.drawing.**load_symbol_def**(*client*, *symbol_file*, *symbol_dir=None*, *drawing=None*)

Load a Creo symbol definition file into Creo from disk.

Args:

client (obj): creopyson Client
symbol_file (str): Name of the symbol file.
symbol_dir (str, optional): Directory containing the symbol file; if relative, assumed to be relative to Creo's current working directory. Defaults: Creo's current working directory.
drawing (str, optional): Drawing name. Defaults: current active drawing.

Returns:

(dict): Symbol definition. id (int): ID of the loaded symbol. name (str): Symbol Name of the loaded symbol.

creopyson.drawing.**regenerate**(*client*, *drawing=None*)

Regenerate a drawing.

Args:

client (obj): creopyson Client
drawing (str, optional): Drawing name. Defaults: current active drawing.

Returns: None

creopyson.drawing.**regenerate_sheet**(*client*, *sheet=None*, *drawing=None*)

Regenerate a sheet on a drawing.

Args:

client (obj): creopyson Client
sheet (int, optional): Sheet number (0 for all sheets). Defaults: all sheets will be regenerated.
drawing (str, optional): Drawing name. Defaults: current active drawing.

Returns: None

creopyson.drawing.**rename_view**(*client*, *view*, *new_view*, *drawing=None*)

Rename a drawing view.

Args:

client (obj): creopyson Client
view (str): Old view name.
new_view (str): New view name.
drawing (str, optional): Drawing name. Defaults: current active drawing.

Returns: None

creopyson.drawing.**scale_sheet**(*client*, *sheet*, *scale*, *drawing=None*, *model=None*)

Set the scale of a drawing sheet.

Args:

client (obj): creopyson Client
sheet (int): Sheet number.
scale (float): View scale.
drawing (str, optional): Drawing name. Defaults: current active drawing.
model (str, optional): Drawing model to scale. Defaults: tThe active model on the drawing.

Returns: None

creopyson.drawing.**scale_view**(*client*, *scale*, *view=None*, *drawing=None*)

Set the scale of one or more drawing views.

Args:

client (obj): creopyson Client
scale (float): View scale.
view (str, optional): View name (wildcards allowed: True). Defaults: all views will be scaled.
drawing (str, optional): Drawing name. Defaults: current active drawing.

Returns:

(dict)

succes_views (list): List of view which were successfully scaled.

failed_views (list): List of view which failed to scale.

creopyson.drawing.**select_sheet**(*client*, *sheet*, *drawing=None*)

Make a drawing sheet the current sheet.

Args:

client (obj): creopyson Client
sheet (int): Sheet number.
drawing (str, optional): Drawing name. Defaults: current active drawing.

Returns: None

creopyson.drawing.**set_cur_model**(*client*, *model*, *drawing=None*)

Set the active model on a drawing.

Args:

client (obj): creopyson Client
model (str): Model name.
drawing (str, optional): Drawing name. Defaults: current active drawing.

Returns: None

creopyson.drawing.**set_sheet_format**(*client*, *sheet*, *file_format*, *dirname=None*, *drawing=None*)

Set the drawing format file of a drawing sheet.

Args:

client (obj): creopyson Client.
sheet (int): Sheet number.
file_format (str): Format file name.
dirname (str, optional): Directory name containing the file format. Defaults to None is current working directory.
drawing (str, optional): Drawing name. Defaults to None is current active drawing.

Returns: None

creopyson.drawing.**set_view_loc**(*client*, *view*, *point*, *drawing=None*)

Set the location of a drawing view.

Args:

client (obj): creopyson Client
view (str): View name.
point (dict): Coordinates for the view in Drawing Units

drawing (str, optional): Drawing name. Defaults: current active drawing.
Returns: None

creopyson.drawing.**view_bound_box**(*client*, *view*, *drawing=None*)
Get the 2D bounding box for a drawing view.
Args:
client (obj): creopyson Client
view (str): View name.
drawing (str, optional): Drawing name. Defaults: current active drawing.
Returns:
(dict): xmin (float): Minimum X-coordinate of drawing view. xmax (float): Maximum X-coordinate of drawing view. ymin (float): Minimum Y-coordinate of drawing view. ymax (float): Maximum Y-coordinate of drawing view.

4.1.7 creopyson.exceptions module

Creopyson exceptions.

exception creopyson.exceptions.**Error**
Bases: Exception

Base class for other exceptions.

exception creopyson.exceptions.**ErrorJsonDecode**
Bases: *creopyson.exceptions.Error*

Raised when creoson result cannot be decoded.

exception creopyson.exceptions.**MissingKey**
Bases: *creopyson.exceptions.Error*

Raised when the input value is too small.

4.1.8 creopyson.familytable module

Familytable module.

creopyson.familytable.**add_inst**(*client*, *instance*, *file_=None*)
Add a new instance to a family table.

Creates a family table if one does not exist.
Args:
client (obj): creopyson Client.
instance (str): New instance name.
***file_* (str, optional):** File name. Defaults is currently active model.
Returns: None

creopyson.familytable.**create_inst**(*client*, *instance*, *file_=None*)
Create a new model from a family table row.
Args:
client (obj): creopyson Client.
instance (str): Instance name.
***file_* (str, optional):** File name. Defaults is currently active model.
Returns: (str): New model name.

creopyson.familytable.**delete**(*client*, *file_=None*)
Delete an entire family table.
Args:

client (obj): creopyson Client.
***file_* (str, optional):** File name (wildcards allowed: True). Defaults is currently active model.
Returns: None

creopyson.familytable.**delete_inst**(*client*, *instance*, *file_=None*)
Delete an instance from a family table.
Args:
client (obj): creopyson Client.
instance (str): Instance name.
***file_* (str, optional):** File name. Defaults is currently active model.
Returns: None

creopyson.familytable.**exists**(*client*, *instance*, *file_=None*)
Check whether an instance exists in a family table.
Args:
client (obj): creopyson Client.
instance (str): Instance name.
***file_* (str, optional):** File name. Defaults is currently active model.
Returns:
(boolean): Whether the instance exists in the model's family table; returns false if there is no family table in the model.

creopyson.familytable.**get_cell**(*client*, *instance*, *colid*, *file_=None*)
Get one cell of a family table.
Args:
client (obj): creopyson Client.
instance (str): Instance name.
colid (str): Colimn ID.
***file_* (str, optional):** File name. Defaults is currently active model.
Returns:
(dict):

instance (str): Family Table instance name.

colid (str): Column ID.

value (depends on datatype): Cell value.

datatype (str): Data type.

coltype (str): Column Type; a string corresponding to the Creo column type.

creopyson.familytable.**get_header**(*client*, *file_=None*)
Get the header of a family table.
Args:
client (obj): creopyson Client.
***file_* (str, optional):** File name. Defaults is currently active model.
Returns:
(list:dict):

colid (str): Column ID.

value (depends on date type): Cell value.

datatype (str): Data type. Valid values: STRING, DOUBLE, INTEGER, BOOL, NOTE.

coltype (str): Column Type; a string corresponding to the Creo column type.

creopyson.familytable.**get_parents**(*client*, *file_=None*)
Get the parent instances of a model in a nested family table.
Args:

client (obj): creopyson Client.
***file_* (str, optional):** File name. Defaults is currently active model.
Returns:
(list:str): List of parent instance names, starting with the immediate parent and working back.

creopyson.familytable.**get_row**(*client*, *instance*, *file_=None*)
Get one row of a family table.
Args:
client (obj): creopyson Client.
instance (str): Instance name.
***file_* (str, optional):** File name. Defaults is currently active model.
Returns:
(dict):

colid (str): Column ID.

value (depends on datatype): Cell value.

datatype (str): Data type.

coltype (str): Column Type; a string corresponding to the Creo column type.

creopyson.familytable.**list_**(*client*, *file_=None*, *instance=None*)
List the instance names in a family table.
Args:
client (obj): creopyson Client.
instance (str, optional): Instance name filter (wildcards allowed: True). Defaults is all instances.
***file_* (str, optional):** File name. Defaults is currently active model.
Returns: (list:str): List of matching instance names

creopyson.familytable.**list_tree**(*client*, *file_=None*, *erase=None*)
Get a hierarchical structure of a nested family table.
Args:
client (obj): creopyson Client.
***file_* (str, optional):** File name. Defaults is currently active model.
erase (boolean, optional): Erase model and non-displayed models afterwards. Defaults is *False*.
Returns:
(list:str):

List of child instances

total (int): Count of all child instances including their decendants.

children (list:dict):

name (str): Name of the family table instance.

total (int): Count of all child instances including their decendants.

children (list:dict): List of child instances.

creopyson.familytable.**replace**(*client*, *cur_model*, *new_inst*, *file_=None*, *cur_inst=None*, *path=None*)
Replace a model in an assembly with another inst in the same family table.

You must specify either cur_inst or path.
Args:
client (obj): creopyson Client.
cur_model (str): Generic model containing the instances.
new_inst (str): New instance name.
***file_* (str, optional):** File name (usually an assembly). Defaults is currently active model.
cur_inst (str): Instance name to replace. Defaults to None.

path (list:int, optional): Path to component to replace. Defaults to None.

Returns: None

creopyson.familytable.**set_cell**(*client*, *instance*, *colid*, *value*, *file_=None*)

Set the value of one cell of a family table.

Args:

client (obj): creopyson Client.

instance (str): Family Table instance name.

colid (str): Column ID.

value (depends on data type): Cell value.

***file_* (str, optional):** File name (usually an assembly). Defaults is currently active model.

Returns: None

4.1.9 creopyson.feature module

Feature module.

creopyson.feature.**delete**(*client*, *name=None*, *file_=None*, *status=None*, *type_=None*, *clip=None*)

Delete one or more features that match criteria.

Args:

client (obj): creopyson Client.

name (str|list:str, optional): Dimension name, (wildcards allowed: True); if empty then all features are listed.

***file_* (str, optional):** Model name (wildcards allowed: True). Defaults is current active model.

status (str, optional): Feature status pattern (wildcards allowed: True). Defaults: All feature statuses. Valid values: ACTIVE, INACTIVE, FAMILY_TABLE_SUPPRESSED, SIMP_REP_SUPPRESSED, PROGRAM_SUPPRESSED, SUPPRESSED, UNREGENERATED

***type_* (str, optional):** Feature type pattern (wildcards allowed: True). Defaults: All feature types.

clip (boolean, optional): Whether to clip-delete ANY features from this feature through the end of the structure. Defaults is False.

Raises: ValueError: status value is incorrect.

Returns: None

creopyson.feature.**delete_param**(*client*, *name=None*, *file_=None*, *param=None*)

Delete a feature parameter.

Args:

client (obj): creopyson Client.

name (str, optional): Parameter name (wildcards allowed: True). Defaults: All parameter names.

***file_* (str, optional):** Model name. Defaults is current active model.

param (str, optional): Parameter name (wildcards allowed: True). Defaults: All parameter names.

Returns: None

creopyson.feature.**list_**(*client*, *file_=None*, *name=None*, *status=None*, *type_=None*, *paths=None*, *no_datum=None*, *inc_unnamed=None*, *no_comp=None*)

List feature parameters that match criteria.

Will only list parameters on visible features.

Args:

client (obj): creopyson Client.

***file_* (str, optional):** File name. Defaults is the currently active model.

name (str, optional): Feature name (wildcards allowed: True). Defaults: All features are listed.

status (str, optionnal): Feature status pattern. Defaults: All feature statuses. Valid values: ACTIVE, INACTIVE, FAMILY_TABLE_SUPPRESSED, SIMP_REP_SUPPRESSED, PROGRAM_SUPPRESSED, SUPPRESSED, UNREGENERATED.

***type_* (str, optional):** Feature type patter (wildcards allowed: True). Defaults: All feature types.

paths (boolean, optionnal): Whether feature ID and feature number are returned with the data Default: False.

no_datum (boolean, optional): Whether to exclude datum-type features from the list; these are COORD_SYS, CURVE, DATUM_AXIS, DATUM_PLANE, DATUM_POINT, DATUM_QUILT, and DATUM_SURFACE features. Defaults is False.

inc_unnamed (boolean, optional): Whether to include unnamed features in the list. Defaults is False.

no_comp (boolean, optional): Whether to include component-type features in the list. Defaults is False.

Raises: ValueError: status value is incorrect.

Returns:

(list:dict): List of parameter information.

name (str): Parameter nam.

value (depends on data type): Parameter value.

type (string): Data type. Valid values: STRING, DOUBLE, INTEGER, BOOL, NOTE.

designate (boolean): Value is designated.

encoded (boolean): Value is Base64-encoded.

owner_name (str): Owner Name.

owner_id (int): Owner ID.

owner_type (str): Owner type.

creopyson.feature.**list_group_features**(*client*, *group_name*, *type_=None*, *file_=None*)

List features in a Creo Group.

Args:

client (obj): creopyson Client.

group_name (str): Group name.

***type_* (str, optional):** Feature type patter (wildcards allowed: True). Defaults: All feature types.

***file_* (str, optional):** File name. Defaults is the currently active model.

Returns: (list:dict): List of feature information

creopyson.feature.**list_params**(*client*, *file_=None*, *name=None*, *type_=None*, *no_datum=None*, *inc_unnamed=None*, *no_comp=None*, *param=None*, *value=None*, *encoded=None*)

List feature parameters that match criteria.

Args:

client (obj): creopyson Client.

***file_* (str, optional):** File name. Defaults is the currently active model.

name (str|int, optional): str: Feature name (wildcards allowed: True). int: Feature ID. Defaults: All features are listed.

***type_* (str, optional):** Feature type patter (wildcards allowed: True). Defaults: All feature types.

no_datum (boolean, optional): Whether to exclude datum-type features from the list; these are COORD_SYS, CURVE, DATUM_AXIS, DATUM_PLANE, DATUM_POINT, DATUM_QUILT, and DATUM_SURFACE features. Defaults is False.

inc_unnamed (boolean, optional): Whether to include unnamed features in the list. Defaults is False.

no_comp (boolean, optional): Whether to include component-type features in the list. Defaults is False.

param (str|list:str, optional): Parameter name; (wildcards allowed: True) if empty all parameters are listed.

value (str, optional): Parameter value filter (wildcards allowed: True). Defaults is no filter.

encoded (boolean, optional): Whether to return the values Base64-encoded. Defaults is False.

Returns:

(list:dict): List of parameter information.

name (str): Parameter nam.

value (depends on data type): Parameter value.

type (string): Data type. Valid values: STRING, DOUBLE, INTEGER, BOOL, NOTE.

designate (boolean): Value is designated.

encoded (boolean): Value is Base64-encoded.

owner_name (str): Owner Name.

owner_id (int): Owner ID.

owner_type (str): Owner type.

description (str): List of parameter information.

creopyson.feature.**list_pattern_features**(*client*, *patter_name*, *type_=None*, *file_=None*)

List features in a Creo Pattern.

Args:

client (obj): creopyson Client.
patter_name (str): Pattern name.
***type_* (str, optional):** Feature type patter (wildcards allowed: True). Defaults: All feature types.
***file_* (str, optional):** File name. Defaults is the currently active model.

Returns: (list:dict): List of feature information

creopyson.feature.**list_selected**(*client*)

List the currently selected features in Creo

Returns:

(list): List of feature informations.

```
[
    { 'file' : model name (str)
      'name' : feature name (str)
      'status' : feature status (str)
      'type' : feature type (str)
      'feat_id' : feature ID (int)
      'feat_number' : feature number (int)
      'path' : feature's component path (list of ints)
    },
]
```

creopyson.feature.**param_exists**(*client*, *file_=None*, *name=None*, *param=None*)

Check whether parameter(s) exists on a feature.

Args:

client (obj): creopyson Client.
***file_* (str, optional):** File name. Defaults is the currently active model.
name (str, optional): Parameter name (wildcards allowed: True). Defaults: All parameter names.
param (str|list:str, optional): Parameter name; (wildcards allowed: True) if empty all parameters are listed.

Returns: (boolean): Whether the parameter exists on the model

creopyson.feature.**rename**(*client*, *name*, *new_name*, *file_=None*)

Rename a feature.

Args:

client (obj): creopyson Client.

name (str|int, optional): Feature name (str) or Feature ID (int).
new_name (str): New name for the feature.
***file_* (str, optional):** File name. Defaults is the currently active model.

Returns: None

creopyson.feature.**resume**(*client*, *file_=None*, *name=None*, *status=None*, *type_=None*, *with_children=None*)

Resume one or more features that match criteria.

Will only resume visible features.

Args:

client (obj): creopyson Client.
***file_* (str, optional):** File name (wildcards allowed: True). Defaults is the currently active model.
name (int|str|list:str, optional): Feature name or Feature ID, (wildcards allowed: True); if empty then all features are resumed. int => Feat_ID str => name list:str => names
status (str, optional): Feature status pattern. Defaults: All feature statuses. Valid values: ACTIVE, INACTIVE, FAMILY_TABLE_SUPPRESSED, SIMP_REP_SUPPRESSED, PROGRAM_SUPPRESSED, SUPPRESSED, UNREGENERATED
***type_* (str, optional):** Feature type pattern (wildcards allowed: True). Defaults: All feature types.
with_children (boolean, optional): Whether to resume any child features of the resumed feature. Defaults is False.

Raises: ValueError: status value is incorrect.

Returns: None

creopyson.feature.**set_param**(*client*, *param*, *file_=None*, *name=None*, *type_=None*, *value=None*, *encoded=None*, *designate=None*, *description=None*, *no_create=None*)

Set the value of a feature parameter.

Will only set parameters on visible features.

Args:

client (obj): creopyson Client.
param (str): Parameter name.
***file_* (str, optional):** File name (wildcards allowed: True). Defaults is the currently active model.
name (str, optional): Feature name. Defaults: All features are updated.
***type_* (str, optional):** Parameter data type. Defaults is True. Valid values: STRING, DOUBLE, INTEGER, BOOL, NOTE.
value (depends on data type, optional): Parameter value. Defaults: Clears the parameter value if missing.
encoded (boolean, optional): Value is Base64-encoded. Defaults is False.
designate (boolean, optional): Set parameter to be designated/not designated, blank=do not set. Defaults is *blank*.
description (str, optionnal): Parameter description. If missing, leaves the currect description in place.
no_create (boolean, optional): If parameter does not already exist, do not create it. Defaults is False.

Returns: None

creopyson.feature.**suppress**(*client*, *file_=None*, *name=None*, *status=None*, *type_=None*, *clip=None*, *with_children=None*)

Suppress one or more features that match criteria.

Will only suppress visible features.

Args:

client (obj): creopyson Client.
***file_* (str, optional):** File name (wildcards allowed: True). Defaults is the currently active model.
name (int|str|list:str, optional): Feature name or Feature ID, (wildcards allowed: True); if empty then all features are suppressed. int => Feat_ID str => name list:str => names
status (str, optional): Feature status pattern. Defaults: All feature statuses. Valid values: ACTIVE, INACTIVE, FAMILY_TABLE_SUPPRESSED, SIMP_REP_SUPPRESSED, PRO-

GRAM_SUPPRESSED, SUPPRESSED, UNREGENERATED

***type_* (str, optional):** Feature type pattern (wildcards allowed: True). Defaults: All feature types.

clip (boolean, optional): Whether to clip-suppress ANY features from this feature through the end of the structure. Defaults is True.

with_children (boolean, optional): Whether to suppress any child features of the suppressed feature. Defaults is True.

Raises: ValueError: status value is incorrect.

Returns: None

creopyson.feature.**user_select_csys**(*client*, *file_=None*, *max_=None*)

Prompt the user to select one or more coordinate systems.

and return their selections.

Args:

client (obj): creopyson Client.

***file_* (str, optional):** File name. Defaults is the currently active model.

***max_* (int, optional):** The maximum number of dimensions that the user can select. Defaults is *1*.

Returns:

(list:dict):

List of feature information.

name (str): Feature name.

type (string): Feature type.

status (str): Feature status. Valid values: ACTIVE, INACTIVE, FAMILY_TABLE_SUPPRESSED, SIMP_REP_SUPPRESSED, PROGRAM_SUPPRESSED, SUPPRESSED, UNREGENERATED.

feat_id (int): Feature ID.

file (str): File name containing the feature.

path (list:int): Component Path to feature (optionnal)

4.1.10 creopyson.file module

File module.

creopyson.file.**assemble**(*client*, *file_*, *dirname=None*, *generic=None*, *into_asm=None*, *path=None*, *ref_model=None*, *transform=None*, *constraints=None*, *package_assembly=None*, *walk_children=None*, *assemble_to_root=None*, *suppress=None*)

Assemble a component into an assembly.

Args:

client (obj): creopyson Client.

***file_* (str):** File name component.

dirname (str, optional): Diretory name. Defaults is Creo's current working directory.

generic (str, optional): Generic model name (if file name represents an instance). Defaults is generic model name (if file name represents an instance).

into_asm (str, optional): Target assembly. Defaults is currently active model.

path (list:int, optional): Path to a component that the new part will be constrained to. Defaults to None.

ref_model (str, optional): Reference model that the new part will be constrained to; only used if path is not given. If there are multiple of this model in the assembly, the component will be assembled multiple times, once to each occurrence. Defaults to None.

transform (obj:JLTransform, optional): Transform structure for the initial position and orientation of the new component; only used if there are no constraints, or for certain constraint types. Defaults to None.

constraints (obj_array:JLConstraint, optional): Assembly constraints. Defaults to None.

package_assembly (bool, optional): Whether to package the component to the assembly; only used if there are no constraints specified. Defaults is If there are no constraints, then the user will be prompted to constrain the component through the Creo user interface.

walk_children (bool, optional): Whether to walk into subassemblies to find reference models to constrain to. Defaults to None.

assemble_to_root (bool, optional): Whether to always assemble to the root assembly, or assemble to the subassembly containing the reference path/model. Defaults to None.

suppress (bool, optional): Whether to suppress the components immediately after assembling them. Defaults to None.

Returns:

(dict):

dirname (str): Directory name of component.

files (list:str): File name of component.

revision (int): Revision of file that was opened; if more than one file was opened, this field is not returned.

featureid (int): Last Feature ID of component after assembly.

`creopyson.file.`**`backup`**(*client*, *target_dir*, *file_=None*)

Backup a model.

Args:

client (obj): creopyson Client.

target_dir (str): Target directory name.

***file_* (str, optional):** File name. Defaults is currently active model.

Returns: None

`creopyson.file.`**`close_window`**(*client*, *file_=None*)

Close the window containing a model.

Args:

client (obj): creopyson object.

***file_* (str, optional):** File name. Defaults is currently active model.

Returns: None

`creopyson.file.`**`delete_material`**(*client*, *material*, *file_=None*)

Delete a material from a part.

Args:

client (obj): creopyson object

material (str): Material name.

***file_* (str, optional):** File name. (Wildcards allowed: True). Defaults is currently active model.

`creopyson.file.`**`display`**(*client*, *file_*, *activate=None*)

Display a model in a window.

Args:

client (obj): creopyson object.

***file_* (str):** File name

activate (bool, optional): Activate the model after displaying. Defaults is True.

Returns: None

`creopyson.file.`**`erase`**(*client*, *file_=None*, *erase_children=None*)

Erase one or more models from memory.

Args:

client (obj): creopyson Client.
***file_* (str|list:str, optional):** File name or List of file names; (Wildcards allowed: True). if empty all models in memory are erased.
erase_children (bool, optional): Erase children of the models too. Defaults is False.
Returns: None

creopyson.file.**erase_not_displayed**(*client*)
Erase all non-displayed models from memory.
Args: client (obj): creopyson Client.
Returns: None

creopyson.file.**exists**(*client*, *file_*)
Check whether a model exists in memory.
Args: client (obj): creopyson Client. *file_* (str): File name.
Returns: (bool): Whether the file is open in Creo.

creopyson.file.**get_accuracy**(*client*, *file_=None*)
Get a solid's accuracy.

If the model has no accuracy value, this function will return null.
Args:
client (obj): creopyson Client.
***file_* (str, optional):** File name. Defaults is current active model
Returns:
(list):

accuracy (float): Accuracy value.

relative (bool): True = relative; False = absolute accuracy.

creopyson.file.**get_active**(*client*)
Get the active model from Creo.
Args: client (obj): creopyson Client.
Returns:
(dict): dirname (str): Directory name of current model. file (str): File name of current model.

creopyson.file.**get_cur_material**(*client*, *file_=None*)
Get the current material for a part.

Note: This is the same as 'get_cur_material_wildcard' but this function does not allow wildcards on the part name. They are separate functions because the return structures are different. This function is retained for backwards compatibility.
Args:
client (obj): creopyson Client.
***file_* (str, optional):** Part name. Defaults to None is current active model.
Returns:
str: Current material for the part, may be null if there is no current material.

creopyson.file.**get_cur_material_wildcard**(*client*, *file_=None*, *include_non_matching_parts=False*)
Get the current material for a part or parts.

Note: This is the same as 'get_cur_material' but this function allows wildcards on the part name. They are separate functions because the return structures are different.
Args:
client (obj): creopyson Client.
***file_* (str, optional):** Part name. Defaults to None is current active model.
include_non_matching_parts (bool, optionnal): Whether to include parts that match the part name pattern but don't have a current material. Defaults to False.
Returns:

list: A list of part and current-material pairs.

creopyson.file.**get_fileinfo**(*client*, *file_=None*)

Open one or more files in memory or from the drive.

Args:

client (obj): creopyson Client.

***file_* (str, optional):** File name. Defaults is currently active model.

Returns:

(dict):

dirname (str): Directory name of the file.

file (str): File name.

revision (int): Revision number of file. (Not availlable if the file is in Windchill)

creopyson.file.**get_length_units**(*client*, *file_=None*)

Get the current length units for a model.

Args:

client (obj): creopyson Client.

***file_* (str, optional):** File name. Defaults is currently active model.

Returns: (str): Length units.

creopyson.file.**get_mass_units**(*client*, *file_=None*)

Get the current mass units for a model.

Args:

client (obj): creopyson Client.

***file_* (str, optional):** File name. Defaults is currently active model.

Returns: (str): mass units.

creopyson.file.**get_transform**(*client*, *asm=None*, *path=None*, *csys=None*)

Get the 3D transform for a component in an assembly.

Args:

client (obj): creopyson Client

asm (str, optional): Assembly name. Defaults is currently active model.

path (list:int, optional): Path to a component in the assembly. Defaults is the transform is calculated for the assembly itself.

csys (str, optional): Coordinate system on the component to calculate the transform for. Defaults is the component's default coordinate system.

Returns: (obj:JLTransform): The 3D transform from the assembly to the component's coordinate system.

creopyson.file.**has_instances**(*client*, *file_=None*)

Check whether a model has a family table.

Args:

client (obj): creopyson Client.

***file_* (str, optional):** File name. Defaults is currently active model.

Returns: (bool): Whether the file has a family table.

creopyson.file.**is_active**(*client*, *file_*)

Check whether a model is the active model.

Args: client (obj): creopyson Client. *file_* (str): File name.

Returns: (bool): Whether the file is the currently active model.

creopyson.file.**list_**(*client*, *file_=None*)

Get a list of files in the current Creo session that match patterns.

Args:

client (obj): creopyson Client.

***file_* (str|list:str, optional):** File name or List of file names;

Returns: (list:str) List of files.

creopyson.file.**list_instances**(*client*, *file_=None*)

List instances in a model's family table.

Args:

client (obj): creopyson Client.

***file_* (str, optional):** File name. Defaults is currently active model.

Returns:

(dict): dirname (str): Directory name of the file. generic (str): Generic name. files (list:str): List of model names in the table.

creopyson.file.**list_materials**(*client*, *file_=None*, *material=None*)

List materials on a part.

Args:

client (obj): creopyson Client.

***file_* (str, optional):** File name. Defaults is currently active model.

material (str, optional): Material name pattern. Wildcards allowed. Defaults to None is all materials.

Returns: list: List of materials in the part.

creopyson.file.**list_materials_wildcard**(*client*, *file_=None*, *material=None*, *include_non_matching_parts=False*)

List materials on a part or parts.

Args:

client (obj): creopyson Client.

***file_* (str, optional):** File name. Defaults is currently active model.

material (str, optional): Material name pattern. Wildcards allowed. Defaults to None is all materials.

include_non_matching_parts (bool, optional): Whether to include parts that match the part name pattern but don't have any materials matching the material pattern. Defaults to False.

Returns:

list: A list of part and material pairs. If a part has more than one material, it will have multiple entries in this array.

creopyson.file.**list_simp_reps**(*client*, *file_=None*, *rep=None*)

List simplified reps in a model.

Args:

client (obj): creopyson Client

***file_* (str, optional):** File name. Defaults is currently active model.

rep (str, optional): Simplified rep name pattern (wildcards_allowed: True). Defaults is all simplified reps.

Returns:

(dict): rep (str): Simplified rep name. reps (list:str): Simplified reps names.

creopyson.file.**load_material_file**(*client*, *material*, *dirname=None*, *file_=None*)

Load a new material file into a part or parts.

Note: If 'material' has a file extension, it will be removed before the material is loaded.

Args:

client (obj): creopyson Client

material (str): Material name

dirname (str, optional): Directory name containing the material file. Default is Creo's 'pro_material_dir' config setting, or search path, or current working directory

***file_* (str, optional):** File name. Wildcards allowed. Defaults is currently active model.

Returns:

list: List of files impacted.

creopyson.file.**massprops**(*client*, *file_=None*)

Get mass property information about a model.

Notes: PTC's description of coord_sys_inertia: "The inertia matrix with respect to coordinate frame:(element ij is the integral of x_i x_j over the object)". PTC's description of coord_sys_inertia_tensor: "The inertia tensor

with respect to coordinate frame: CoordSysInertiaTensor = trace(CoordSysInertia) * identity - CoordSysInertia".

Args:

client (obj): creopyson Client.

***file_* (str, optional):** File name. Defaults is currently active model.

Returns:

(dict):

volume (float): Model volume.

mass (float): Model mass.

density (float): Model density.

surface_area (float): Model surface area.

ctr_grav_inertia_tensor (object:JLInertia): Model's Inertia Tensor translated to center of gravity.

coord_sys_inertia (object:JLInertia): Model's Inertia Matrix with respect to the coordinate frame.

coord_sys_inertia_tensor (object:JLInertia): Model's Inertia Tensor with respect to the coordinate frame.

ctr_grav (object:JLPoint): Model's center of gravity.

creopyson.file.**open_**(*client*, *file_*, *dirname=None*, *generic=None*, *display=None*, *activate=None*, *new_window=None*, *regen_force=None*)

Open one or more files in memory or from the drive.

note: if you open more than one file, it will only put file in your session you won't be able to display more than one file at once.

Args:

client (obj): creopyson Client.

***file_* (str|list:str):** File name or List of file names;

dirname (str, optional): Directory name. Defaults is Creo's current working directory.

generic (str, optional): Generic model name (if file name represents an instance). Defaults to None.

display (bool, optional): Display the model after opening. Defaults is True.

activate (bool, optional): Activate the model after opening. Defaults is True.

new_window (bool, optional): Open model in a new window. Defaults is False.

regen_force (bool, optional): Force regeneration after opening. Defaults is False.

Returns:

(dict):

dirname (str): Directory name of opened file(s).

files (list:str): File names that were opened.

revision (int): Revision of file that was opened; if more than one file was opened, this field is not returned.

creopyson.file.**open_errors**(*client*, *file_=None*)

Check whether Creo errors have occurred opening a model.

Args:

client (obj): creopyson Client.

***file_* (str, optional):** File name. Defaults is currently active model.

Returns: (bool): Whether errors exist in Creo.

creopyson.file.**postregen_relations_get**(*client*, *file_=None*)

Get post-regeneration relations for a model.

Args:

client (obj): creopyson Client.
***file_* (str, optional):** File name. Defaults is currently active model.
Returns: (list:str): Exported relations text, one entry per line.

creopyson.file.**postregen_relations_set**(*client*, *file_=None*, *relations=None*)
Set post-regeneration relations for a model.
Args:
client (obj): creopyson Client.
***file_* (str, optional):** File name. Defaults is currently active model.
relations (list:str, optional): Relations text to import, one line per entry. Clear the relations if missing.
Retunrs: None

creopyson.file.**refresh**(*client*, *file_=None*)
Refresh the window containing a model.
Args:
client (obj): creopyson Client.
***file_* (str, optional):** File name. Defaults is currently active model.
Returns: None

creopyson.file.**regenerate**(*client*, *file_=None*, *display=None*)
Regenerate one or more models.
Args:
client (obj): creopyson Client.
***file_* (str|list:str, optional):** File name or List of file names; Defaults is currently active model
display (bool, optional): Display the model before regenerating. Defaults is False.
Returns: None

creopyson.file.**relations_get**(*client*, *file_=None*)
Get relations for a model.
Args:
client (obj): creopyson Client.
***file_* (str, optional):** File name. Defaults is currently active model.
Returns: (list:srt): Exported relations text, one entry per line.

creopyson.file.**relations_set**(*client*, *file_=None*, *relations=None*)
Set relations for a model.
Args:
client (obj): creopyson Client.
***file_* (str, optional):** File name. Defaults is currently active model.
relations (list:str, optional): Relations text to import, one line per entry. Clear the relations if missing.
Returns: None

creopyson.file.**rename**(*client*, *new_name*, *file_=None*, *onlysession=None*)
Rename a model.
Args:
client (obj): creopyson Client.
new_name (str): New file name.
***file_* (str, optional):** File name. Defaults is currently active model.
onlysession (bool, optional): Modify only in memory, not on disk. Defaults is False.
Returns: (str): The new model name.

creopyson.file.**repaint**(*client*, *file_=None*)
Repaint the window containing a model.
Args:
client (obj): creopyson Client.
***file_* (str, optional):** File name. Defaults is currently active model.
Returns: None

creopyson.file.**save**(*client*, *file_=None*)

Save one or more models.

Args:

client (obj): creopyson Client.

***file_* (str|list:str, optional):** File name or List of file names; Defaults is currently active model.

Returns: None

creopyson.file.**set_cur_material**(*client*, *material*, *file_=None*)

Set the current material for a part or parts.

Args:

client (obj): creopyson Client.

material (str): Material name.

***file_* (str, optional):** Part name. Wildcard allowed. Defaults is currently active model.

Returns:

list: list of impacted files.

creopyson.file.**set_length_units**(*client*, *units*, *file_=None*, *convert=None*)

Set the current length units for a model.

This will search the model's available Unit Systems for the first one which contains the given length unit.

Args:

client (obj): creopyson Client.

units (str): New length units.

***file_* (str|list:str, optional):** File name or List of file names; Defaults is currently active model.

convert (bool, optional): Whether to convert the model's length values to the new units (True) or leave them the same value (False). Defaults is True.

Returns: None

creopyson.file.**set_mass_units**(*client*, *units*, *file_=None*, *convert=None*)

Set the mass units for a model.

This will search the model's available Unit Systems for the first one which contains the given mass unit.

Args:

client (obj): creopyson Client.

units (str): New mass units.

***file_* (str|list:str, optional):** File name or List of file names; Defaults is currently active model.

convert (bool, optional): Whether to convert the model's mass values to the new units (True) or leave them the same value (False). Defaults is True.

Returns: None

4.1.11 creopyson.geometry module

Geometry module.

creopyson.geometry.**bound_box**(*client*, *file_=None*)

Get the bounding box for a model.

Args:

client (obj): creopyson Client.

***file_* (str, optional):** File name. Defaults is current active model.

Returns:

(dict): xmin (float): Minimum X-coordinate of model. xmax (float): Maximum X-coordinate of model. ymin (float): Minimum Y-coordinate of model. ymax (float): Maximum Y-coordinate of model zmin (float): Minimum Z-coordinate of model. zmax (float): Maximum Z-coordinate of model

creopyson.geometry.**get_edges**(*client*, *surface_ids*, *file_=None*)

Get the list of edges for a model for given surfaces.

Args:

client (obj): creopyson Client.
surface_ids (list:int): List of surface IDs.
***file_* (str, optional):** File name. Defaults is current active model
Returns:
(list:dict):
surface_id (int): Surface ID.
traversal (string): Traversal type. Valid values: internal, external.
edglist (list:dict):
Information about an edge.
edgeid (integer): Edge ID.
length (float): Edge length.
start (list:dict):
A 3D coordinate.
x (float): X-coordinate.
y (float): Y-coordinate.
z (float): Z-coordinate.
end (list:dict):
A 3D coordinate.
x (float): X-coordinate.
y (float): Y-coordinate.
z (float): Z-coordinate.

creopyson.geometry.**get_surfaces**(*client*, *file_=None*)
Get the list of surfaces for a model.
Args:
client (obj): creopyson Client.
***file_* (str, optional):** File name. Defaults is current active model
Returns:
(list:dict):
surface_id (int): Surface ID.
area (float): Surface area
min_extent (list:dict):
A 3D coordinate.
x (float): X-coordinate.
y (float): Y-coordinate.
z (float): Z-coordinate.
max_extent (list:dict):
A 3D coordinate.
x (float): X-coordinate.
y (float): Y-coordinate.

z (float): Z-coordinate.

4.1.12 creopyson.interface module

Interface module.

Export 3D pdf, file (step, iges, dxf, etc...), picture, plot Run mapkey Import/Export program (pls, als)

creopyson.interface.**export_3dpdf**(*client*, *file_=None*, *filename=None*, *dirname=None*, *height=None*, *width=None*, *dpi=None*, *use_drawing_settings=None*, *sheet_range='all'*)

Export a model to a 3D PDF file.

Args:

client (obj): creopyson Client.

***file_* (str, optional):** Model name. Defaults is current active model.

filename (str, optional): Destination file name. May also contain a path to the file. Defaults is the model name with the appropriate file extension, in Creo's working directory.

dirname (str, optional): Destination directory. Defaults is Creo's current working directory.

height (float, optional): PDF image height. Defaults is Creo default export height.

width (float, optional): PDF image width. Defaults is Creo default export width.

dpi (int, optional): PDF Image DPI. Default is Creo default export DPI.

use_drawing_settings (boolean, optional): Whether to use special settings for exporting drawings. Defaut is False.

sheet_range (string): Range of drawing sheets to export. Default vale is "all". Valid values: "all", "current", range of sheet numbers (ex: "1,3-4")

Returns:

dict: dirname (str): Directory of the output file filename (str): Name of the output file

creopyson.interface.**export_file**(*client*, *file_type*, *file_=None*, *filename=None*, *dirname=None*, *geom_flags=None*, *advanced=None*)

Export a model to a file.

The geom_flags option only applies to IGES and STEP exports. Setting geom_flags to 'default' will cause it to check the Creo config option 'intf3d_out_default_option' for the setting The advanced option will cause the Export to use settings defined in the appropriate *export_profiles* Creo Config Option for the file type. The advanced option only applies to DXF, IGES and STEP exports. The advanced option will only work with Creo 4 M030 or later.

Args:

client (obj): creopyson Client.

file_type (str): File type. Valid values: "DXF", "IGES", "NEUTRAL", "PV", "STEP", "VRML".

***file_* (str, optional):** Model name. Defaults is current active model.

filename (str, optional): Destination file name. May also contain a path to the file. Defaults is the model name with the appropriate file extension, in Creo's working directory.

dirname (str, optional): Destination directory. Defaults is Creo's current working directory.

geom_flags (str, optional): Geometry type for the export. Defaults is *solids*. Valid values: solids, surfaces, wireframe, wireframe_surfaces, quilts, default.

advanced (Boolean, optional): Whether to use the newer Creo 4 file export function. Defaults is False.

Returns:

dict: dirname (str): Directory of the output file filename (str): Name of the output file

creopyson.interface.**export_image**(*client*, *file_type*, *file_=None*, *filename=None*, *height=None*, *width=None*, *dpi=None*, *depth=None*)

Export a model to an image file.

Args:

client (obj): creopyson Client.

file_type (str): Image Type. Valid values: BMP, EPS, JPEG, TIFF.

***file_* (str, optional):** Model name. Defaults is current active model.
filename (str, optional): Destination file name. May also contain a path to the file. Defaults is the model name with the appropriate file extension, in Creo's working directory.
height (float, optional): Image height. Defaults is *10.0*.
width (float, optional): Image width. Defaults is *7.5*.
dpi (int, optional): Image DPI. Defaults is *100*. Valid values: 100, 200, 300, 400, 500, 600.
depth (int, optional): Image depth. Defaults is *24*. Valid values: 8, 24.

Returns:
dict: dirname (str): Directory of the output file filename (str): Name of the output file

creopyson.interface.**export_pdf**(*client*, *file_=None*, *filename=None*, *dirname=None*, *height=None*, *width=None*, *dpi=None*, *use_drawing_settings=None*, *sheet_range='all'*)

Export a model to a PDF file.

When use_drawing_settings is true, the Font Stroke option will be set to Stroke All Fonts, and the Color Depth option will be set to Grayscale.

Args:
client (obj): creopyson Client.
***file_* (str, optional):** Model name. Defaults is current active model.
filename (str, optional): Destination file name. May also contain a path to the file. Defaults is the model name with the appropriate file extension, in Creo's working directory.
dirname (str, optional): Destination directory. Defaults is Creo's current working directory.
height (float, optional): PDF image height. Defaults is Creo default export height.
width (float, optional): PDF image width. Defaults is Creo default export width.
dpi (int, optional): PDF Image DPI. Default is Creo default export DPI.
use_drawing_settings (boolean, optional): Whether to use special settings for exporting drawings. Defaut is False.
sheet_range (string): Range of drawing sheets to export. Default vale is "all". Valid values: "all", "current", range of sheet numbers (ex: "1,3-4")

Returns:
dict: dirname (str): Directory of the output file filename (str): Name of the output file

creopyson.interface.**export_program**(*client*, *file_=None*)

Export a model's program to a file.

Args:
client (obj): creopyson Client.
***file_* (str, optional):** Model name. Defaults is current active model.

Returns:
dict: dirname (str): Directory of the output file filename (str): Name of the output file

creopyson.interface.**import_file**(*client*, *filename*, *file_type=None*, *dirname=None*, *new_name=None*, *new_model_type='asm'*)

Import a file as a model.

Note: This function will not automatically display or activate the imported model. If you want that, you should take the file name returned by this function and pass it to file:open. Users of the old import_pv function should start using this function instead.

Args:
client (obj): creopyson Client.
filename (str): Source file name.
file_type (str, optional): File type. Valid values: "IGES", "NEUTRAL", "PV", "STEP". Defaults to None. Will analyse filename extension *.igs*|.iges* => IGES *.stp*|.step* => STEP *.neu* => NEUTRAL *.pvz* => PV
dirname (str, optional): Source directory. Defaults is Creo's current working directory.
new_name (str, optional): New model name. Any extension will be stripped off and replaced with one based on new_model_type. Defaults to None is the name of the file with an extension based on

new_model_type..

new_model_type (str, optional): New model type. Valid values: "asm", "prt" Defaults to "asm".

Returns: str: Name of the model imported

creopyson.interface.**import_program**(*client*, *file_=None*, *filename=None*, *dirname=None*)

Import a program file for a model.

Cannot specify both file and filename parameters.

Args:

client (obj): creopyson Client.

***file_* (str, optional):** Destination Model name. Defaults is currently active model, or the model for the filename parameter if given.

filename (str, optional): Source file name. Default is the model name with the appropriate file extension.

dirname (str, optional): Source directory. Defaults is Creo's current working directory.

Returns: str: Name of the model updated

creopyson.interface.**mapkey**(*client*, *script*, *delay=0*)

Run a Mapkey script in Creo.

Make sure to remove any *mapkey(continued)* clauses from the script argument. The tilde at the start of a line should occur immediately after the semicolon at the end of the previous line.

Args:

client (obj): creopyson Client.

script (str): The mapkey script to run.

delay (int): Amount of time to wait after starting the mapkey, in milliseconds. Default is 0.

Returns: None

creopyson.interface.**plot**(*client*, *file_=None*, *dirname=None*, *driver=None*)

Export a model plot.

Args:

client (obj): creopyson Client.

***file_* (str, optional):** Model name. Defaults is currently active model.

dirname (str, optional): Destination directory. Defaults is Creo's current working directory.

driver (str, optional): Driver export. Defaults is *POSTSCRIPT*. Valid values: POSTSCRIPT, JPEG, TIFF.

Returns:

dict: dirname (str): Directory of the output file filename (str): Name of the output file

4.1.13 creopyson.layer module

Layer module.

creopyson.layer.**delete**(*client*, *name=None*, *file_=None*)

Delete one or more layers.

Args:

client (obj): creopyson Client.

name (str, optional): Layer name (wildcards allowed: True). Defaults: All layers will be deleted.

***file_* (str, optional):** File name (wildcards alloawed: True). Defaults is current active model.

Returns: None

creopyson.layer.**exists**(*client*, *name=None*, *file_=None*)

Check whether layer(s) exists on a model.

Args:

client (obj): creopyson Client.

name (str, optional): Layer name (wildcards allowed: True). Defaults: All layers are listed.

***file_* (str, optional):** File name. Defaults is current active model.

Returns: (boolean): Whether the layer exists on the model.

creopyson.layer.**list_**(*client*, *name=None*, *file_=None*)
List layers that match criteria.
Args:
client (obj): creopyson Client.
name (str, optional): Layer name (wildcards allowed: True). Defaults: All layers are listed.
***file_* (str, optional):** File name. Defaults is current active model.
Returns:
(list:dict):
name (str): Layer name.
status (str): Layer status. Valid values: BLANK, DISPLAY, HIDDEN, NORMAL.
ID (int): Layer ID.

creopyson.layer.**show**(*client*, *name=None*, *file_=None*, *show_=None*)
how/Hide one or more layers.
Args:
client (obj): creopyson Client.
name (str, optional): Layer name (wildcards allowed: True). Defaults: All layers are listed.
***file_* (str, optional):** File name (wildcards allowed: True). Defaults is current active model.
***show_* (boolean, optional):** Whether to show or hide the layers. Defaults is True (show).
Returns: None

4.1.14 creopyson.note module

Note module.

creopyson.note.**copy**(*client*, *name*, *to_name=None*, *file_=None*, *to_file=None*)
Copy note to another in the same model or another model.
Args:
client (obj): creopyson Client.
name (str): Note name to copy (wildcards allowed: True).
to_name (str): Destination note. Defaults is the source note name
***file_* (str, optional):** Model name (wildcards allowed: True). Defaults is current active model.
to_file (str, optional): Destination model. Defaults is the source model.
Returns: None

creopyson.note.**delete**(*client*, *name*, *file_=None*)
Delete a model or drawing note.
Args:
client (obj): creopyson Client.
name (str): Note name (wildcards allowed: True).
***file_* (str, optional):** Model name (wildcards allowed: True). Defaults is current active model.
Returns: None

creopyson.note.**exists**(*client*, *file_=None*, *name=None*)
Check whether note(s) exists on a model.
Args:
client (obj): creopyson Client.
***file_* (str, optional):** Model name. Defaults is current active model.
name (str|list:str, optional): Note name; List of note names. if empty it checks for any note's existence.
Returns: (boolean): Whether the note exists on the model.

creopyson.note.**get**(*client*, *name*, *file_=None*)
Get the text of a model or drawing note.

Values will automatically be returned Base64-encoded if they are strings which contain Creo Symbols or other non-ASCII data.

Args:

client (obj): creopyson Client.

name (str): Note name.

***file_* (str, optional):** Model name. Defaults is current active model or drawing.

Returns:

(dict):

file (str): File name.

name (str): Note name.

encoded (boolean): Value is Base64-encoded.

url (str): "Note URL, if there is one.

location (obj:JLPoint): Note location in Drawing Units (drawing notes only)

creopyson.note.**list_**(*client*, *file_=None*, *name=None*, *value=None*, *get_expanded=None*, *select=False*)

Get a list of notes from one or more models.

Values will automatically be returned Base64-encoded if they are strings which contain Creo Symbols or other non-ASCII data.

Args:

client (obj): creopyson Client.

***file_* (str, optional):** Model name (wildcards allows: True). Defaults is current active model.

name (str|list:str, optional): Note name; List of note names. if empty all notes are listed.

value (str, optional): Parameter value filter (wildcards allows: True). Defaults is *no filter*.

get_expanded (boolean, optional): Whether to return text with parameter values replaced. Defaults is False.

select (boolean, optional): If true, the notes that are found will be selected in Creo. Defaults is False.

Returns:

(list:dict): name (str): Note name. value (str): Note text with parameters not expanded. value_expanded (str): Note text with parameters expanded. encoded (boolean): Value is Base64-encoded. location (jlpoint): 3D coordinate dict.

creopyson.note.**set_**(*client*, *name*, *file_=None*, *location=None*, *encoded=None*, *value=None*)

Set the text of a model or drawing note.

The location parameter can used to position a new note, or to change the position of an existing note. If the text contains Creo Symbols or other non-ASCII text, you must Base64-encode the value and set encoded to true You may be able to avoid Base64-encoding symbols by using Unicode for the binary characters, for example including \u0001#\u0002 in the value to insert a plus/minus symbol. Embed newlines in the value for line breaks.

Args:

client (obj): creopyson Client.

name (str): Note name (wildcards allowed: True).

***file_* (str, optional):** Model name. Defaults is current active model or drawing.

location (JLPoint): Coordinates for the note placement in Drawing Units. If missing and this is a new note, note will be placed at 0, 0.

encoded (boolean, optional): Whether the value is Base64-encoded. Defaults is False.

value (str, optional): Note text with parameters not expanded. Defaults to None: clears the note if missing.

Returns: None

4.1.15 creopyson.objects module

Objects Module.

creopyson.objects.**jlpoint**(*x*, *y*, *z*)

Return a 3D coordinate dict.

Args:

x (float): X-coordinate.
y (float): Y-coordinate.
z (float): Z-coordinate.

Returns: (dict): 3D coordinate object.

4.1.16 creopyson.parameter module

Parameter module.

creopyson.parameter.**copy**(*client*, *name*, *to_name*, *file_=None*, *to_file=None*, *designate=None*)

Copy parameter to another in the same model or another model.

Args:

client (obj): creopyson Client.
name (str): Parameter name to copy (wildcards allowed: True).
to_name (str): Destination parameter.
***file_* (str, optional):** Model name. Defaults is current active model.
to_file (str, optional): Destination model (wildcards allowed: True). Defaults is the source model.
designate (boolean, optional): Set copied parameter to be designated/not designated, blank=do not set. Defaults is *blank*.

Returns: None

creopyson.parameter.**delete**(*client*, *name*, *file_=None*)

Delete a parameter.

Args:

client (obj): creopyson Client.
name (str): Parameter name (wildcards allowed: True).
***file_* (str, optional):** Model name. Defaults is current active model.

Returns: None

creopyson.parameter.**exists**(*client*, *name=None*, *file_=None*)

Check whether parameter(s) exists on a model.

Args:

client (obj): creopyson Client.
name (str|list:str, optional): Parameter name; List of parameter names. if empty it checks for any parameter's existence.
***file_* (str, optional):** Model name. Defaults is current active model.

Returns: (boolean): Whether the parameter exists on the model.

creopyson.parameter.**list_**(*client*, *name=None*, *file_=None*, *encoded=None*, *value=None*)

Get a list of parameters from one or more models.

Args:

client (obj): creopyson Client.
name (str|list:str, optional): Parameter name; List of parameter names. if empty it checks for any parameter's existence.
***file_* (str, optional):** Model name. Defaults is current active model.
encoded (boolean, optional): Whether to return the values Base64-encoded. Defaults is False.
value (str, optional): Parameter value filter. Defaults is *no filter*.

Returns:

(list:dict): name (str): Parameter name. type (str): Parameter type. value (various): Parameter value # TODO designate (boolean): Whether the parameter is designated. description (str): Description. encoded (boolean): Whether the parameter is encoded. owner_name (str): File name.

creopyson.parameter.**set_**(*client*, *name*, *value=None*, *file_=None*, *type_=None*, *encoded=None*, *designate=None*, *description=None*, *no_create=None*)

Set the value of a parameter.

Args:

client (obj): creopyson Client.

name (str): Parameter name (wildcards allowed: True).

value (depends on data type, optional): Parameter value. Defaults to None. Clears the parameter value if missing.

***file_* (str, optional):** Model name. Defaults is current active model.

***type_* (str, optional):** Data type. Defaults is *STRING*. Valid values: STRING, DOUBLE, INTEGER, BOOL, NOTE.

encoded (boolean, optional): Whether the value is Base64-encoded. Defaults is False.

designate (boolean, optional): Set parameter to be designated/not designated, blank=do not set. Defaults is *blank*.

description (str, optional): Parameter description. If missing, leaves the current description in place.

no_create (boolean, optional): If parameter does not already exist, do not create it. Defaults is False.

Returns: None

creopyson.parameter.**set_designated**(*client*, *name*, *designate*, *file_=None*)

Set the designated state of a parameter.

Args:

client (obj): creopyson Client.

name (str): Parameter name (wildcards allowed: True).

designate (boolean): Set parameter to be designated/not designated.

***file_* (str, optional):** Model name. Defaults is current active model.

Returns: None

4.1.17 creopyson.server module

Server module.

creopyson.server.**pwd**(*client*)

Return the creoson server's execution directory.

Args:

client (obj): creopyson Client.

Raises: Warning: error message from creoson.

Returns: (str): Full name of working directory.

4.1.18 creopyson.view module

View module.

creopyson.view.**activate**(*client*, *name*, *file_=None*)

Activate a model view.

Args:

client (obj): creopyson Client.

name (str): View name.

***file_* (str, optional):** Model name. Defaults is current active model.

Returns: None

creopyson.view.**list_**(*client*, *file_=None*, *name=None*)
List views that match criteria.
Args:
client (obj): creopyson Client.
***file_* (str, optional):** Model name. Defaults is current active model.
name (str, optional): View name (wildcards allowed: True). Defaults is None: all views are listed.
Returns: (list:str): List of view names.

creopyson.view.**list_exploded**(*client*, *file_=None*, *name=None*)
List views that match criteria and are exploded.
Args:
client (obj): creopyson Client.
***file_* (str, optional):** Model name. Defaults is current active model.
name (str, optional): View name (wildcards allowed: True). Defaults is None: all views are listed.
Returns: (list:str): List of view names.

creopyson.view.**save**(*client*, *name*, *file_=None*)
Save a model's current orientation as a new view.
Args:
client (obj): creopyson Client.
name (str): View name.
***file_* (str, optional):** Model name. Defaults is current active model.
Returns: None

4.1.19 creopyson.windchill module

Windchill module.

Connect to Windchill server, use workspaces (create/delete/list) List files and checkout status.

creopyson.windchill.**authorize**(*client*, *user*, *password*)
Set user's Windchill login/password.
Args: client (obj): creopyson Client user (str): user name password (str): password
Returns: None

creopyson.windchill.**clear_workspace**(*client*, *workspace=None*, *filenames=None*)
Clear a workspace on the active server.
Args:
client (obj): creopyson Client
workspace (str, optionnal): Workspace name. Default is current workspace.
filenames (str|list, optionnal): List of files to delete from the workspace. Default: All files are deleted.
Returns: None

creopyson.windchill.**create_workspace**(*client*, *workspace*, *context_name*)
Create a workspace on the active server.
Args: client (obj): creopyson Client workspace (str): Workspace name context_name (str): Context name
Returns: None

creopyson.windchill.**delete_workspace**(*client*, *workspace*)
Delete a workspace on the active server.
Args: client (obj): creopyson Client workspace (str): Workspace name
Returns: None

creopyson.windchill.**file_checked_out**(*client*, *filename*, *workspace=None*)
Check whether a file is checked out in a workspace on the active server.
Args:
client (obj): creopyson Client

filename (str): File name
workspace (str, optionnal): Workspace name. Default is current workspace.
Returns: Boolean: Whether the file is checked out in the workspace.

creopyson.windchill.**get_workspace**(*client*)
Retrieve the name of the active workspace on the active server.
Args: client (obj): creopyson Client
Returns: str: Active Workspace name.

creopyson.windchill.**list_workspace_files**(*client*, *workspace=None*, *filename=None*)
Get a list of files in a workspace on the active server.
Args:
client (obj): creopyson Client
workspace (str, optionnal): Workspace name. Default is current workspace.
filename (str, optional): File name or search. Default is all files. ex: **.asm*, *screw_*.prt*
Returns: list: List of files in the workspace correspnding to the data.

creopyson.windchill.**list_workspaces**(*client*)
Get a list of workspaces the user can access on the active server.
Args: client (obj): creopyson Client
Returns: list: List of workspaces

creopyson.windchill.**server_exists**(*client*, *server_url*)
Check whether a server exists.
Args: client (obj): creopyson Client server_url (str): server URL or Alias
Returns: Boolean: Whether the server exists

creopyson.windchill.**set_server**(*client*, *server_url*)
Select a Windchill server.
Args: client (obj): creopyson Client server_url (str): server URL or Alias
Returns: None

creopyson.windchill.**set_workspace**(*client*, *workspace*)
Select a workspace on the active server.
Args: client (obj): creopyson Client workspace (str): Workspace name
Returns: None

creopyson.windchill.**workspace_exists**(*client*, *workspace*)
Check whether a workspace exists on the active server.
Args: client (obj): creopyson Client workspace (str): Workspace name
Returns: Boolean: Whether the workspace exists

4.1.20 Module contents

Top-level package for Creopyson.

CHAPTER 5

Contributing

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

You can contribute in many ways:

5.1 Types of Contributions

5.1.1 Report Bugs

Report bugs at https://github.com/Zepmanbc/creopyson/issues.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

5.1.2 Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with "bug" and "help wanted" is open to whoever wants to implement it.

5.1.3 Implement Features

Look through the GitHub issues for features. Anything tagged with "enhancement" and "help wanted" is open to whoever wants to implement it.

5.1.4 Write Documentation

Creopyson could always use more documentation, whether as part of the official Creopyson docs, in docstrings, or even on the web in blog posts, articles, and such.

5.1.5 Submit Feedback

The best way to send feedback is to file an issue at https://github.com/Zepmanbc/creopyson/issues.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

5.2 Get Started!

Ready to contribute? Here's how to set up *creopyson* for local development.

1. Fork the *creopyson* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/creopyson.git
```

3. Install your local copy into a virtualenv. Assuming you use pipenv:

```
$ cd creopyson/
$ pipenv install --dev
```

4. Create a branch for local development:

```
$ git checkout -b name-of-your-bugfix-or-feature
```

   Now you can make your changes locally.

5. When you're done making changes, check that your changes pass flake8 and the tests, including testing other Python versions with tox:

```
$ flake8 creopyson tests
$ python setup.py test # or pytest
$ tox
```

   To get flake8 and tox, just install them with *pipenv install -r requirements-dev.txt –dev*

6. Commit your changes and push your branch to GitHub:

```
$ git add .
$ git commit -m "Your detailed description of your changes."
$ git push origin name-of-your-bugfix-or-feature
```

7. Submit a pull request through the GitHub website.

5.3 Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.
2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in README.rst.
3. The pull request should work for Python 3.6 and 3.7, and for PyPy. Check https://travis-ci.org/Zepmanbc/creopyson/pull_requests and make sure that the tests pass for all supported Python versions.

5.4 Tips

To run a subset of tests:

```
$ pytest tests/test_creo.py
```

5.5 Deploying

A reminder for the maintainers on how to deploy. Make sure all your changes are committed (including an entry in HISTORY.rst). Then run:

```
$ bumpversion patch # possible: major / minor / patch
$ git tag -a vx.x.x -m "my version vx.x.x" # add the tag manualy if you don't use git-
↪flow
$ git push --tags
```

Travis will then deploy to PyPI if tests pass.

CHAPTER 6

Credits

6.1 Development Lead

- Benjamin C. <zepman@gmail.com>

6.2 Contributors

None yet. Why not be the first?

CHAPTER 7

History

7.1 0.7.6 (2023-02-26)

- **BugFix:**
  - Correction for *layer.delete* (issue#68)

7.2 Documentation update (2023-02-16)

Update for Creoson 2.8.2 release.

- No modification in creopyson, just documentation update, see Creoson's release note.

7.3 0.7.5 (2022-12-10)

- **BugFix:**
  - Add missing *location* argument on note_set (issue#62)

7.4 0.7.4 (2022-05-16)

Update for Creoson 2.8.1 release.

- **Add parameters:**
  - bom_get_paths : get_simpreps
- **Documentation update:**
  - creo_set_creo_version : Creo 8 support.

7.5 0.7.3 (2021-08-29)

- Add *is not None* almost everywhere to prevent *False* and *None* confusion
- Migrates to Github Workflow
- Remove pipenv stuff

7.6 0.7.2 (2021-04-01)

- change python_requires to ">=3.7"

7.7 0.7.1 (2021-03-23)

- **BugFix:**
  - *file_get_active* returns *{}* instead of *None* with Creoson 2.8.0

7.8 0.7.0 (2021-03-22)

Update for Creoson 2.8.0 release.

- **New functions:**
  - creo_set_creo_version (prevent Creo 7 problems)
  - feature_list_selected
- **Add parameters:**
  - feature_set_param: description
  - parameter_set: description
- **Documentation update:**
  - drawing_list: add *view_model*, *simp_rep* in Return
  - feature_list_param: add *description* in Return
  - feature_user_select_csys: add *file*, remove "feat_number* in Return object.
  - parameter_list: add *description* in Return
- **BugFix:** file_set_cur_material: now working with Creoson >2.8.0

7.9 0.6.2 (2021-02-17)

Bugfix:

- drawing.get_cur_model: correction data_key (issue#27)

7.10 0.6.1 (2021-01-30)

Bugfix:

- familytable.list_tree: correction data_key (issue#21)

Documentation update:

- Usage: Add vanilla Creoson
- Usage: Add logging
- Issues template: add Creo version

7.11 0.6.0 (2020-07-16)

Update for Creoson 2.7.0 release.

- **New functions:**
  - file_get_accuracy
- **Add parameters:**
  - interface_mapkey: delay
  - interface_export_pdf interface_export_pdf3d : sheet_range

7.12 0.5.2 (2020-07-09)

Documentation update

- usage: path slashes correction

Docstring correction

- drawing_list_models : correction (issue#18)

Bugfix

- file_get_transform : Does not return *transform* key (issue#17)

7.13 0.5.1 (2020-05-19)

Docstring updates:

- interface_import_file: PV extension correction.
- interface_export_image: add infos about valid values in docstrings for dpi and depth.

Features updates:

- interface_mapkey: remove extra white space in script string.
- connection_start_creo: add use_desktop param.

New Feature:

- Add **logging DEBUG** on request & response.

7.14 0.5.0 (2020-03-08)

Update for Creoson 2.6.0 release.

- **New functions:**
  - interface_import_file
- **Add parameters:**
  - bom_get_paths: add *has_simprep*
  - file_delete_material: *file* now allows wildcard
  - interface_export_file: add *NEUTRAL* to *file_type*
  - file_load_material: *file_* allows wildcard
- **New returns:**
  - file_massprops: add inertia matrices to output (*ctr_grav_inertia_tensor*, *coord_sys_inertia*, *coord_sys_inertia_tensor*)

7.15 0.4.3 (2020-03-07)

Update missing features from previous Creoson updates.

- **New Features:**
  - drawing_set_sheet_format
  - file_get_cur_material
  - file_get_cur_material_wildcard
  - file_list_materials
  - file_list_materials_wildcard
  - file_load_material_file
  - file_set_cur_material
- **New param:**
  - **note_list:**
    - add *select* param
    - add *location* in response

7.16 0.4.2 (2020-03-03)

Bugfix:

- feature_list params correction (ADD: status, paths, no_comp. REMOVE: param, value, encoded)
- feature_list_params params correction(inc_unnamed)
- feature_param_exists params correction (name)

- add test on *status* correct values in feature's functions (feature_delete, feature_list feature_resume, feature_suppress)

modify pipenv config for bleach security alert.

7.17 0.4.1 (2020-01-30)

Bugfix:

- view_list_exploded(): name param was in request even if empty (issue#4)
- start_creo(): path decomposition did not worked with Windows style (issue#5)
- geometry_get_surfaces(): wrong data_key waited in result, need *surflist* (issue#6)

7.18 0.4.0 (2019-10-12)

Update for Creoson 2.5.0 release.

- New functions:
  - file_delete_material
  - drawing_get_sheet_format
  - dimension_set_text
- Add parameters:
  - windchill_clear_workspace: filenames
  - dimension_list: select
  - dimension_list_detail: select
  - feature_resume: *name* can be an integer for *feat_ID*
  - feature_suppress: *name* can be an integer for *feat_ID*
- **New returns:**
  - note_get: location
  - dimension_list: dwg_dim
  - dimension_list_detail: dwg_dim
- Few notes updates

7.19 0.3.3 (2019-07-13)

Bugfix:

- feature_resume: *with_children* paramt set default to *False* (issue #3)

7.20 0.3.2 (2019-07-03)

Bugfix:

- creo_list_dirs: return empty list if there is no folder in the directory (issue #1)

Add basic usage video on README

7.21 0.3.1 (2019-06-30)

Bugfixes:

- view_list: default query name="*"

7.22 0.3.0 (2019-06-29)

Bugfixes:

- file_set_mass_units: function param correction
- file_list: function param correction
- general: set active file when file is optionnal

Improvement:

- file_open: *activate* and *display* default to True
- dimension_set: file is optionnal

7.23 0.2.0 (2019-06-28)

Update for Creoson 2.4.0 release. New functions:

- parameter_set_designated
- feature_list_group_features
- feature_list_pattern_features

Add missing function:

- feature_list_params

7.24 0.1.0 (2019-06-22)

First release on PyPI.

CHAPTER 8

Indices and tables

- genindex
- modindex
- search

Python Module Index

C

Index

D

G

H

I

J

K

L

M

N

O

P

R

S

U

V

W