```
#--------------------------------------------------------------------------------
# Copyright (c) Bentley Systems, Incorporated. All rights reserved.
# See COPYRIGHT.md in the repository root for full copyright notice
#--------------------------------------------------------------------------------
from .openStaadHelper import *
from .osgeometry import OSGeometry
from .osview import OSView
from .ossupport import OSSupport
from .osload import OSLoad
from .osproperty import OSProperty
from .osoutput import OSOutput
from .ostable import OSTable
from .oscommand import OSCommand
from .osdesign import OSDesign
from .oserrors import *
from .staadVersionHelper import compare_versions
from comtypes import COMError, automation
from comtypes import client
from comtypes import CoInitialize
```

[docs]

```
def getActiveObject(filePath:str=""):
    return OSRoot(filePath)
```

[docs]

```
class OSRoot:
    CoInitialize()
```

[docs]

```
    def __init__(self, filePath=None):
        try:
            if filePath:
                self._staad = client.CoGetObject(filePath, dynamic=True)
            else:
                self._staad = client.GetActiveObject("StaadPro.OpenSTAAD")
            self._functions = [
                "Analyze",
                "AnalyzeEx",
                "AnalyzeModel",
                "CloseSTAADFile",
                "GetAnalysisStatus",
                "GetApplicationVersion",
                "GetBaseUnit",
                "GetErrorMessage",
                "GetFullJobInfo",
                "GetInputUnitForForce",
                "GetInputUnitForLength",
                "GetMainWindowHandle",
                "GetProcessHandle",
                "GetProcessId",
                "GetShortJobInfo",
```

```
                "GetSTAADFile",
                "GetSTAADFileFolder",
                "IsAnalyzing",
                "IsPhysicalModel",
                "NewSTAADFile",
                "OpenSTAADFile",
                "Quit",
                "SaveModel",
                "SetFullJobInfo",
                "SetInputUnitForForce",
                "SetInputUnitForLength",
                "SetInputUnits",
                "SetShortJobInfo",
                "SetSilentMode",
                "ShowApplication",
                "UpdateStructure",
                "Internal_TrackPython"
            ]
            for function_name in self._functions:
                self._staad._FlagAsMethod(function_name)

            self.Geometry = OSGeometry(self._staad)
            self.View = OSView(self._staad)
            self.Support = OSSupport(self._staad)
            self.Load = OSLoad(self._staad)
            self.Property = OSProperty(self._staad)
            self.Output = OSOutput(self._staad)
            self.Command = OSCommand(self._staad)
            self.Table = OSTable(self._staad)
            self.Design = OSDesign(self._staad)

            version = self.GetApplicationVersion()
            if(compare_versions(version, '25.0.1.293') == 1):
                self._staad.Internal_TrackPython()

        except(OSError, COMError):
            self._staad = None
            raise OsErrorBase("StaadPro Application not found!! ERROR with COM"
```

[docs]

```
    def Quit(self):
        """
        Close the STAAD.Pro application environment.

        Returns
        -------
        None

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> if staad_obj is None:
```

```
...     print("staad object not found")
...     exit()
>>> staad_obj.Quit()
"""
self._staad.Quit()
```

[docs]

```
def OpenSTAADFile(self, file: str):
    """
    Open the specified .STD file.

    Parameters
    ----------
    file : str
        Path to the .STD file to open.

    Returns
    -------
    None

    Examples
    --------
    >>> from openstaadpy import os_analytical
    >>> staad_obj = os_analytical.connect()
    >>> if staad_obj is None:
    ...     print("staad object not found")
    ...     exit()
    >>> staad_obj.OpenSTAADFile("C:/Models/structure.std")
    """
    file_vt = make_variant_vt(file)
    self._staad.OpenSTAADFile(file_vt)
```

[docs]

```
def NewSTAADFile(self, fileName: str, lengthUnit: int, forceUnit: int):
    """
    Create a .STD file with specified length and force units.

    Parameters
    ----------
    fileName : str
        The file name to save.
    lengthUnit : int
        Integer from 0 to 7 representing length unit
        (0-Inch, 1-Feet, 2-Feet, 3-CentiMeter, 4-Meter, 5-MilliMeter, 6-Deci
    forceUnit : int
        Integer from 0 to 7 representing force unit
        (0-Kilopound, 1-Pound, 2-Kilogram, 3-Metric Ton, 4-Newton, 5-Kilo Ne

    Returns
    -------
```

```
        None

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> if staad_obj is None:
        ...     print("staad object not found")
        ...     exit()
        >>> staad_obj.NewSTAADFile("C:/Models/new_model.std", 4, 5)
        """
        file_vt = make_variant_vt(fileName)
        if not (0 <= forceUnit <= 7):
            return
        if not (0 <= lengthUnit <= 7):
            return
        lengthUnit_vt = create_variant_int(lengthUnit)
        forceUnit_vt = create_variant_int(forceUnit)
        self._staad.NewSTAADFile(file_vt, lengthUnit_vt, forceUnit_vt)
```

[docs]

```
    def CloseSTAADFile(self):
        """
        Close the currently open .STD file.

        Returns
        -------
        None

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> if staad_obj is None:
        ...     print("staad object not found")
        ...     exit()
        >>> staad_obj.CloseSTAADFile()
        """
        self._staad.CloseSTAADFile()
```

[docs]

```
    def GetAnalysisStatus(self, modelPath: str = None):
        """
        Get analysis status for the open STAAD Model.

        Parameters
        ----------
        modelPath : str, optional
            The full path of the STAAD model. If not provided, the currently ope
```

```
        Returns
        -------
        dict
            A dictionary containing the analysis status with following keys:

            - 'ReturnValue' : int
                Status code of the analysis:

            - 'ReturnString' : str
                Description corresponding to the `ReturnValue`.

            - 'NoOfWarnings' : int
                Number of warnings generated during the analysis.

            - 'NoOfErrors' : int
                Number of errors generated during the analysis.

            - 'CPUTime' : int
                CPU time taken for the analysis in seconds.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> if staad_obj is None:
        ...     print("staad object not found")
        ...     exit()
        >>> status_dict = staad_obj.GetAnalysisStatus() # or staad_obj.GetAnalys
        >>> print(f"Return Value: {status_dict['ReturnValue']}")
        >>> print(f"Return String: {status_dict['ReturnString']}")
        >>> print(f"No Of Warnings: {status_dict['NoOfWarnings']}")
        >>> print(f"No Of Errors: {status_dict['NoOfErrors']}")
        >>> print(f"CPU Time (sec): {status_dict['CPUTime']}")
        """

        safe_NoOfWarnings = make_safe_array_long(0)
        NoOfWarnings = make_variant_vt_ref(safe_NoOfWarnings,  automation.VT_I4

        safe_NoofErrors = make_safe_array_long(0)
        NoofErrors = make_variant_vt_ref(safe_NoofErrors,  automation.VT_I4)

        safe_CPUTime = make_safe_array_double(0)
        CPUTime = make_variant_vt_ref(safe_CPUTime,  automation.VT_R8)

        if modelPath is None:
            modelPath = self.GetSTAADFile(bFullPath=True)

        retval = self._staad.GetAnalysisStatus(modelPath, NoOfWarnings, NoofErr

        status_dict = {-2: "Invalid model path",
                       -1: "Analysis Terminated",
                       0: "General Error",
                       1: "Analysis is in progress",
                       2: "Analysis completed without errors or warnings",
                       3: "Analysis completed with warnings but without errors"
                       4: "Analysis completed with errors",
```

```
                    5: "Analysis has not been performed"}

        output = {'ReturnValue':retval,
                  'ReturnString':status_dict[retval],
                   'NoOfWarnings':NoOfWarnings[0],
                  'NoOfErrors':NoofErrors[0],
                  'CPUTime':int(CPUTime[0])}

        return output
```

[docs]

```
    def GetApplicationVersion(self):
        """
        Get the current application version as a string.

        Returns
        -------
        str
            The application version in the format 'X.X.X.X'.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> if staad_obj is None:
        ...     print("staad object not found")
        ...     exit()
        >>> version = staad_obj.GetApplicationVersion()
        >>> print(version)
        """
        safe_MajorA = make_safe_array_long(0)
        MajorA = make_variant_vt_ref(safe_MajorA,  automation.VT_I4)

        safe_MajorB = make_safe_array_long(0)
        MajorB = make_variant_vt_ref(safe_MajorB,  automation.VT_I4)

        safe_Minor = make_safe_array_long(0)
        Minor = make_variant_vt_ref(safe_Minor,  automation.VT_I4)

        safe_Build = make_safe_array_long(0)
        Build = make_variant_vt_ref(safe_Build,  automation.VT_I4)

        self._staad.GetApplicationVersion(MajorA, MajorB, Minor, Build)

        output = str(MajorA[0]) + '.' + str(MajorB[0]) + '.' + str(Minor[0]) +

        return output
```

[docs]

```
    def GetBaseUnit(self):
```

```
        """
        Get the base unit for the currently open .STD file.
            - For English system of units (The values that are derived from a le
            - For Metric system of units (The values that are derived from a ler

        Returns
        -------
        str
            'English' or 'Metric'.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> if staad_obj is None:
        ...     print("staad object not found")
        ...     exit()
        >>> base_unit = staad_obj.GetBaseUnit()
        >>> print(base_unit)
        """
        safe_retVal  = make_safe_array_long(0)
        retVal  = make_variant_vt_ref(safe_retVal ,  automation.VT_I4)

        retVal = self._staad.GetBaseUnit()

        output = {1:'English',2:'Metric'}

        return output[retVal]




                                                                        [docs]
    def GetInputUnitForForce(self):
        """
        Retrieve the input unit of force of the currently open .STD file.

        Returns
        -------
        str
            The force unit name. ('Kilopound', 'Pound', 'Kilogram', 'Metric Ton

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> if staad_obj is None:
        ...     print("staad object not found")
        ...     exit()
        >>> force_unit = staad_obj.GetInputUnitForForce()
        >>> print(force_unit)
        """
        output = {'kip':'Kilopound','lb':'Pound','kg':'Kilogram','mton':'Metric

        vt_retVal  = make_safe_str()
        retVal  = make_variant_vt_ref(vt_retVal, automation.VT_BSTR)
```

```
        self._staad.GetInputUnitForForce(retVal)
        if retVal[0] not in output:
            raise_os_error_if_error_code(-1)
        return output[retVal[0]]
```

[docs]

```
    def GetInputUnitForLength(self):
        """
        Retrieve the input unit of length of the currently open .STD file.

        Returns
        -------
        str
            The length unit name. ('Inch', 'Feet', 'Feet', 'CentiMeter', 'Meter

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> if staad_obj is None:
        ...     print("staad object not found")
        ...     exit()
        >>> length_unit = staad_obj.GetInputUnitForLength()
        >>> print(length_unit)
        """
        output = {'in':'Inch','ft':'Feet', 'cm':'CentiMeter','m':'Meter','mm':'M

        vt_retVal  = make_safe_str()
        retVal  = make_variant_vt_ref(vt_retVal, automation.VT_BSTR)
        self._staad.GetInputUnitForLength(retVal)
        if retVal[0] not in output:
            raise_os_error_if_error_code(-1)
        return output[retVal[0]]
```

[docs]

```
    def GetSTAADFile(self, bFullPath: bool = True):
        """
        Retrieve the path or the name of the current .STD file.

        Parameters
        ----------
        bFullPath : bool, optional
            If True, returns the full path. If False, returns only the file name

        Returns
        -------
        str
            The file path or name.

        Examples
```

```
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> if staad_obj is None:
        ...     print("staad object not found")
        ...     exit()
        >>> file_path = staad_obj.GetSTAADFile() # or staad_obj.GetSTAADFile(bFu
        >>> print(file_path)
        """
        safe_fileName = make_safe_str()
        fileName = make_variant_vt_ref(safe_fileName,  automation.VT_BSTR)

        self._staad.GetSTAADFile(fileName, bFullPath)
        return fileName[0]
```

[docs]

```
    def GetSTAADFileFolder(self):
        """
        Retrieve the folder path of the current STAAD file.

        Returns
        -------
        str
            The folder path.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> if staad_obj is None:
        ...     print("staad object not found")
        ...     exit()
        >>> folder = staad_obj.GetSTAADFileFolder()
        >>> print(folder)
        """
        safe_fileFolder = make_safe_str()
        fileFolder = make_variant_vt_ref(safe_fileFolder,  automation.VT_BSTR)

        self._staad.GetSTAADFileFolder(fileFolder)
        return fileFolder[0]
```

[docs]

```
    def SetInputUnitForForce(self, forceUnit: int):
        """
        Set the input unit of force of the currently open .STD file.

        Parameters
        ----------
        forceUnit : int
            Integer from 0 to 7 representing force unit. (0- Kilopound, 1- Pound
```

```
        Returns
        -------
        None

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> if staad_obj is None:
        ...     print("staad object not found")
        ...     exit()
        >>> staad_obj.SetInputUnitForForce(4) # Setting force unit to Newton
        """
        if not (0 <= forceUnit <= 7):
            return
        forceUnit_vt = create_variant_int(forceUnit)
        self._staad.SetInputUnitForForce(forceUnit_vt)
```

[docs]

```
    def SetInputUnitForLength(self, lengthUnit: int):
        """
        Set the input unit of length of the currently open .STD file.

        Parameters
        ----------
        lengthUnit : int
            Integer from 0 to 7 representing length unit. (0- Inch, 1- Feet, 2-

        Returns
        -------
        None

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> if staad_obj is None:
        ...     print("staad object not found")
        ...     exit()
        >>> staad_obj.SetInputUnitForLength(4)
        """
        if not (0 <= lengthUnit <= 7):
            return
        lengthUnit_vt = create_variant_int(lengthUnit)
        self._staad.SetInputUnitForLength(lengthUnit_vt)
```

[docs]

```
    def SetInputUnits(self, lengthUnit: int, forceUnit: int):
        """
```

```
        Set the input units of length and force of the currently open .STD file

        Parameters
        ----------
        lengthUnit : int
            Integer from 0 to 7 representing length unit. (0- Inch, 1- Feet, 2-
        forceUnit : int
            Integer from 0 to 7 representing force unit. (0- Kilopound, 1- Pound

        Returns
        -------
        None

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> if staad_obj is None:
        ...     print("staad object not found")
        ...     exit()
        >>> staad_obj.SetInputUnits(4, 5)
        """
        if not (0 <= forceUnit <= 7):
            return
        if not (0 <= lengthUnit <= 7):
            return
        lengthUnit_vt = create_variant_int(lengthUnit)
        forceUnit_vt = create_variant_int(forceUnit)
        self._staad.SetInputUnits(lengthUnit_vt, forceUnit_vt)




                                                                    [docs]
    def ShowApplication(self):
        """
        Bring the STAAD.Pro Application to the foreground.

        Returns
        -------
        None

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> if staad_obj is None:
        ...     print("staad object not found")
        ...     exit()
        >>> staad_obj.ShowApplication()
        """
        self._staad.ShowApplication()
```

[docs]

```
def GetProcessHandle(self):
    """
    Retrieve the current STAAD.Pro process handle.

    Returns
    -------
    int
        The process handle.

    Examples
    --------
    >>> from openstaadpy import os_analytical
    >>> staad_obj = os_analytical.connect()
    >>> if staad_obj is None:
    ...     print("staad object not found")
    ...     exit()
    >>> handle = staad_obj.GetProcessHandle()
    >>> print(handle)
    """
    handle = self._staad.GetProcessHandle()

    return handle
```

[docs]

```
def GetProcessId(self):
    """
    Retrieve the current STAAD.Pro process ID.

    Returns
    -------
    int
        The process ID.

    Examples
    --------
    >>> from openstaadpy import os_analytical
    >>> staad_obj = os_analytical.connect()
    >>> if staad_obj is None:
    ...     print("staad object not found")
    ...     exit()
    >>> pid = staad_obj.GetProcessId()
    >>> print(pid)
    """
    procID = self._staad.GetProcessId()

    return procID
```

[docs]

```
    def Analyze(self):
        """
        Analyze the currently opened .STD file.

        Returns
        -------
        None

        See Also
        --------
        AnalyzeEx : For more options.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> if staad_obj is None:
        ...     print("staad object not found")
        ...     exit()
        >>> staad_obj.Analyze()
        """
        self._staad.Analyze()
```

[docs]

```
    def SaveModel(self, saveSilent: bool = False):
        """
        Save the current structure with optional silent mode.

        Parameters
        ----------
        saveSilent : bool, optional
            If True, saves the model silently. Default is False.

        Returns
        -------
        None

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> if staad_obj is None:
        ...     print("staad object not found")
        ...     exit()
        >>> staad_obj.SaveModel()
        >>> staad_obj.SaveModel(saveSilent=True)
        """
        self._staad.SaveModel(int(saveSilent))
```

[docs]

```
    def IsAnalyzing(self):
        """
        Specify whether the analysis is running or not.

        Returns
        -------
        bool
            True if analysis is running, False otherwise.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> if staad_obj is None:
        ...     print("staad object not found")
        ...     exit()
        >>> staad_obj.IsAnalyzing()
        """
        bAnalysing = self._staad.IsAnalyzing()
        return bool(bAnalysing)
```

[docs]

```
    def UpdateStructure(self):
        """
        Update the structure in the STAAD.Pro application.

        Returns
        -------
        None

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> if staad_obj is None:
        ...     print("staad object not found")
        ...     exit()
        >>> staad_obj.UpdateStructure()
        """
        self._staad.UpdateStructure()
```

[docs]

```
    def GetMainWindowHandle(self):
        """
        Get the main window handle of the STAAD.Pro application.

        Returns
        -------
        int
            Window handle.
```

```
    Examples
    --------
    >>> from openstaadpy import os_analytical
    >>> staad_obj = os_analytical.connect()
    >>> if staad_obj is None:
    ...     print("staad object not found")
    ...     exit()
    >>> hwnd = staad_obj.GetMainWindowHandle()
    """
    return self._staad.GetMainWindowHandle()
```

[docs]

```
def GetShortJobInfo(self):
    """
    Get short job information for the current model.

    Returns
    -------
    tuple of 3 Strings
        (job name, job ID, job status)

    Examples
    --------
    >>> from openstaadpy import os_analytical
    >>> staad_obj = os_analytical.connect()
    >>> if staad_obj is None:
    ...     print("staad object not found")
    ...     exit()
    >>> job_name, job_id, job_status = staad_obj.GetShortJobInfo()
    >>> print(f"Job Name: {job_name}")
    >>> print(f"Job ID: {job_id}")
    >>> print(f"Job Status: {job_status}")
    """
    safe_str = make_safe_str()
    job_name = make_variant_vt_ref(safe_str, automation.VT_BSTR)

    safe_str = make_safe_str()
    job_id = make_variant_vt_ref(safe_str, automation.VT_BSTR)

    safe_str = make_safe_str()
    job_status = make_variant_vt_ref(safe_str, automation.VT_BSTR)
    self._staad.GetShortJobInfo(job_name, job_id, job_status)
    return job_name[0], job_id[0], job_status[0]
```

[docs]

```
def SetShortJobInfo(self, job_name: str, job_id: str, job_status: str):
    """
    Set short job information for the current model.
```

```
        Parameters
        ----------
        job_name : str
            Job name.
        job_id : str
            Job ID.
        job_status : str
            Job status.

        Returns
        -------
        None

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> if staad_obj is None:
        ...     print("staad object not found")
        ...     exit()
        >>> staad_obj.SetShortJobInfo("NewJob", "12345", "In Progress")
        """
        job_name_vt = make_variant_vt(job_name)
        job_id_vt = make_variant_vt(job_id)
        job_status_vt = make_variant_vt(job_status)
        self._staad.SetShortJobInfo(job_name_vt, job_id_vt, job_status_vt)
```

[docs]

```
    def AnalyzeModel(self):
        """
        Analyze the current model.

        Returns
        -------
        None

        see Also
        --------
        AnalyzeEx : For more options.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> if staad_obj is None:
        ...     print("staad object not found")
        ...     exit()
        >>> staad_obj.AnalyzeModel()
        """
        self._staad.AnalyzeModel()
```

[docs]

```
def SetSilentMode(self, silent: bool):
    """
    Set silent mode for the application.

    Parameters
    ----------
    silent : bool

    Returns
    -------
    None

    Examples
    --------
    >>> from openstaadpy import os_analytical
    >>> staad_obj = os_analytical.connect()
    >>> if staad_obj is None:
    ...     print("staad object not found")
    ...     exit()
    >>> staad_obj.SetSilentMode(True)
    """
    self._staad.SetSilentMode(int(silent))
```

[docs]

```
def GetErrorMessage(self):
    """
    Returns error messages thrown by OpenSTAAD (e.g. - for unavailability of

    Returns
    -------
    str
        The error message string.

    Examples
    --------
    >>> from openstaadpy import os_analytical
    >>> staad_obj = os_analytical.connect()
    >>> if staad_obj is None:
    ...     print("staad object not found")
    ...     exit()
    >>> msg = staad_obj.GetErrorMessage()
    """
    msg = self._staad.GetErrorMessage()
    return msg
```

[docs]

```
def AnalyzeEx(self, silentMode: int, hiddenMode: int, waitTillComplete: int
    """
```

```
        Analyze the model with extended options.

        Notes
        -----
        - This extended method analyzes the currently opened .STD file. This met
        - However, it has additional three arguments to specify whether to run t
        - The third parameter specifies whether the method should wait for the a
        - This method may be used in conjunction with SetSilentMode(), if one wa

        Parameters
        ----------
        silentMode : int
            Integer value to enable silent mode. [1 = Enable, 0 otherwise].
            Enabling silent mode will suppress all dialog boxes in the engine wh
            The analysis dialog box however will be displayed and close automati
        hiddenMode : int
            Integer value to enable hidden mode. [1 = Enable, 0 = Disable].
            Enabling hidden mode will suppress the display of analysis dialog.
            The analysis dialog box will not be displayed.
        waitTillComplete : int
            Integer value to specify whether to wait for the analysis process to

        Returns
        -------
        int
            Returns -1 if Analysis Terminated.
            Returns 0 if General Error.
            Returns 1 if Analysis is in progress.
            Returns 2 if Analysis completed without errors or warnings.
            Returns 3 if Analysis completed with warnings but without errors.
            Returns 4 if Analysis completed with errors.
            Returns 5 if Analysis has not been performed.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> if staad_obj is None:
        ...     print("staad object not found")
        ...     exit()
        >>> status = staad_obj.AnalyzeEx(1, 0, 1)
        """
        return self._staad.AnalyzeEx(silentMode, hiddenMode, waitTillComplete)




                                                                    [docs]
    def IsPhysicalModel(self):
        """
        Check if the current model is a physical model.

        Returns
        -------
        bool
```

```
        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> if staad_obj is None:
        ...     print("staad object not found")
        ...     exit()
        >>> staad_obj.IsPhysicalModel()
        """
        return bool(self._staad.IsPhysicalModel())
```

[docs]

```
    def GetFullJobInfo(self):
        """
        Get full job information for the current model.

        Returns
        -------
        list of 14 Strings
            [job name, job client, engineer's name, engineer date, job number, r

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> if staad_obj is None:
        ...     print("staad object not found")
        ...     exit()
        >>> info = staad_obj.GetFullJobInfo()
        """
        job_name = make_safe_str()
        job_name_ref = make_variant_vt_ref(job_name, automation.VT_BSTR)

        job_client = make_safe_str()
        job_client_ref = make_variant_vt_ref(job_client, automation.VT_BSTR)

        eng_name = make_safe_str()
        eng_name_ref = make_variant_vt_ref(eng_name, automation.VT_BSTR)

        eng_date = make_safe_str()
        eng_date_ref = make_variant_vt_ref(eng_date, automation.VT_BSTR)

        job_number = make_safe_str()
        job_number_ref = make_variant_vt_ref(job_number, automation.VT_BSTR)

        revision = make_safe_str()
        revision_ref = make_variant_vt_ref(revision, automation.VT_BSTR)

        part_name = make_safe_str()
        part_name_ref = make_variant_vt_ref(part_name, automation.VT_BSTR)

        reference = make_safe_str()
        reference_ref = make_variant_vt_ref(reference, automation.VT_BSTR)
```

```
        checker_name = make_safe_str()
        checker_name_ref = make_variant_vt_ref(checker_name, automation.VT_BSTR

        checker_date = make_safe_str()
        checker_date_ref = make_variant_vt_ref(checker_date, automation.VT_BSTR

        approver_name = make_safe_str()
        approver_name_ref = make_variant_vt_ref(approver_name, automation.VT_BS

        approval_date = make_safe_str()
        approval_date_ref = make_variant_vt_ref(approval_date, automation.VT_BS

        comments = make_safe_str()
        comments_ref = make_variant_vt_ref(comments, automation.VT_BSTR)

        self._staad.GetFullJobInfo(job_name_ref, job_client_ref, eng_name_ref, e
                                   reference_ref, checker_name_ref, checker_date

        return [job_name_ref[0], job_client_ref[0], eng_name_ref[0], eng_date_re
                reference_ref[0], checker_name_ref[0], checker_date_ref[0], appr
```

[docs]

```
    def SetFullJobInfo(self, job_name: str, job_client: str = "", eng_name: str
                       revision: str = "", part_name: str = "", reference: str =
                       approver_name: str = "" , approval_date: str = "", comme
        """
        Set full job information for the current model.

        Parameters
        ----------
        job_name : str
            Name of the job.
        job_client : str
            Client name for the job.
        eng_name : str
            Engineer's name.
        eng_date : str
            Engineer's date.
        job_number : str
            Job number.
        revision : str
            Revision number.
        part_name : str
            Part name.
        reference : str
            Reference.
        checker_name : str
            Checker name.
        checker_date : str
            Checker date.
        approver_name : str
            Approver name.
```

```
        approval_date : str
            Approval date.
        comments : str
            Comments.

        Returns
        -------
        None

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> if staad_obj is None:
        ...     print("staad object not found")
        ...     exit()
        >>> staad_obj.SetFullJobInfo("Full info")
        """
        job_name_vt = make_variant_vt(job_name)
        job_client_vt = make_variant_vt(job_client)
        eng_name_vt = make_variant_vt(eng_name)
        eng_date_vt = make_variant_vt(eng_date)
        job_number_vt = make_variant_vt(job_number)
        revision_vt = make_variant_vt(revision)
        part_name_vt = make_variant_vt(part_name)
        reference_vt = make_variant_vt(reference)
        checker_name_vt = make_variant_vt(checker_name)
        checker_date_vt = make_variant_vt(checker_date)
        approver_name_vt = make_variant_vt(approver_name)
        approval_date_vt = make_variant_vt(approval_date)
        comments_vt = make_variant_vt(comments)
        self._staad.SetFullJobInfo(job_name_vt, job_client_vt, eng_name_vt, eng_
                                   reference_vt, checker_name_vt, checker_date_v
```