```
#---------------------------------------------------------------------------
# Copyright (c) Bentley Systems, Incorporated. All rights reserved.
# See COPYRIGHT.md in the repository root for full copyright notice
#---------------------------------------------------------------------------
from .openStaadHelper import *
from comtypes import automation
from comtypes import CoInitialize
from .oserrors import *
```

[docs]

```
class OSProperty:
    CoInitialize()
```

[docs]

```
    def __init__(self, staadObj):
        self._staad = staadObj
        self._property = self._staad.Property
        self._functions= [
            "AssignBeamProperty",
            "AssignPlateThickness",
            "AssignMemberSpecToBeam",
            "AssignMaterialToPlate",
            "AssignMaterialToMember",
            "CreatePlateThicknessProperty",
            "CreateBeamPropertyFromTable",
            "CreateAnglePropertyFromTable",
            "CreateMemberOffsetSpec",
            "CreateMemberReleaseSpec",
            "GetMemberReleaseSpec",
            "GetPlateThickness",
            "GetBeamPropertyAll",
            "GetBeamProperty",
            "GetMaterialProperty",
            "GetBeamMaterialName",
            "GetElementMaterialName",
            "GetPlateMaterialName",
            "DeleteMaterial",
            "SetMaterialName",
            "RemoveMaterialFromBeam",
            "RemoveMaterialFromPlate",
            "CreateChannelPropertyFromTable",
            "CreateTubePropertyFromTable",
            "CreatePipePropertyFromTable",
            "CreatePrismaticRectangleProperty",
            "CreatePrismaticCircleProperty",
            "CreatePrismaticTeeProperty",
            "CreatePrismaticTrapezoidalProperty",
            "CreatePrismaticGeneralProperty",
            "CreateTaperedIProperty",
            "CreateTaperedTubeProperty",
            "CreateAssignProfileProperty",
            "AssignBetaAngle",
            "CreateMemberTrussSpec",
            "CreateMemberInactiveSpec",
            "CreateMemberTensionSpec",
            "CreateMemberCompressionSpec",
            "CreateMemberIgnoreStiffSpec",
            "CreateMemberCableSpec",
            "CreateElementPlaneStressSpec",
            "CreateElementIgnoreInplaneRotnSpec",
            "AssignElementSpecToPlate",
            "CreateMemberPartialReleaseSpec",
            "CreateElementNodeReleaseSpec",
            "GetCountryTableNo",
            "GetSectionTableNo",
            "GetBeamSectionName",
            "GetBeamSectionPropertyTypeNo",
            "GetBetaAngle",
            "GetSectionPropertyCount",
            "GetSectionPropertyName",
            "GetSectionPropertyType",
            "GetSectionPropertyCountry",
            "GetIsotropicMaterialCount",
            "GetIsotropicMaterialProperties",
            "GetOrthotropic2DMaterialCount",
            "GetOrthotropic2DMaterialProperties",
            "GetOrthotropic3DMaterialCount",
            "GetOrthotropic3DMaterialProperties",
            "GetMemberGlobalOffSet",
            "GetMemberLocalOffSet",
            "GetIsotropicMaterialPropertiesAssigned",
            "AddControlDependentRelation",
            "CreateIsotropicMaterialProperties",
            "CreateUPTTable",
            "RemoveUPTTable",
            "AddUPTPropertyWIDEFLANGE",
            "AddUPTPropertyCHANNEL",
            "AddUPTPropertyANGLE",
            "AddUPTPropertyDOUBLEANGLE",
            "AddUPTPropertyTEE",
            "AddUPTPropertyPIPE",
            "AddUPTPropertyTUBE",
            "AddUPTPropertyGENERAL",
            "AddUPTPropertyISECTION",
            "AddUPTPropertyPRISMATIC",
            "RemovePropertyFromUPTTable",
            "CreateMemberAttribute",
            "AssignMemberAttribute",
            "DeleteMemberAttribute",
            "GetMemberCountByAttribute",
            "GetMemberListByAttribute",
            "CreateElementAttribute",
            "AssignElementAttribute",
            "DeleteElementAttribute",
            "GetElementCountByAttribute",
            "GetElementListByAttribute",
            "GetAssignedAttributeCount",
            "GetAssignedAttributeByIndex",
            "RemoveAttribute",
            "GetMemberSpecCode",
            "GetPublishedProfileName",
            "GetSTAADProfileName",
            "GetSectionPropertyValues",
            "GetSectionPropertyValuesEx",
            "DeleteMemberReleaseSpec",
            "GetBeamSectionPropertyValuesEx",
            "GetSectionPropertyAssignedBeamCount",
            "GetSectionPropertyAssignedBeamList",
            "GetIsotropicMaterialAssignedBeamCount",
            "GetIsotropicMaterialAssignedBeamList",
            "CreatePropertyFromUserTable",
            "GetBeamSectionPropertyRefNo",
            "GetUserProvidedTableCount",
            "GetSectionPropertyList",
            "RemovePropertyFromBeam",
            "DeleteProperty",
            "GetUserProvidedTableList",
            "GetUserProvidedTableSectionCount",
            "GetUserProvidedTableSectionList",
            "GetUserProvidedTableSectionProperties",
            "GetPropertyUniqueID",
            "SetPropertyUniqueID",
            "DeleteMemberSpec",
            "RemoveMemberReleaseSpecFromBeam",
            "RemoveMemberOffsetSpecFromBeam",
            "RemoveMemberTrussSpecFromBeam",
            "RemoveMemberInactiveSpecFromBeam",
            "RemoveMemberTensionSpecFromBeam",
            "RemoveMemberIgnoreStiffSpecFromBeam",
            "GetBeamConstants",
            "CreateBeamPropertyFromTableEx",
            "RemoveMemberCompressionSpecFromBeam",
            "RemoveMemberCableSpecFromBeam",
            "RemoveElementPlaneStressSpecFromPlate",
            "RemoveElementIgnoreInplaneRotnSpecFromPlate",
            "RemoveElementNodeReleaseSpecFromPlate",
            "GetUserProvidedTableNo",
            "GetUserProvidedTableSectionType",
            "GetMemberReleaseSpecEx",
            "GetThicknessPropertyCount",
            "GetThicknessPropertyList",
            "GetThicknessPropertyAssignedPlateCount",
            "GetThicknessPropertyAssignedPlateList",
            "GetThicknessPropertyValues",
            "GetPlateSectionPropertyRefNo",
            "RemovePropertyFromPlate",
            "GetIsotropicMaterialAssignedPlateCount",
            "GetIsotropicMaterialAssignedPlateList",
            "AssignMaterialToSolid",
            "RemoveMaterialFromSolid",
            "GetSolidMaterialName",
            "GetIsotropicMaterialAssignedSolidCount",
            "GetIsotropicMaterialAssignedSolidList",
            "CreateIsotropicMaterialPropertiesEx",
            "GetIsotropicMaterialPropertiesEx",
            "GetMaterialPropertyEx",
            "CreateUPTTableEx",
            "GetShapeCode",
            "GetRecordForSection",
            "GetMemberAttributeCount",
            "GetMemberAttributeList",
            "GetUserProvidedTableSectionPropertyCount",
            "CreateBeamPropertyFromTableComposite",
            "CreateBeamPropertyFromTableWithCoverPlates",
            "AddUPTPropertyWIDEFLANGEUNEQUAL",
            "AddUPTPropertyWIDEFLANGECOMPOSITE",
            "CreateTeePropertyFromTable",
            "SetTypeToIsotropicMaterial",
            "GetTypeForIsotropicMaterial",
            "CreatePropertyFromUPTTable",
            "CreatePlateThicknessProperty",
            "GetUptGeneralProfilePointsCount",
            "GetUptGeneralProfileBoundaryPoints",
            "GetUptGeneralStressLocationPoints",
            "GetInactiveMemberCount",
            "GetInactiveMemberList",
            "GetAlphaAngleForSection",
            "GetCentroidLocationForSection",
            "DeleteAllControlDependentRelations",
            "CreateWideFlangePropertyFromTable",
            "CreateIsotropicMaterialSteel",
            "CreateIsotropicMaterialConcrete",
            "CreateIsotropicMaterialAluminum",
            "CreateIsotropicMaterialTimber",
            "RemoveAllElementNodeReleaseSpec",
            "CreateElementOffsetSpec",
            "CreateElementLocalZOffsetSpec",
            "GetElementLocalOffset",
            "GetElementGlobalOffSet",
            "GetElementOffSetSpec",
            "GetCountofSectionPropertyValuesEx",
            "CreateMemberCableSpecEx",
            "GetElementOffsetSpecCount",
            "RemoveAllElementOffsetSpec",
            "UpdatePropertiesToDesignSection",
            "GetFireProofedBeamCount",
            "GetFireProofedBeamList",
            "GetFireProofDataForBeam",
            "GetFireProofingSpecCount",
            "GetFireProofingSpecDetails",
            "GetFireProofingSpecAssignedBeamCount",
            "GetFireProofingSpecAssignedBeamList",
            "CreateMemberFireProofingSpec",
            "RemoveMemberFireProofingSpecFromBeam",
            "GetBeamSectionDisplayName",
            "SetStandardProfileDBFolder",
            "GetStandardProfileDBFolder",
            "GetDefaultStandardProfileDBFolder",
            "IsStandardDatabaseSection",
            "GetStandardSectionDatabaseName",
            "GetStandardSectionTableName",
            "GetStandardSectionName"
```

```
        ]

        for function_name in self._functions:
            self._property._FlagAsMethod(function_name)
```

[docs]

```
    def AssignBeamProperty(self, beam_ids: list|int, property_id: int):
        """
        Assign beam property to a single or multiple beams.

        Parameters
        ----------
        beam_ids : list of int or int
            List of beam ids or a single beam id to which the property will be a
        property_id : int
            ID of the property to assign.

        Returns
        -------
        int
            Status code indicating the result of the operation:
                - 0 : Success
                - -106 : BeamNo array dimension error.
                - -3006 : Invalid member number ID(s).
                - -6001 : Invalid section The assigned section property ID.
                - -6002 : Library Error: Property Assign.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> country_code = 6
        >>> section_name = "HE100A"
        >>> type_spec = 0
        >>> add_spec_1 = 0.0
        >>> add_spec_2 = 0.0
        >>> property_id = staad_obj.Property.CreateBeamPropertyFromTable(country
        >>> beam_ids = staad_obj.Geometry.GetBeamList()    # Getting all beam Id
        >>> output = staad_obj.Property.AssignBeamProperty(beam_ids[0:3], proper
        >>> output = staad_obj.Property.AssignBeamProperty(4, property_id)    #
        """
        if (isinstance(beam_ids, int)):
            beam_ids = [beam_ids]

        beamId_safe_list = make_safe_array_long_input(beam_ids)
        beamId_Array_vt = make_variant_vt_ref(beamId_safe_list,  automation.VT_A
        return self._property.AssignBeamProperty(beamId_Array_vt, property_id)
```

[docs]

```
    def AssignPlateThickness(self, plate_ids: list, thickness_property_id: int)
```

```
        """
        Assigns a plate thickness property to the specified plates.

        Parameters
        ----------
        plate_ids : list of int
            List of plate numbers to which the thickness property will be assign

        thickness_property_id : int
            ID of the plate thickness property to assign.

        Returns
        -------
        int
            Status code indicating the result of the operation:
                - 0 : Success
                - -1 : General error.
                - -106 : Plate number array dimension error.
                - -113 : Plate number type error (expected int or long).
                - -4009 : All provided plate numbers are invalid.
                - -4008 : Some of the plate numbers provided are invalid.
                - -6001 : The specified thickness property ID is invalid.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> property_id = staad_obj.Property.CreatePlateThicknessProperty([2, 2,
        >>> plate_list = staad_obj.Geometry.GetPlateList()
        >>> status = staad_obj.Property.AssignPlateThickness(plate_list, propert
        """
        if (isinstance(plate_ids, int)):
            plate_ids = [plate_ids]
        plateNosId_safe_list = make_safe_array_long_input(plate_ids)
        plateNoId_Array_vt = make_variant_vt_ref(plateNosId_safe_list,  automat
        result = self._property.AssignPlateThickness(plateNoId_Array_vt, thickne
        if result < 0:
            raise_os_error_if_error_code(result)
        return result
```

[docs]

```
    def AssignMemberSpecToBeam(self, beam_ids: list|int, spec_id: int):
        """
        Assign a member specification to specified beams.

        Parameters
        ----------
        beam_ids : list of int or int
            List of member numbers to assign the specification to.
        spec_id : int
            The ID of the member specification.

        Returns
        -------
        int
            Status code indicating the result of the operation:
                - 0 : Success
                - -106 : List of long expected.
                - -6017 : Library Error: Unable to assign specification.


        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> beam_ids = staad_obj.Geometry.GetBeamList()
        >>> status = staad_obj.Property.AssignMemberSpecToBeam(beam_ids, 2)
        """
        if (isinstance(beam_ids, int)):
            beam_ids = [beam_ids]
        beam_ids_safe_list = make_safe_array_long_input(beam_ids)
        beam_ids_array_vt = make_variant_vt_ref(beam_ids_safe_list,  automation
        return self._property.AssignMemberSpecToBeam(beam_ids_array_vt, spec_id
```

[docs]

```
    def AssignMaterialToPlate(self, material_name: str, plate_ids: list|int):
        """
        Assign a material property to specified plates.

        Parameters
        ----------
        material_name : str
            The ID of the material property.
        plate_ids : list of int
            List of plate numbers to assign the material to.

        Returns
        -------
        int
            Status code indicating the result of the operation:
                - 0 : Success
                - -113 : Invalid data type(Long or Int Expected)
                - -4009 : All the plate numbers are invalid.
                - -4008 : Some of the plate numbers are invalid.
                - -6023 : Material not found.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> plate_list = staad_obj.Geometry.GetPlateList()
        >>> status = staad_obj.Property.AssignMaterialToPlate("CONCRETE1", plate
        """
        if (isinstance(plate_ids, int)):
            plate_ids = [plate_ids]
        plate_ids_safe_list = make_safe_array_long_input(plate_ids)
        plate_ids_array_vt = make_variant_vt_ref(plate_ids_safe_list,  automatic
        return self._property.AssignMaterialToPlate(material_name, plate_ids_arr
```

[docs]

```
    def AssignMaterialToMember(self, material_name: str, member_ids: list):
        """
        Assign a material property to specified members.

        Parameters
        ----------
        material_name : str
            The ID of the material property.
        member_ids : list of int
            List of member numbers to assign the material to.

        Returns
        -------
        bool
            - 'True' if it succeeds
            - 'False' if it fails

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> result = staad_obj.Property.AssignMaterialToMember("CONCRETE1", [5,
        """
        if (isinstance(member_ids, int)):
            member_ids = [member_ids]
        member_ids_safe_list = make_safe_array_long_input(member_ids)
        member_ids_array_vt = make_variant_vt_ref(member_ids_safe_list,  automat
        return self._property.AssignMaterialToMember(material_name, member_ids_a
```

[docs]

```
    def CreatePlateThicknessProperty(self, thickness_list: list):
        """
        Create a new plate thickness property.

        Parameters
        ----------
        thickness_list : list of float
            The thickness value for the plate.

        Returns
        -------
        int
            Returns id of the created plate thickness property if successful.
            Returns -1 if it encounters an issue regarding the thickness list.
            Returns -6003 if it is unable to create property (library error).
        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> propertyId = staad_obj.Property.CreatePlateThicknessProperty([2, 2,
        """
        safe_thickness_array = make_safe_array_double_input(thickness_list)
        thickness_array_vt = make_variant_vt_ref(safe_thickness_array,  automat

        return self._property.CreatePlateThicknessProperty(thickness_array_vt)
```

[docs]

```
    def CreateBeamPropertyFromTable(self, country_code: int, section_name: str,
        """
        Create a new beam property from a table.

        Parameters
        ----------
        country_code : int
            Code resembling specific country:
                +--------------+----------------------+
                | Country Code | Country              |
                +==============+======================+
                | 1            | American             |
                +--------------+----------------------+
                | 2            | Australian           |
                +--------------+----------------------+
                | 3            | British              |
                +--------------+----------------------+
                | 4            | Canadian             |
                +--------------+----------------------+
                | 5            | Chinese              |
                +--------------+----------------------+
                | 6            | Dutch                |
                +--------------+----------------------+
                | 7            | European             |
                +--------------+----------------------+
                | 8            | French               |
                +--------------+----------------------+
                | 9            | German               |
                +--------------+----------------------+
                | 10           | Indian               |
                +--------------+----------------------+
                | 11           | Japanese             |
                +--------------+----------------------+
                | 12           | Russian              |
                +--------------+----------------------+
                | 13           | SouthAfrican         |
                +--------------+----------------------+
                | 14           | Spanish              |
                +--------------+----------------------+
                | 15           | Venezuelan           |
                +--------------+----------------------+
                | 16           | Korean                |
                +--------------+----------------------+
        section_name : str
            Name of the section
        type_spec : int
            Specification Type Number:
                +-------+------------+-------------------------------------------
                | Value | Type Spec. | Description
                +=======+============+===========================================
                | 0     | ST         |
                +-------+------------+-------------------------------------------
                | 2     | D          | Double profile.
                +-------+------------+-------------------------------------------
                | 5     | T          | Tee section cut from I shaped section   (
                +-------+------------+-------------------------------------------
        add_spec_1 : float
            Clear Spacing for Double profile.
        add_spec_2 : float
            Please set it with 0.0.

        Returns
        -------
        int
            The ID of the created beam property if successful else returns 0 if

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> openstaad = os_analytical.connect()
        >>> country_code = 6
        >>> section_name = "HE100A"
        >>> type_spec = 0
        >>> add_spec_1 = 0.0
        >>> add_spec_2 = 0.0
        >>> property_id = openstaad.Property.CreateBeamPropertyFromTable(country
        """
        return self._property.CreateBeamPropertyFromTable(country_code, section_
```

[docs]

```
    def CreateAnglePropertyFromTable(self, country_code: int, section_name: str
        """
        Create a new angle property from a table.

        Parameters
        ----------
        country_code : int
            Code resembling specific country.
        section_name : str
            The section name in the table.
        specification_type_no : int
            Specification type to use while creating angle property. [Please ref
                +-------+-----------+-------------------------------------------
                | Value | Spec Type |                               Descript
                +=======+===========+==============================================
                |0      |ST         |Single section from the standard built-in t
                +-------+-----------+----------------------------------------------
                |1      |RA         |Single angle with reverse Y-Z axes (refer t
                +-------+-----------+----------------------------------------------
                |3      |LD         |Double angle with long legs back-to-back.
                +-------+-----------+----------------------------------------------
                |4      |SD         |Double angle with short legs back-to-back
                +-------+-----------+----------------------------------------------
                |12     |SA         |Double angle in a star arrangement (heel to
                +-------+-----------+----------------------------------------------
        add_spec : float
            Additional Specification Value :
                +-------------+--------------------------+
                | Spec Value  | Specification Description |
                +=============+==========================+
                | WP TH       | for TC and BC            |
                +-------------+--------------------------+
                | WP TH BW BT | for TB / WP TH for TB    |
                +-------------+--------------------------+
                | CT FC       | for CM                   |
                +-------------+--------------------------+
                | SP          | for D, BA and FR         |
                +-------------+--------------------------+
                | SP          | for LD and SD            |
                +-------------+--------------------------+
                | TH WT DT    | for Tube define          |
                +-------------+--------------------------+
                | OD ID       | for Pipe define          |
                +-------------+--------------------------+

        Returns
        -------
        int
            The ID of the created beam property if successful else returns 0 if

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> openstaad = os_analytical.connect()
        >>> country_code = 6
        >>> section_name = "HE100A"
        >>> specification_type_no = 0
        >>> add_spec = 0.0
        >>> property_id = openstaad.Property.CreateAnglePropertyFromTable(countr
        """
        return self._property.CreateAnglePropertyFromTable(country_code, sectio
```

[docs]

```
    def CreateMemberOffsetSpec(self, offset_location: int, offset_with_respect_t
        """
        Create a member offset specification.
        Parameters
        ----------
        offset_location : int
            Sets Offset Location at start if passed '0' else at the end if passe
        offset_with_respect_to: int
            Sets Offset with respect to Global Axis if passed '0' else to Local 
        offset_x : float
            The offset x coordinate.
        offset_y : float
            The offset y coordinate.
        offset_z : float
            The offset z coordinate.

        Returns
        -------
        int
            The id of the created member offset specification if successful else

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> spec_id = staad_obj.Property.CreateMemberOffsetSpec(0, 0, 0.5, 0.0, 
        """
        return self._property.CreateMemberOffsetSpec(offset_location, offset_wi




                                                                          [docs]
    def CreateMemberReleaseSpec(self, offset_location: int, dof_values: list, s
        """
        Create a member release specification.

        Parameters
        ----------
        offset_location: int
            The offset location at START (= 0) or END (= 1) of the member.
        dof_values : list of int
            Degrees of freedom: No Release (= 0) or Release (= 1) for FX, FY, FZ
        spring_constant_values : list of float
            The variable spring constants KFX, KFY, KFZ, KMX, KMY and KMZ.

        Returns
        -------
        int
            Returns the ID of the created member release specification if succes
            Returns -106 if list of long for dof_values and list of double for s
            Returns -108 if array size is smaller than expected (size should be 
            Returns -6020 if library error: unable to create member release spec


        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> property_id = staad_obj.Property.CreateMemberReleaseSpec(0, [0, 1, 0
        """
        dof_values_safe_list = make_safe_array_long_input(dof_values)
        spring_constant_values_safe_list = make_safe_array_double_input(spring_c
        ref_dof_values_array_vt = make_variant_vt_ref(dof_values_safe_list,  aut
        ref_spring_constant_values_array_vt = make_variant_vt_ref(spring_constan
        result = self._property.CreateMemberReleaseSpec(offset_location, ref_dof
        if result < 0:
            raise_os_error_if_error_code(result)
        return result
```

[docs]

```
    def GetMemberReleaseSpec(self, member_no: int, end: int):
        """
        Get the release specification for a member at the specified end.

        Parameters
        ----------
        member_no : int
            The member number.
        end: int
            Sets End at start if passed '0' else at the end if passed '1', for w

        Returns
        -------
        Tuple : Tuple(List, List)
            Tuple consisting of List of Release Values (6 elements for 6 DOFs. E


        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> beam_ids = staad_obj.Geometry.GetBeamList()
        >>> release_values, spring_constant_values  = staad_obj.Property.GetMemb
        """
        release_values_safe_list = make_safe_array_long(6)
        release_values_array_vt = make_variant_vt_ref(release_values_safe_list,
        spring_constant_values_safe_list = make_safe_array_double(6)
        spring_constant_values_array_vt = make_variant_vt_ref(spring_constant_va
        result = self._property.GetMemberReleaseSpec(member_no, end, release_val
        if not result:
            raise_os_error_if_error_code(-1)
        return (release_values_array_vt[0], spring_constant_values_array_vt[0])
```

[docs]

```
    def GetPlateThickness(self, plate_no: int):
        """
        Get the thickness property of a plate.
        Parameters
        ----------
        plate_no : int
            The plate number.

        Returns
        -------
        List : Float list
            The thickness of nodes in the plate.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> plate_ids = staad_obj.Geometry.GetPlateList()
        >>> thickness_values = staad_obj.Property.GetPlateThickness(plate_ids[0
        """
        safe_array = make_safe_array_double(4)
        thickness_array_vt = make_variant_vt_ref(safe_array,  automation.VT_ARR
        result = self._property.GetPlateThickness(plate_no, thickness_array_vt)
        if result < 0:
            raise_os_error_if_error_code(result)
        return list(thickness_array_vt[0])
```

[docs]

```
    def GetBeamPropertyAll(self, beam_id:int):
        """
        Gets long member properties of the specified beam member.

        Parameters
        ----------
        beam_id : int
            The ID of the beam property.

        Returns
        -------
        tuple : tuple<float, float, float, float, float, float, float, float, f
            Tuple of short member properties consisting of width of the section
            cross section area, shear area in local y-axis, shear area in local
            Moment of inertia about local z-axis, moment of inertia about local
            thickness of top flange and thickness of web respectively.

            If shear area in local y-axis & z-axis is zero, shear deformation i

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> beam_ids = staad_obj.Geometry.GetBeamList()
        >>> section_width, section_depth, cross_section_area, shear_area_y, she
        """
        safe_width = make_safe_array_double(1)
        width = make_variant_vt_ref(safe_width, automation.VT_R8)
        safe_depth = make_safe_array_double(1)
        depth = make_variant_vt_ref(safe_depth, automation.VT_R8)

        safe_ax = make_safe_array_double(1)
        ax = make_variant_vt_ref(safe_ax, automation.VT_R8)

        safe_ay = make_safe_array_double(1)
        ay = make_variant_vt_ref(safe_ay, automation.VT_R8)

        safe_az = make_safe_array_double(1)
        az = make_variant_vt_ref(safe_az, automation.VT_R8)

        safe_mIz = make_safe_array_double(1)
        mIz = make_variant_vt_ref(safe_mIz, automation.VT_R8)

        safe_mIy = make_safe_array_double(1)
        mIy = make_variant_vt_ref(safe_mIy, automation.VT_R8)

        safe_iz = make_safe_array_double(1)
        iz = make_variant_vt_ref(safe_iz, automation.VT_R8)

        safe_tf = make_safe_array_double(1)
        tf = make_variant_vt_ref(safe_tf, automation.VT_R8)

        safe_tw = make_safe_array_double(1)
        tw = make_variant_vt_ref(safe_tw, automation.VT_R8)


        result = self._property.GetBeamPropertyAll(beam_id, width, depth, ax, a
        if result != 1:
            raise_os_error_if_error_code(-1)
        return width[0], depth[0], ax[0], ay[0], az[0], mIz[0], mIy[0], iz[0],
```

[docs]

```
    def GetBeamProperty(self, beam_id: int):
        """
        Get a short member properties of the specified beam member.

        Parameters
        ----------
        beam_id : int
            The ID of the beam property.

        Returns
        -------
        tuple : tuple<float, float, float, float, float, float, float, float>
            Tuple of short member properties consisting of width of the section
            cross section area, shear area in local y-axis, shear area in local
            moment of inertia about local z-axis, moment of inertia about local
            respectively.

            If shear area in local y-axis & z-axis is zero, shear deformation i
        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> beam_ids = staad_obj.Geometry.GetBeamList()
        >>> section_width, section_depth, cross_section_area, shear_area_y, shea
        """
        safe_width = make_safe_array_double(1)
        width = make_variant_vt_ref(safe_width, automation.VT_R8)

        safe_depth = make_safe_array_double(1)
        depth = make_variant_vt_ref(safe_depth, automation.VT_R8)

        safe_ax = make_safe_array_double(1)
        ax = make_variant_vt_ref(safe_ax, automation.VT_R8)

        safe_ay = make_safe_array_double(1)
        ay = make_variant_vt_ref(safe_ay, automation.VT_R8)

        safe_az = make_safe_array_double(1)
        az = make_variant_vt_ref(safe_az, automation.VT_R8)

        safe_mIz = make_safe_array_double(1)
        mIz = make_variant_vt_ref(safe_mIz, automation.VT_R8)

        safe_mIy = make_safe_array_double(1)
        mIy = make_variant_vt_ref(safe_mIy, automation.VT_R8)

        safe_iz = make_safe_array_double(1)
        iz = make_variant_vt_ref(safe_iz, automation.VT_R8)


        result = self._property.GetBeamProperty(beam_id, width, depth, ax, ay, 
        if not result:
            raise_os_error_if_error_code(-1)
        return width[0], depth[0], ax[0], ay[0], az[0], mIz[0], mIy[0], iz[0]
```

[docs]

```
    def GetMaterialProperty(self, MaterialName: str):
        """
        Get a specific material property.

        Parameters
        ----------
        MaterialName : str
            The Name of the material .

        Returns
        -------
        tuple
            Tuple consisting of elasticity, possion, density, alpha and damping
        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> elasticity, section_depth, density, alpha, damping_ratio = staad_obj
        """
        safe_elasticity = make_safe_array_double(1)
        elasticity = make_variant_vt_ref(safe_elasticity, automation.VT_R8)

        safe_possion = make_safe_array_double(1)
        possion = make_variant_vt_ref(safe_possion, automation.VT_R8)

        safe_density = make_safe_array_double(1)
        density = make_variant_vt_ref(safe_density, automation.VT_R8)

        safe_alpha = make_safe_array_double(1)
        alpha = make_variant_vt_ref(safe_alpha, automation.VT_R8)

        safe_damp_ratio = make_safe_array_double(1)
        damp_ratio = make_variant_vt_ref(safe_damp_ratio, automation.VT_R8)

        result = self._property.GetMaterialProperty(MaterialName, elasticity, po
        if result < 0:
            raise_os_error_if_error_code(result)
        return elasticity[0], possion[0], density[0], alpha[0], damp_ratio[0]
```

[docs]

```
    def GetBeamMaterialName(self, beam_id: int):
        """
        Get the material name assigned to a beam.

        Parameters
        ----------
        beam_id : int
            The beam number id.

        Returns
        -------
        str
            The material name.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> beam_ids = staad_obj.Geometry.GetBeamList()
        >>> material_name = staad_obj.Property.GetBeamMaterialName(beam_ids[0])
        """
        return self._property.GetBeamMaterialName(beam_id)
```

[docs]

```
    def GetElementMaterialName(self, element_id: int):
        """
        Get the material name assigned to an element.

        Parameters
        ----------
        element_id : int
            The element number Id.

        Returns
        -------
        str
            The material name.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> solid_ids = staad_obj.Geometry.GetSolidList()
        >>> material_name = staad_obj.Property.GetSolidMaterialName(solid_ids[0
        """
        return self._property.GetElementMaterialName(element_id)
```

[docs]

```
    def GetPlateMaterialName(self, plate_id: int):
        """
        Get the material name assigned to a plate.

        Parameters
        ----------
        plate_id : int
            The plate number Id.

        Returns
        -------
        str
            The material name.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> plate_ids = staad_obj.Geometry.GetPlateList()
        >>> material_name = staad_obj.Property.GetPlateMaterialName(plate_ids[0
        """
        return self._property.GetPlateMaterialName(plate_id)
```

[docs]

```
    def DeleteMaterial(self, material_name: str):
        """
        Delete a material.

        Parameters
        ----------
        material_name : str
            Material Name

        Returns
        -------
        bool:
            'True' if succeeds 'else' False

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> result = staad_obj.Property.DeleteMaterial("Q235")
        """
        return self._property.DeleteMaterial(material_name)
```

[docs]

```
    def SetMaterialName(self, material_name: str):
        """
        Set the material name for a member.

        Parameters
        ----------
        material_name : str
            The material name to assign.

        Returns
        -------
        None

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> staad_obj.Property.SetMaterialName("UserDefineMaterial_1")
        """
        self._property.SetMaterialName(material_name)
```

[docs]

```
    def RemoveMaterialFromBeam(self, beam_id: int):
        """
        Remove the material assignment from a beam.

        Parameters
        ----------
        beam_id : int
            The Beam number ID.

        Returns
        -------
        Int
            Returns 0 if Ok else returns -1 if it fails.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> beam_ids = staad_obj.Geometry.GetBeamList()
        >>> result = staad_obj.Property.RemoveMaterialFromBeam(beam_ids[0])
        """
        return self._property.RemoveMaterialFromBeam(beam_id)
```

[docs]

```
    def RemoveMaterialFromPlate(self, plate_ids: list|int):
        """
        Remove the material assignment from a plate.

        Parameters
        ----------
        plate_ids : list of int or int
            The plate number.

        Returns
        -------
        Bool
            'True' if removes material else
            'False' if it fails

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> plate_ids = staad_obj.Geometry.GetPlateList()
        >>> result = staad_obj.Property.RemoveMaterialFromPlate(plate_ids[0])
        """
        if (isinstance(plate_ids, int)):
            plate_ids = [plate_ids]
        plate_ids_safe_list = make_safe_array_long_input(plate_ids)
        plate_ids_array_vt = make_variant_vt_ref(plate_ids_safe_list,  automati
        return self._property.RemoveMaterialFromPlate(plate_ids_array_vt)
```

[docs]

```
    def CreateChannelPropertyFromTable(self, country_code: int, section_name: s
        """
        Creates channel property from database.

        Parameters
        ----------
        country_code : int
            The value for the specified country
        section_name : str
            Name of the section
        spec_type : int
            The specification type number:
                +-------+----------------------+
                | Index | Spec Type            |
                +=======+======================+
                |-1     | Define               |
                +-------+----------------------+
                |0      | ST                   |
                +-------+----------------------+
                |1      | RA                   |
                +-------+----------------------+
                |2      | D                    |
                +-------+----------------------+
                |3      | LD                   |
                +-------+----------------------+
                |4      | SD                   |
                +-------+----------------------+
                |5      | T (for aluminum)     |
                +-------+----------------------+
                |6      | CM                   |
                +-------+----------------------+
                |7      | TC                   |
                +-------+----------------------+
                |8      | BC                   |
                +-------+----------------------+
                |9      | TB                   |
                +-------+----------------------+
                |10     | BA (for aluminum)    |
                +-------+----------------------+
                |11     | FR                   |
                +-------+----------------------+
                |12     | SA (for aluminum)    |
                +-------+----------------------+
        additional_spec_1 : float
            Additional specification value:
                +----------------+----------------------------+
                | Spec Value     | Specification Description  |
                +================+============================+
                | WP TH          | for TC and BC              |
                +----------------+----------------------------+
                | WP TH BW BT    | for TB / WP TH for TB      |
                +----------------+----------------------------+
                | CT FC          | for CM                     |
                +----------------+----------------------------+
                | SP             | for D, BA and FR           |
                +----------------+----------------------------+
                | SP             | for LD and SD              |
                +----------------+----------------------------+
                | TH WT DT        | for Tube define              |
                +----------------+----------------------------+
                | OD ID           | for Pipe define              |
                +----------------+----------------------------+

        Returns
        -------
        int
            Returns assigned section property ID if successful.\n
            Else returns a status code indicating the result of the operation:
                - 0 : Library Error: Unable to create property.
                - -6004 : Section is not found in profile database.
                - -6005 : Section data for a section is not found.
                - -6006 : Invalid section type.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> output = staad_obj.Property.CreateChannelPropertyFromTable(10, "ISM
        """
        return self._property.CreateChannelPropertyFromTable(country_code, sect
```

[docs]

```
    def CreateTubePropertyFromTable(self, country_code: int, section_name: str,
        """
        Creates tube property from database.

        Parameters
        ----------
        country_code : int
            The value for the specified country
        section_name : str
            Name of the section
        spec_type : int
            The specification type number:
                +-------+----------------------+
                | Index | Spec Type            |
                +=======+======================+
                |-1     | Define               |
                +-------+----------------------+
                |0      | ST                   |
                +-------+----------------------+
                |1      | RA                   |
                +-------+----------------------+
                |2      | D                    |
                +-------+----------------------+
                |3      | LD                   |
                +-------+----------------------+
                |4      | SD                   |
                +-------+----------------------+
                |5      | T (for aluminum)     |
                +-------+----------------------+
                |6      | CM                    |
                +-------+-----------------------+
                |7      | TC                    |
                +-------+-----------------------+
                |8      | BC                    |
                +-------+-----------------------+
                |9      | TB                    |
                +-------+-----------------------+
                |10     | BA (for aluminum)     |
                +-------+-----------------------+
                |11     | FR                    |
                +-------+-----------------------+
                |12     | SA (for aluminum)     |
                +-------+-----------------------+
        add_spec_1 : float
            Additional specification value:
                +----------------+-----------------------------+
                | Spec Value     | Specification Description   |
                +================+=============================+
                | WP TH          | for TC and BC               |
                +----------------+-----------------------------+
                | WP TH BW BT    | for TB / WP TH for TB       |
                +----------------+-----------------------------+
                | CT FC          | for CM                      |
                +----------------+-----------------------------+
                | SP             | for D, BA and FR            |
                +----------------+-----------------------------+
                | SP             | for LD and SD               |
                +----------------+-----------------------------+
                | TH WT DT       | for Tube define             |
                +----------------+-----------------------------+
                | OD ID          | for Pipe define             |
                +----------------+-----------------------------+
        add_spec_2 : float
            Additional specification value:
                +----------------+-----------------------------+
                | Spec Value     | Specification Description   |
                +================+=============================+
                | WP TH          | for TC and BC               |
                +----------------+-----------------------------+
                | WP TH BW BT    | for TB / WP TH for TB       |
                +----------------+-----------------------------+
                | CT FC          | for CM                      |
                +----------------+-----------------------------+
                | SP             | for D, BA and FR            |
                +----------------+-----------------------------+
                | SP             | for LD and SD               |
                +----------------+-----------------------------+
                | TH WT DT       | for Tube define             |
                +----------------+-----------------------------+
                | OD ID          | for Pipe define             |
                +----------------+-----------------------------+
        add_spec_3 : float
            Additional specification value:
                +----------------+-----------------------------+
                | Spec Value     | Specification Description   |
                +================+=============================+
                | WP TH          | for TC and BC               |
                +----------------+-----------------------------+
                | WP TH BW BT    | for TB / WP TH for TB       |
                +----------------+-----------------------------+
                | CT FC          | for CM                      |
                +----------------+-----------------------------+
                | SP             | for D, BA and FR            |
                +----------------+-----------------------------+
                | SP             | for LD and SD               |
                +----------------+-----------------------------+
                | TH WT DT       | for Tube define             |
                +----------------+-----------------------------+
                | OD ID          | for Pipe define             |
                +----------------+-----------------------------+

        Returns
        -------
        int
            Status code indicating the result of the operation:
                - 0 : Library Error: Unable to create property.
                - -6004 : Section is not found in profile database.
                - -6005 : Section data for a section is not found.
                - -6006 : Invalid section type.


        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> output = staad_obj.Property.CreateTubePropertyFromTable(10, "TUB303
        """
        return self._property.CreateTubePropertyFromTable(country_code, section
```

[docs]

```
    def CreatePipePropertyFromTable(self, country_code: int, section_name: str,
        """
        Creates pipe property from database.

        Parameters
        ----------
        country_code : int
            The value for the specified country
        section_name : str
            Name of the section
        spec_type : int
            The specification type number:
                +-------+---------------------+
                | Index | Spec Type           |
                +=======+=====================+
                |-1     | Define              |
                +-------+---------------------+
                |0      | ST                  |
                +-------+----------------------+
                |1      | RA                   |
                +-------+----------------------+
                |2      | D                    |
                +-------+----------------------+
                |3      | LD                   |
                +-------+----------------------+
                |4      | SD                   |
                +-------+----------------------+
                |5      | T (for aluminum)     |
                +-------+----------------------+
                |6      | CM                   |
                +-------+----------------------+
                |7      | TC                   |
                +-------+----------------------+
                |8      | BC                   |
                +-------+----------------------+
                |9      | TB                   |
                +-------+----------------------+
                |10     | BA (for aluminum)    |
                +-------+----------------------+
                |11     | FR                   |
                +-------+----------------------+
                |12     | SA (for aluminum)    |
                +-------+----------------------+
        additional_spec_1 : float
            Additional specification value
                +----------------+----------------------------+
                | Spec Value     | Specification Description  |
                +================+============================+
                | WP TH          | for TC and BC              |
                +----------------+----------------------------+
                | WP TH BW BT    | for TB / WP TH for TB      |
                +----------------+----------------------------+
                | CT FC          | for CM                     |
                +----------------+----------------------------+
                | SP             | for D, BA and FR           |
                +----------------+----------------------------+
                | SP             | for LD and SD              |
                +----------------+----------------------------+
                | TH WT DT       | for Tube define            |
                +----------------+----------------------------+
                | OD ID          | for Pipe define            |
                +----------------+----------------------------+
        additional_spec_2 : float
            Additional specification value
                +----------------+----------------------------+
                | Spec Value     | Specification Description  |
                +================+============================+
                | WP TH          | for TC and BC              |
                +----------------+----------------------------+
                | WP TH BW BT    | for TB / WP TH for TB      |
                +----------------+----------------------------+
                | CT FC          | for CM                     |
                +----------------+----------------------------+
                | SP             | for D, BA and FR           |
                +----------------+----------------------------+
                | SP             | for LD and SD              |
                +----------------+----------------------------+
                | TH WT DT       | for Tube define            |
                +----------------+----------------------------+
                | OD ID          | for Pipe define            |
                +----------------+----------------------------+

        Returns
        -------
        int
            Returns assigned section property ID if successful.\n
            Else returns a status code indicating the result of the operation:
                - 0 : Library Error: Unable to create property.
                - -6004 : Section is not found in profile database.
                - -6005 : Section data for a section is not found.
                - -6006 : Invalid section type.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> output = staad_obj.Property.CreatePipePropertyFromTable(17, "0.500Pi
        """
        return self._property.CreatePipePropertyFromTable(country_code, section_
```

[docs]

```
def CreatePrismaticRectangleProperty(self, depth_along_y_axis: float, depth_
    """
    Creates prismatic rectangle property.

    Parameters
    ----------
    depth_along_y_axis : float
        The depth along the local Y-axis.
    depth_along_z_axis : float
        The width along the local Z-axis.

    Returns
    -------
    int
        Returns the assigned section property ID else '0' if it gets library

    Examples
    --------
    >>> from openstaadpy import os_analytical
    >>> staad_obj = os_analytical.connect()
    >>> output = staad_obj.Property.CreatePrismaticRectangleProperty(0.5, 0
    """
    return self._property.CreatePrismaticRectangleProperty(depth_along_y_ax
```

[docs]

```
def CreatePrismaticCircleProperty(self, circle_diameter: float):
    """
    Creates prismatic circle property.

    Parameters
    ----------
    circle_diameter : float
        The circle diameter.

    Returns
    -------
    int
        Returns the assigned section property ID else '0' if it gets  Librar

    Examples
    --------
    >>> from openstaadpy import os_analytical
    >>> staad_obj = os_analytical.connect()
    >>> output = staad_obj.Property.CreatePrismaticCircleProperty(0.25)
    """
    return self._property.CreatePrismaticCircleProperty(circle_diameter)
```

[docs]

```
def CreatePrismaticTeeProperty(self, total_section_depth: float, flange_wid
    """
    Creates prismatic tee property.

    Parameters
    ----------
    total_section_depth : float
        Total depth of section (top fiber of flange to bottom fiber of web)
    flange_width : float
        Width of flange.
    stem_depth : float
        Depth of stem.
    stem_width : float
        Width of stem.

    Returns
    -------
    int
        Returns the assigned section property ID else '0' if it gets library

    Examples
    --------
    >>> from openstaadpy import os_analytical
    >>> staad_obj = os_analytical.connect()
    >>> output = staad_obj.Property.CreatePrismaticTeeProperty(0.5, 0.25, 0
    """
    return self._property.CreatePrismaticTeeProperty(total_section_depth, fl
```

[docs]

```
def CreatePrismaticTrapezoidalProperty(self, section_depth: float, top_fiber
    """
    Creates prismatic trapezoidal section property.

    Parameters
    ----------
    section_depth : float
        Total depth of section.
    top_fiber_section_width : float
        Width of section at top fiber.
    bottom_fiber_section_width : float
        Width of section at bottom fiber.

    Returns
    -------
    int
        Returns the assigned section property ID else '0' if it gets library

    Examples
    --------
    >>> from openstaadpy import os_analytical
    >>> staad_obj = os_analytical.connect()
    >>> output = staad_obj.Property.CreatePrismaticTrapezoidalProperty(0.5,
    """
    return self._property.CreatePrismaticTrapezoidalProperty(section_depth,
```

[docs]

```
def CreatePrismaticGeneralProperty(self, property_value_list: list):
    """
    Creates prismatic general property.

    Parameters
    ----------
    property_value_list : list
        The property values in double type list:
            +-------------+--------------+----------------------------------
            |    Index    | Property Type |
            +=============+==============+==================================
            |      0      |      AX      |Cross section area
            +-------------+--------------+----------------------------------
            |      1      |      AY      |Shear area in local Y-axis. If ze
            +-------------+--------------+----------------------------------
            |      2      |      AZ      |Shear area in local Z-axis. If ze
            +-------------+--------------+----------------------------------
            |      3      |      IX      |Torsional constant
            +-------------+--------------+----------------------------------
            |      4      |      IY      |Moment of inertia about local Y-a
            +-------------+--------------+----------------------------------
```

```
                |      5      |      IZ      |Moment of inertia about local Z-a
                +-------------+--------------+---------------------------------
                |      6      |      YD      |Depth of the section in the direc
                +-------------+--------------+---------------------------------
                |      7      |      ZD      |Depth of the section in the direc
                +-------------+--------------+---------------------------------
                |      8      |      YB      |Depth of stem (T-beams); width of
                +-------------+--------------+---------------------------------
                |      9      |      ZB      |Width of stem (T-beams); width of
                +-------------+--------------+---------------------------------

        Returns
        -------
        int
            Returns the assigned section property ID if successful.\n
            Else returns status code indicating the result of the operation:
                - 0 : Library Error: Unable to create property.
                - -106 : One dimensional array of double expected.
                - -108 : Array size is smaller than expected.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> output = staad_obj.Property.CreatePrismaticGeneralProperty([216, 216
        """
        safe_varfaProperties = make_safe_array_double_input(property_value_list
        vt_varfaProperties = make_variant_vt_ref(safe_varfaProperties, automatic
        return self._property.CreatePrismaticGeneralProperty(vt_varfaProperties)
```

[docs]

```
    def CreateTaperedIProperty(self, property_value_list: list):
        """
        Creates tapered I property.

        Parameters
        ----------
        property_value_list : list
            Arrange the values with respect to following table in provided prope
                +-------------+--------------+---------------------------------
                | Array Index | Property Type |
                +=============+==============+=================================
                |     0       |      F1      |Depth of section at start node.
                +-------------+--------------+---------------------------------
                |     1       |      F2      |Thickness of web.
                +-------------+--------------+---------------------------------
                |     2       |      F3      |Depth of section at end node.
                +-------------+--------------+---------------------------------
                |     3       |      F4      |Width of top flange.
                +-------------+--------------+---------------------------------
                |     4       |      F5      |Thickness of top flange.
                +-------------+--------------+---------------------------------
                |     5       |      F6      |Width of bottom flange. Defaults
                +-------------+--------------+--------------------------------
                |     6       |      F7      |Thickness of bottom flange. Defau
                +-------------+--------------+--------------------------------

        Returns
        -------
        int
            Returns the assigned section property ID if successful.\n
            Else returns status code indicating the result of the operation:
                - 0 : Library Error: Unable to create property.
                - -106 : List of double expected.
                - -108 : Length of list is smaller than expected.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> output = staad_obj.Property.CreateTaperedIProperty([13.98, 0.285, 13
        """
        safe_varfaProperties = make_safe_array_double_input(property_value_list
        vt_varfaProperties = make_variant_vt_ref(safe_varfaProperties, automatic
        return self._property.CreateTaperedIProperty(vt_varfaProperties)



                                                                          [docs]
    def CreateTaperedTubeProperty(self, tube_type: int, start_member_section_dep
        """
        Creates tapered tube property.

        Parameters
        ----------
        tube_type : int
            Type of the tube:
                +--------------+-------+
                | Type of Tube | Value |
                +==============+=======+
                | Round        | 0     |
                +--------------+-------+
                | HexDecagonal | 1     |
                +--------------+-------+
                | Dodecagonal  | 2     |
                +--------------+-------+
                | Octagonal    | 3     |
                +--------------+-------+
                | Hexagonal    | 4     |
                +--------------+-------+
                | Square       | 5     |
                +--------------+-------+
        start_member_section_depth : float
             Depth of section at start of member.
        end_member_section_depth : float
            Depth of section at end of member.
        section_thickness : float
            Thickness of section (constant throughout the member length).
        Returns
        -------
        int
            Returns the assigned section property ID if successful.\n
            Else returns status code indicating the result of the operation:
                - 0 : Library Error of being unable to create property.
                - -6008 : Invalid assign profile type.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> output = staad_obj.Property.CreateTaperedTubeProperty(0, 0.5, 0.4, 0
        """
        return self._property.CreateTaperedTubeProperty(tube_type, start_member_
```

[docs]

```
    def CreateAssignProfileProperty(self, profile_type: int):
        """
        Create "Assign Profile" property.

        Parameters
        ----------
        profile_type : int
            Profile type number ID:
                +-----------------+-------+
                | Type of Profile | Value |
                +=================+=======+
                | Angle           | 0     |
                +-----------------+-------+
                | Double Angle    | 1     |
                +-----------------+-------+
                | Beam            | 2     |
                +-----------------+-------+
                | Column          | 3     |
                +-----------------+-------+
                | Channel         | 4     |
                +-----------------+-------+

        Returns
        -------
        int
            Returns the assigned section property ID if successful.\n
            Else returns status code indicating the result of the operation:
                - 0 : Library Error of being unable to create property.
                - -6008 : Invalid assign profile type.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> output = staad_obj.Property.CreateAssignProfileProperty(2)
        """
        return self._property.CreateAssignProfileProperty(profile_type)
```

[docs]

```
    def AssignBetaAngle(self, beam_ids: list, beta_angle: float):
        """
        Assign beta angle to beam(s).

        Parameters
        ----------
        beam_ids : list
            List of beam ids.
        beta_angle : float
            The beta angle in degrees.

        Returns
        -------
        int
            Status code indicating the result of the operation:
                - 1 : OK
                - 0 : General error
                - 0 : List of long expected.
                - 0 : Library Error of being unable to assign BETA angle.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> beam_ids = staad_obj.Geometry.GetBeamList()
        >>> output = staad_obj.Property.AssignBetaAngle([beam_ids[0], beam_ids[
        >>> output = staad_obj.Property.AssignBetaAngle(beam_ids[2], 90.0) # As
        """
        if (isinstance(beam_ids, int)):
            beam_ids = [beam_ids]
        safe_beam_id_list = make_safe_array_long_input(beam_ids)
        vt_beam_ids = make_variant_vt_ref(safe_beam_id_list, automation.VT_ARRA
        return self._property.AssignBetaAngle(vt_beam_ids, beta_angle)
```

[docs]

```
    def CreateMemberTrussSpec(self):
        """
        Create MEMBER TRUSS specification.


        Returns
        -------
        int
            Returns the assigned specification number ID.
            Else returns status code -6010 for unable to create MEMBER TRUSS spe
        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> output = staad_obj.Property.CreateMemberTrussSpec()
        """
        return self._property.CreateMemberTrussSpec()
```

[docs]

```
    def CreateMemberInactiveSpec(self):
        """
        Create MEMBER INACTIVE specification.

        Returns
        -------
        int
            Returns the assigned specification number ID if successful.\n
            Else returns status code '-6011' if it encounters library error(Unab

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> output = staad_obj.Property.CreateMemberInactiveSpec()
        """
        return self._property.CreateMemberInactiveSpec()
```

[docs]

```
    def CreateMemberTensionSpec(self):
        """
        Create MEMBER TENSION specification.

        Returns
        -------
        int
            Returns the assigned specification number ID if successful.\n
            Else returns status code '-6012' if it encounters library error (Una

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> output = staad_obj.Property.CreateMemberTensionSpec()
        """
        return self._property.CreateMemberTensionSpec()
```

[docs]

```
    def CreateMemberCompressionSpec(self):
        """
        Create MEMBER COMPRESSION specification.

        Returns
        -------
        int
            Returns the assigned specification number ID if successful.\n
            Else returns status code '-6013' if library error (Unable to create

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> output = staad_obj.Property.CreateMemberCompressionSpec()
        """
        return self._property.CreateMemberCompressionSpec()
```

[docs]

```
    def CreateMemberIgnoreStiffSpec(self):
        """
        Create MEMBER IGNORE STIFFNESS specification.

        Returns
        -------
        int
            Returns the assigned specification number ID if successful.\n
            Else returns status code '-6014' if library error (Unable to create

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> output = staad_obj.Property.CreateMemberIgnoreStiffSpec()
        """
        return self._property.CreateMemberIgnoreStiffSpec()
```

[docs]

```
    def CreateMemberCableSpec(self, tension_or_unstressed_len: int, spec_value:
        """
        Create MEMBER CABLE specification.

        Parameters
        ----------
        tension_or_unstressed_len : int
            Specify additional information about the cable:
                - 0 = Initial TENSION of Value in the cable to be considered.
                - 1 = Unstressed LENGTH of Value to be considered.
        spec_value : float
            Value for TENSION or Unstressed LENGTH

        Returns
        -------
        int
            Returns the assigned specification number ID if successful.\n
            Else returns status code -6015 if library error (Unable to create ME

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> output = staad_obj.Property.CreateMemberCableSpec(0, 4.5)
        """
        return self._property.CreateMemberCableSpec(tension_or_unstressed_len, 
```

[docs]

```
    def CreateElementPlaneStressSpec(self):
        """
        Create MEMBER PLANE STRESS specification.

        Returns
        -------
        int
            Returns the assigned specification number ID if successful.\n
            Else returns status code '-6018' if library error (Unable to create

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> output = staad_obj.Property.CreateElementPlaneStressSpec()
        """
        return self._property.CreateElementPlaneStressSpec()
```

[docs]

```
    def CreateElementIgnoreInplaneRotnSpec(self):
        """
        Create MEMBER INPLANE ROTATION specification.

        Returns
        -------
        int
            Returns the assigned specification number ID if successful.\n
            Else returns status code '-6019' if library error (Unable to create

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> output = staad_obj.Property.CreateElementIgnoreInplaneRotnSpec()
        """
        return self._property.CreateElementIgnoreInplaneRotnSpec()
```

[docs]

```
    def AssignElementSpecToPlate(self, plate_ids: list, spec_no: int):
        """
        Assign specifications to plate(s).

        Parameters
        ----------
        plate_ids : list
            The plate number ID(s) list
        spec_no : int
            The specification number ID.

        Returns
        -------
        int
            Status code indicating the result of the operation:
                - 0 : OK
                - -106 : List of long expected.
                - -6017 : Library Error: Unable to assign specification.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> plate_ids = staad_obj.Geometry.GetPlateList()
        >>> node_ids = staad_obj.Geometry.GetNodeList()
        >>> specification_number = staad_obj.Property.CreateElementNodeReleaseSp
        >>> output = staad_obj.Property.AssignElementSpecToPlate(plate_ids[0:2],
        >>> output = staad_obj.Property.AssignElementSpecToPlate(plate_ids[3], s
        """
        if (isinstance(plate_ids, int)):
            plate_ids = [plate_ids]
        safe_plate_ids = make_safe_array_long_input(plate_ids)
        vt_plate_ids = make_variant_vt_ref(safe_plate_ids, automation.VT_ARRAY
        return self._property.AssignElementSpecToPlate(vt_plate_ids, spec_no)
```

[docs]

```
    def CreateMemberPartialReleaseSpec(self, location: int, dof_release: list,
        """
        Creates MEMBER RELEASE specification.

        Parameters
        ----------
        location : int
            The offset location at START (= 0) or END (= 1) of the member.
        dof_release : list
            Degrees of freedom: No Release (= 0) or Release (= 1) for FX, FY, FZ
        factor : list
            List of partial release factors arranged in respective DOFs.

        Returns
        -------
        int
            Status code indicating the result of the operation:
                - 0 : OK
                - -106 : List of long for dof_release and list of double for fac
                - -108 : Array size is smaller than expected (size should be 6)
                - -6020 : Library Error: Unable to create MEMBER RELEASE specifi

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> output = staad_obj.Property.CreateMemberPartialReleaseSpec(1, [0, 0,
        """
        safe_varDOFRelease = make_safe_array_long_input(dof_release)
        varDOFRelease = make_variant_vt_ref(safe_varDOFRelease, automation.VT_AR
        safe_varFactor = make_safe_array_double_input(factor)
        varFactor = make_variant_vt_ref(safe_varFactor, automation.VT_ARRAY | au
        return self._property.CreateMemberPartialReleaseSpec(location, varDOFRel
```

[docs]

```
    def CreateElementNodeReleaseSpec(self, node_id: int, dof_release: list):
        """
        Creates ELEMENT NODE RELEASE specification.

        Parameters
        ----------
        node_id : int
            The node number ID to be released.
        dof_release : list
            Degrees of freedom: No Release (=0) or Release (=1) for FX, FY, FZ,

        Returns
        -------
        int
            Gives specification id if successful, else gives status code indicat
                - -106 : List of long type elements for dof_release parameter ex
                - -108 : Array size is smaller than expected (size should be 6)
                - -6020 : Library Error of being unable to create ELEMENT NODE R

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> output = staad_obj.Property.CreateElementNodeReleaseSpec(5, [0, 0, 1
        """
        safe_varDOFRelease = make_safe_array_long_input(dof_release)
        dof_release = make_variant_vt_ref(safe_varDOFRelease, automation.VT_ARR
        return self._property.CreateElementNodeReleaseSpec(node_id, dof_release
```

[docs]

```
    def GetCountryTableNo(self, beam_id: int):
        """
        Get the country Code for the specified member.

        Parameters
        ----------
        beam_id : int
            The beam number ID

        Returns
        -------
        int
            Returns the country CODE if successful.\n
            Else returns status code :
                - -3001 : It is unable to find member.
                - -6022 : No property is attached to the member/element.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> beam_ids = staad_obj.Geometry.GetBeamList()
        >>> country_code = staad_obj.Property.GetCountryTableNo(beam_ids[0])
        """
        return self._property.GetCountryTableNo(beam_id)
```

[docs]

```
    def GetSectionTableNo(self, beam_id: int):
        """
        Get section table number.

        Parameters
        ----------
        beam_id : int
            The beam number ID

        Returns
        -------
        int
            Returns the section table number if successful.\n
            Else returns status code :
                - -3001 : It is unable to find member.
                - -6004 : Section not found in profile database.
                - -6022 : No property is attached to the member/element.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> beam_ids = staad_obj.Geometry.GetBeamList()
        >>> output = staad_obj.Property.GetSectionTableNo(beam_ids[0])
        """
        return self._property.GetSectionTableNo(beam_id)
```

[docs]

```
    def GetBeamSectionName(self, beam_id: int):
        """
        Get beam section string name.

        Parameters
        ----------
        beam_id : int
            The beam number ID

        Returns
        -------
        int
            Returns the section string name. Refer to the table below for probab
                +-----------+-------------------------------------------+--------
                | Sl No.    | Section Type                              | In STD
                +===========+===========================================+========
                | 1         | Standard Section from Steel Database      | | TABLE
                |           |                                           | | TABLE
                |           |                                           | | 5 TAE
                +-----------+-------------------------------------------+--------
                | 2         | Pipe and Tube definition                  | | 8 TAE
                |           |                                           | | 8 TAE
                +-----------+-------------------------------------------+--------
                | 3         | Prismatic                                 | | 3 PRI
                |           |                                           | | 8 PRI
                +-----------+-------------------------------------------+--------
                | 4         | Tapered                                   | 3 TAPER
                +-----------+-------------------------------------------+--------
                | 5         | Assign Profile                            | 3 ASSIG
                +-----------+-------------------------------------------+--------
                | 6         | User Provided Table                       | 14 TO 2
                +-----------+-------------------------------------------+--------

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> beam_ids = staad_obj.Geometry.GetBeamList()
        >>> output = staad_obj.Property.GetBeamSectionName(beam_ids[0])
        """
        return self._property.GetBeamSectionName(beam_id)
```

[docs]

```
def GetBeamSectionPropertyTypeNo(self, beam_id: int):
    """
    Get the section property type number of the specified beam.

    Parameters
    ----------
    beam_id : int
        The beam number ID

    Returns
    -------
    int
        Returns the section property type number if successful else returns
            +----------------------------+------------------------+
            | Section Type               | Property Type Number   |
            +============================+========================+
            | BEAM ST                    | 610                    |
            +----------------------------+------------------------+
            | BEAM D                     | 616                    |
            +----------------------------+------------------------+
            | BEAM TC                    | 613                    |
            +----------------------------+------------------------+
            | BEAM BC                    | 614                    |
            +----------------------------+------------------------+
            | BEAM TB                    | 615                    |
            +----------------------------+------------------------+
            | BEAM T                     | 611                    |
            +----------------------------+------------------------+
            | BEAM CM                    | 612                    |
            +----------------------------+------------------------+
            | CHANNEL ST                 | 630                    |
            +----------------------------+------------------------+
            | CHANNEL D                  | 631                    |
            +----------------------------+------------------------+
            | CHANNEL FR                 | 633                    |
            +----------------------------+------------------------+
            | ANGLE ST                   | 640                    |
            +----------------------------+------------------------+
            | ANGLE LD                   | 642                    |
            +----------------------------+------------------------+
            | ANGLE SD                   | 643                    |
            +----------------------------+------------------------+
            | ANGLE RA                   | 641                    |
            +----------------------------+------------------------+
            | ANGLE SA                   | 646                    |
            +----------------------------+------------------------+
            | PIPE ST                    | 660                    |
            +----------------------------+------------------------+
            | HSS RECTANGLE              | 654                    |
            +----------------------------+------------------------+
            | HSS ROUND                  | 655                    |
            +----------------------------+------------------------+
            | CASTEL ST                  | 656                    |
            +----------------------------+------------------------+
            | TUBE ST                    | 650                    |
            +----------------------------+------------------------+
            | TEE ST                     | 620                    |
            +----------------------------+------------------------+
            | PLATE STRIP                | 666                    |
            +----------------------------+------------------------+
            | ANGLE COLD ST              | 644                    |
            +----------------------------+------------------------+
            | ANGLE COLD ST WITH LIPS    | 645                    |
            +----------------------------+------------------------+
            | CHANNEL COLD ST            | 634                    |
            +----------------------------+------------------------+
            | CHANNEL COLD ST WITH LIPS  | 635                    |
            +----------------------------+------------------------+
            | ZEE COLD ST                | 662                    |
            +----------------------------+------------------------+
            | ZEE COLD ST  WITH LIPS     | 663                    |
            +----------------------------+------------------------+
            | HAT COLD ST                | 664                    |
            +----------------------------+------------------------+
            | TAPER                      | 680                    |
            +----------------------------+------------------------+
            | TAPERED TUBE               | 675                    |
            +----------------------------+------------------------+
            | PRISMATIC CIRCLE           | 671                    |
            +----------------------------+------------------------+
            | PRISMATIC RECT             | 672                    |
            +----------------------------+------------------------+
            | PRISMATIC TRAP             | 674                    |
            +----------------------------+------------------------+
            | PRISMATIC TEE              | 673                    |
            +----------------------------+------------------------+
            | PRISMATIC GENERAL          | 676                    |
            +----------------------------+------------------------+
            | SOLID ROUND                | 668                    |
            +----------------------------+------------------------+
            | UPT PRISMATIC              | 699                    |
            +----------------------------+------------------------+
            | UPT GENERAL                | 697                    |
            +----------------------------+------------------------+
            | UPT WIDE FLANGE            | 690                    |
            +----------------------------+------------------------+
            | UPT CHANNEL                | 691                    |
            +----------------------------+------------------------+
            | UPT ANGLE                  | 692                    |
            +----------------------------+------------------------+
            | UPT DOUBLE ANGLE           | 693                    |
            +----------------------------+------------------------+
            | UPT TEE                    | 694                    |
            +----------------------------+------------------------+
            | UPT PIPE                   | 695                    |
            +----------------------------+------------------------+
            | UPT TUBE                   | 696                    |
            +----------------------------+------------------------+
            | UPT ISECTION               | 698                    |
            +----------------------------+------------------------+
    Examples
    --------
    >>> from openstaadpy import os_analytical
    >>> staad_obj = os_analytical.connect()
    >>> beam_ids = staad_obj.Geometry.GetBeamList()
    >>> output = staad_obj.Property.GetBeamSectionPropertyTypeNo(beam_ids[0
    """
    return self._property.GetBeamSectionPropertyTypeNo(beam_id)
```

[docs]

```
    def GetBeamConstants(self, beam_id: int):
        """
        Retrieve beta angle of the specified beam member.

        Parameters
        ----------
        beam_id : int
            The beam number ID

        Returns
        -------
        Tuple
            Returns a tuple of Beam Constants found in following order Elasticit

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> beam_ids = staad_obj.Geometry.GetBeamList()
        >>> elasticity, poisson, density, alpha, damp = staad_obj.Property.GetBe
        """
        safe_Elasticity = make_safe_array_double(1)
        Elasticity = make_variant_vt_ref(safe_Elasticity,  automation.VT_R8)

        safe_Poisson = make_safe_array_double(1)
        Poisson = make_variant_vt_ref(safe_Poisson,  automation.VT_R8)

        safe_Density = make_safe_array_double(1)
        Density = make_variant_vt_ref(safe_Density,  automation.VT_R8)

        safe_Alpha = make_safe_array_double(1)
        Alpha = make_variant_vt_ref(safe_Alpha,  automation.VT_R8)

        safe_Damp = make_safe_array_double(1)
        Damp = make_variant_vt_ref(safe_Damp,  automation.VT_R8)

        result = self._property.GetBeamConstants(beam_id, Elasticity, Poisson, 
        if not result:
            raise_os_error_if_error_code(-1)
        return Elasticity[0], Poisson[0], Density[0], Alpha[0], Damp[0]
```

[docs]

```
def GetBetaAngle(self, beam_id: int):
    """
    Retrieves beta angle of the specified beam member.

    Parameters
    ----------
    beam_id : int
        The beam number ID

    Returns
    -------
    int
        Returns Beta angle else returns status code '-3001' if it can't fin

    Examples
    --------
    >>> from openstaadpy import os_analytical
    >>> staad_obj = os_analytical.connect()
    >>> beam_ids = staad_obj.Geometry.GetBeamList()
    >>> output = staad_obj.Property.GetBetaAngle(beam_ids[0])
    """
    result = self._property.GetBetaAngle(beam_id)
    if result < 0:
        raise_os_error_if_error_code(result)
    return result
```

[docs]

```
def GetSectionPropertyCount(self):
    """
    Returns total number of different sectional properties exist in the cur

    Returns
    -------
    int
        Returns total number of different sectional properties.

    Examples
    --------
    >>> from openstaadpy import os_analytical
    >>> staad_obj = os_analytical.connect()
    >>> count = staad_obj.Property.GetSectionPropertyCount()
    """
    return self._property.GetSectionPropertyCount()
```

[docs]

```
    def GetSectionPropertyName(self, sctn_prop_id: int):
        """
        Get the property name for the specified section property reference numbe

        Parameters
        ----------
        sctn_prop_id : int
            The assigned section property ID

        Returns
        -------
        string
            Returns a string for identification title of material.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> nAssignedSectionPropID = staad_obj.Property.CreateBeamPropertyFromTa
        >>> output = staad_obj.Property.GetSectionPropertyName(nAssignedSectionP
        """
        section_property_name = create_bstr()
        ref_section_property_name = make_byref(section_property_name)
        result = self._property.GetSectionPropertyName(sctn_prop_id, ref_sectio
        if not result:
            raise_os_error_if_error_code(-1)
        return section_property_name.value
```

[docs]

```
    def GetSectionPropertyType(self, sec_ref_no: int):
        """
        Returns the section property type for the specified section property ref

        Parameters
        ----------
        sec_ref_no : int
            The assigned section property ID

        Returns
        -------
        int
            Returns number referring to section type code table.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> nAssignedSectionPropID = staad_obj.Property.CreateBeamPropertyFromTa
        >>> output = staad_obj.Property.GetSectionPropertyType(nAssignedSectionP
        """
        result = self._property.GetSectionPropertyType(sec_ref_no)
        if result < 0:
            raise_os_error_if_error_code(result)
        return result
```

[docs]

```
    def GetSectionPropertyCountry(self, sec_ref_no: int):
        """
        Returns the country reference number for the section property reference

        Parameters
        ----------
        sec_ref_no : int
            The assigned section property ID

        Returns
        -------
        int
            Returns country code else returns -6025 if no property is defined ir

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> nAssignedSectionPropID = staad_obj.Property.CreateBeamPropertyFromTa
        >>> output = staad_obj.Property.GetSectionPropertyCountry(nAssignedSecti
        """
        return self._property.GetSectionPropertyCountry(sec_ref_no)
```

[docs]

```
    def GetIsotropicMaterialCount(self):
        """
        Gets the number of isotropic material present in the current structure.

        Returns
        -------
        int
            Returns the number of isotropic materials.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> output = staad_obj.Property.GetIsotropicMaterialCount()
        """
        return self._property.GetIsotropicMaterialCount()
```

[docs]

```
    def GetIsotropicMaterialProperties(self, material_number: int):
        """
        Get the properties for the specified isotropic material number.

        Parameters
        ----------
        material_number : int
            Zero based index of the material

        Returns
        -------
        tuple : Tuple <str, float, float, float, float, float, float>
            Returns a tuple consisting of Material Name, Modulus of elasticity,

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> isotropic_mat_no = staad_obj.Property.GetIsotropicMaterialCount()
        >>> if isotropic_mat_no > 0:
        >>>     material, elasticity, poisson, shear_mod, density, coef_thermal_
        """
        safe_varE = make_safe_array_double(1)
        vt_varE = make_variant_vt_ref(safe_varE,  automation.VT_R8)
        safe_varPoisson = make_safe_array_double(1)
        vt_varPoisson = make_variant_vt_ref(safe_varPoisson,  automation.VT_R8)
        safe_varG = make_safe_array_double(1)
        vt_varG = make_variant_vt_ref(safe_varG,  automation.VT_R8)
        safe_varDensity = make_safe_array_double(1)
        vt_varDensity = make_variant_vt_ref(safe_varDensity,  automation.VT_R8)
        safe_varAlpha = make_safe_array_double(1)
        vt_varAlpha = make_variant_vt_ref(safe_varAlpha,  automation.VT_R8)
        safe_varCrDamp = make_safe_array_double(1)
        vt_varCrDamp = make_variant_vt_ref(safe_varCrDamp,  automation.VT_R8)
        result = self._property.GetIsotropicMaterialProperties(material_number,
        if result == "":
            raise_os_error_if_error_code(-6023)
        return result, vt_varE[0], vt_varPoisson[0], vt_varG[0], vt_varDensity[
```

[docs]

```
    def GetOrthotropic2DMaterialCount(self):
        """
        Return the number of 2D orthotropic material present in the current stru

        Returns
        -------
        int
            Returns the number of 2D orthotropic material.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> ortho_2d_mat_count = staad_obj.Property.GetOrthotropic2DMaterialCou
        """
        return self._property.GetOrthotropic2DMaterialCount()
```

[docs]

```
    def GetOrthotropic2DMaterialProperties(self, material_no: int):
        """
        Get the properties for the specified 2D orthotropic material.

        Parameters
        ----------
        material_no : int
            Material Number ID

        Returns
        -------
        tuple: Tuple(float, float, float, float, float, float)
            Returns a tuple consisting of Modulus of elasticity, Poisson's ratic

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> ortho_2d_mat_count = staad_obj.Property.GetOrthotropic2DMaterialCour
        >>> if ortho_2d_mat_count > 0:
        >>>     elasticity, poisson, shear_mod, density, coef_thermal_exp, damp_
        """
        safe_varE = make_safe_array_double(1)
        vt_varE = make_variant_vt_ref(safe_varE,  automation.VT_R8)
        safe_varPoisson = make_safe_array_double(1)
        vt_varPoisson = make_variant_vt_ref(safe_varPoisson,  automation.VT_R8)
        safe_varG = make_safe_array_double(1)
        vt_varG = make_variant_vt_ref(safe_varG,  automation.VT_R8)
        safe_varDensity = make_safe_array_double(1)
        vt_varDensity = make_variant_vt_ref(safe_varDensity,  automation.VT_R8)
        safe_varAlpha = make_safe_array_double(1)
        vt_varAlpha = make_variant_vt_ref(safe_varAlpha,  automation.VT_R8)
        safe_varCrDamp = make_safe_array_double(1)
        vt_varCrDamp = make_variant_vt_ref(safe_varCrDamp,  automation.VT_R8)
        result = self._property.GetOrthotropic2DMaterialProperties(material_no,
        if result == "":
            raise_os_error_if_error_code(-6023)
        return vt_varE[0], vt_varPoisson[0], vt_varG[0], vt_varDensity[0], vt_v
```

[docs]

```
    def GetOrthotropic3DMaterialCount(self):
        """
        Gets orthotropic 3D material count.

        Returns
        -------
        int
            Returns the orthotropic 3D material count.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> ortho_3d_mat_count = staad_obj.PropertyGetOrthotropic3DMaterialCount
        """
        return self._property.GetOrthotropic3DMaterialCount()
```

[docs]

```
    def GetOrthotropic3DMaterialProperties(self, material_no: int):
        """
        Get the properties for the specified 3D orthotropic material.

        Parameters
        ----------
        material_no : int
            Material Number ID

        Returns
        -------
        tuple : Tuple(float, float, float, float, float, float)
            Returns a tuple consisting of Modulus of elasticity, Poisson's ratio

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> ortho_3d_mat_count = staad_obj.Property.GetOrthotropic3DMaterialCour
        >>> if ortho_3d_mat_count > 0:
        >>>     elasticity, poisson, shear_mod, density, coef_thermal_exp, damp_
        """
        safe_varE = make_safe_array_double(1)
        vt_varE = make_variant_vt_ref(safe_varE,  automation.VT_R8)
        safe_varPoisson = make_safe_array_double(1)
        vt_varPoisson = make_variant_vt_ref(safe_varPoisson,  automation.VT_R8)
        safe_varG = make_safe_array_double(1)
        vt_varG = make_variant_vt_ref(safe_varG,  automation.VT_R8)
        safe_varDensity = make_safe_array_double(1)
        vt_varDensity = make_variant_vt_ref(safe_varDensity,  automation.VT_R8)
        safe_varAlpha = make_safe_array_double(1)
        vt_varAlpha = make_variant_vt_ref(safe_varAlpha,  automation.VT_R8)
        safe_varCrDamp = make_safe_array_double(1)
        vt_varCrDamp = make_variant_vt_ref(safe_varCrDamp,  automation.VT_R8)
        result = self._property.GetOrthotropic3DMaterialProperties(material_no,
        if not result:
            raise_os_error_if_error_code(-1)
        return vt_varE[0], vt_varPoisson[0], vt_varG[0], vt_varDensity[0], vt_v
```

[docs]

```
    def GetMemberGlobalOffSet(self, beam_id: int, member_offset_position: int):
        """
        Get beam end offsets in all three global directions.

        Parameters
        ----------
        beam_id : int
            The beam number ID
        member_offset_position : int
            Member Start position (= 0); member End position (= 1).

        Returns
        -------
        tuple : Tuple(float, float, float)
            Returns a tuple consisting of member End position (= 1), the offset

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> beam_ids = staad_obj.Geometry.GetBeamList()
        >>> offset_x, offset_y, offset_z = staad_obj.Property.GetMemberGlobalOff
        """
        safe_varfxOffSet = make_safe_array_double(1)
        vt_varfxOffSet = make_variant_vt_ref(safe_varfxOffSet,  automation.VT_R8
        safe_varfyOffSet = make_safe_array_double(1)
        vt_varfyOffSet = make_variant_vt_ref(safe_varfyOffSet,  automation.VT_R8
        safe_varfzOffSet = make_safe_array_double(1)
        vt_varfzOffSet = make_variant_vt_ref(safe_varfzOffSet,  automation.VT_R8
        result = self._property.GetMemberGlobalOffSet(beam_id, member_offset_pos
        if result < 0:
            raise_os_error_if_error_code(result)
        return vt_varfxOffSet[0], vt_varfyOffSet[0], vt_varfzOffSet[0]
```

[docs]

```
    def GetMemberLocalOffSet(self, beam_id: int, member_offset_position: int):
        """
        Get beam end offsets in all three local directions.

        Parameters
        ----------
        beam_id : int
            The beam number ID
        member_offset_position : int
            Member Start position (= 0); member End position (= 1).

        Returns
        -------
        List
            Returns a List consisting of member End position (= 1), the offset x
        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> beam_ids = staad_obj.Geometry.GetBeamList()
        >>> offset_x, offset_y, offset_z = staad_obj.Property.GetMemberLocalOffS
        """
        safe_varfxOffSet = make_safe_array_double(1)
        vt_varfxOffSet = make_variant_vt_ref(safe_varfxOffSet,  automation.VT_R8
        safe_varfyOffSet = make_safe_array_double(1)
        vt_varfyOffSet = make_variant_vt_ref(safe_varfyOffSet,  automation.VT_R8
        safe_varfzOffSet = make_safe_array_double(1)
        vt_varfzOffSet = make_variant_vt_ref(safe_varfzOffSet,  automation.VT_R8
        result = self._property.GetMemberLocalOffSet(beam_id, member_offset_pos
        if result < 0:
            raise_os_error_if_error_code(result)
        return vt_varfxOffSet[0], vt_varfyOffSet[0], vt_varfzOffSet[0]
```

[docs]

```
    def GetIsotropicMaterialPropertiesAssigned(self, material_no: int):
        """
        Gets isotropic material properties and if material assigned to element(s

        Parameters
        ----------
        material_no : int
            Material number ID

        Returns
        -------
        tuple : Tuple <str, float, float, float, float, float, float, int>
            Returns a Tuple consisting of material name, modulus of elasticity,

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> isotropic_mat_no = staad_obj.Property.GetIsotropicMaterialCount()
        >>> if isotropic_mat_no > 0:
        >>>     elasticity, poisson, shear_modulus, weight_density, thermal_expa
        """
        safe_varE = make_safe_array_double(1)
        vt_varE = make_variant_vt_ref(safe_varE,  automation.VT_R8)
        safe_varPoisson = make_safe_array_double(1)
        vt_varPoisson = make_variant_vt_ref(safe_varPoisson,  automation.VT_R8)
        safe_varG = make_safe_array_double(1)
        vt_varG = make_variant_vt_ref(safe_varG,  automation.VT_R8)
        safe_varDensity = make_safe_array_double(1)
        vt_varDensity = make_variant_vt_ref(safe_varDensity,  automation.VT_R8)
        safe_varAlpha = make_safe_array_double(1)
        vt_varAlpha = make_variant_vt_ref(safe_varAlpha,  automation.VT_R8)
        safe_varCrDamp = make_safe_array_double(1)
        vt_varCrDamp = make_variant_vt_ref(safe_varCrDamp,  automation.VT_R8)
        safe_varAssigned = make_safe_array_long(1)
        vt_varAssigned = make_variant_vt_ref(safe_varAssigned,  automation.VT_I4
        material_name = self._property.GetIsotropicMaterialPropertiesAssigned(ma
        if material_name == "":
            raise_os_error_if_error_code(-6023)
        return material_name,vt_varE[0], vt_varPoisson[0], vt_varG[0], vt_varDen
```

[docs]

```
    def AddControlDependentRelation(self, control_node: int, rigid_type: int, fx
        """
        Add a control/dependent joint specification to specified node(s).

        Parameters
        ----------
        control_node : int
            Set node (number ID) control node.
        rigid_type : int
            Set plate rigid: all directions rigid (= 0), XY plate rigid (= 1), Y
        fx : int
            Rigid in X direction translation (Rigid = 1, Not Rigid = 0)
        fy : int
            Rigid in Y direction translation (Rigid = 1, Not Rigid = 0)
        fz : int
            Rigid in Z direction translation (Rigid = 1, Not Rigid = 0)
        mx : int
            Rigid in X direction rotation (Rigid = 1, Not Rigid = 0)
        my : int
            Rigid in Y direction rotation (Rigid = 1, Not Rigid = 0)
        mz : int
            Rigid in Z direction rotation (Rigid = 1, Not Rigid = 0)
        dependent_node_list : list
            Nodes number ID list

        Returns
        -------
        int
            Returns 0 if successful.\n
            Returns -106 if list of long expected.\n
            Returns -6029 if Library Error: Unable to create CONTROL/DEPENDENT s

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> result = staad_obj.Property.AddControlDependentRelation(3, -1, 1, 1,
        """
        safe_dependent_node_list = make_safe_array_long_input(dependent_node_lis
        vt_dependant_node_list = make_variant_vt_ref(safe_dependent_node_list,
        result = self._property.AddControlDependentRelation(control_node, rigid_
        if result < 0:
            raise_os_error_if_error_code(result)
        return result
```

[docs]

```
def CreateIsotropicMaterialProperties(self, material_name: str, elasticity_
    """
    Creates isotropic material properties.

    Parameters
    ----------
    material_name : str
        Material Name
    elasticity_mod : float
        Modulus of elasticity List (of size 3).
    poisson : float
        Poisson's ratio List (of size 3).
    shear_mod : float
        Shear modulus List (of size 3).
    density : float
        Weight density List (of size 3).
    coef_thermal_exp : float
        Coefficient of thermal expansion List (of size 3).
    damp_ratio : float
        Damping ratio List (of size 3).

    Returns
    -------
    int
        Returns 1 if Material is updated as a material with that name was al
        Returns 0 if Material is created.
        Returns -1 if General Error.

    Examples
    --------
    >>> from openstaadpy import os_analytical
    >>> staad_obj = os_analytical.connect()
    >>> result = staad_obj.Property.CreateIsotropicMaterialProperties("MATER
    """
    return self._property.CreateIsotropicMaterialProperties(material_name,
```

[docs]

```
def CreateUPTTable(self, table_type: int):
    """
    Creates user provided table (UPT).

    Parameters
    ----------
    table_type : int
        Type of the table:
            +------+----------------------------+
            |  No. |         Table Type         |
            +======+============================+
            |  1   | scUserTableWideFlangeTitle  |
            +------+-----------------------------+
            |  2   | scUserTableChannelTitle     |
            +------+-----------------------------+
            |  3   | scUserTableAngleTitle       |
            +------+-----------------------------+
            |  4   | scUserTableDoubleAngleTitle |
            +------+-----------------------------+
            |  5   | scUserTableTeeTitle         |
            +------+-----------------------------+
            |  6   | scUserTablePipeTitle        |
            +------+-----------------------------+
            |  7   | scUserTableTubeTitle        |
            +------+-----------------------------+
            |  8   | scUserTableGeneralTitle     |
            +------+-----------------------------+
            |  9   | scUserTableIsectionTitle    |
            +------+-----------------------------+
            |  10  | scUserTablePrismaticTitle   |
            +------+-----------------------------+

    Returns
    -------
    int
        Returns User Provided Table (UPT) number id else -6031 if it is unab

    Examples
    --------
    >>> from openstaadpy import os_analytical
    >>> staad_obj = os_analytical.connect()
    >>> result = staad_obj.Property.CreateUPTTable(1)
    """
    return self._property.CreateUPTTable(table_type)
```

[docs]

```
    def RemoveUPTTable(self, table_ref_id: int):
        """
        Remove the whole User Provided Table (UPT) specified by table number ID

        Parameters
        ----------
        table_ref_id : int
            The existing table number ID

        Returns
        -------
        int
            Returns 'True' if successful else 'False' if general error.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> result = staad_obj.Property.RemoveUPTTable(1)
        """
        return self._property.RemoveUPTTable(table_ref_id)
```

[docs]

```
    def AddUPTPropertyWIDEFLANGE(self, table_ref_id: int, stn_name: str, cro_se
        """
        Add wide flange type to an defined UPT section.

        Parameters
        ----------
        table_ref_id : int
            The existing table number ID.
        stn_name : str
            UPT section string name.
        cro_sec_area : float
            Cross section area.
        sectn_depth : float
            Depth of the section.
        web_Thickness : float
            Thickness of web.
        top_flange_width : float
            Width of the top flange.
        top_flange_thickness : float
            Thickness of top flange.
        torsional_constant : float
            Torsional constant.
        moi_l_y : float
            Moment of inertia about local y-axis.
        moi_l_z : float
            Moment of inertia about local z-axis.
        shear_area_y : float
            Shear area in local y-axis. If zero, shear deformation is ignored ir
        shear_area_z : float
            Shear area in local z-axis. If zero, shear deformation is ignored ir

        Returns
        -------
        int
            Returns 0 if OK.\n
            Returns -6032 if unable to add section stn_name in UPT table_referer
            Returns -6045 if a section with the same section_name already exists

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> upt_num_id = staad_obj.Property.CreateUPTTable(1)
        >>> result = staad_obj.Property.AddUPTPropertyWIDEFLANGE(upt_num_id, "VI
        """
        return self._property.AddUPTPropertyWIDEFLANGE(table_ref_id, stn_name, 
```

[docs]

```
    def AddUPTPropertyCHANNEL(self, table_reference_id: int, stn_name: str, cro
        """
        Add channel type to an defined UPT section.

        Parameters
        ----------
        table_reference_id : int
            The existing table number ID.
        stn_name : str
            UPT section string name.
        cro_sec_area : float
            Cross section area.
        sectn_depth : float
            Depth of the section.
        web_Thickness : float
            Thickness of web.
        top_flange_width : float
            Width of the top flange.
        top_flange_thickness : float
            Thickness of top flange.
        torsional_constant : float
            Torsional constant.
        moi_l_y : float
            Moment of inertia about local y-axis.
        moi_l_z : float
            Moment of inertia about local z-axis.
        c_z : float
            CZ value.
        shear_area_y : float
            Shear area in local y-axis. If zero, shear deformation is ignored ir
        shear_area_z : float
            Shear area in local z-axis. If zero, shear deformation is ignored ir

        Returns
        -------
        int
            Returns 0 if OK.
            Returns -6032 if unable to add section stn_name in UPT table_referer
            Returns -6045 if a section with the same section_name already exists

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> upt_num_id = staad_obj.Property.CreateUPTTable(2)
        >>> result = staad_obj.Property.AddUPTPropertyCHANNEL(upt_num_id, "VJG2(
        """
        return self._property.AddUPTPropertyCHANNEL(table_reference_id, stn_name
```

[docs]

```
    def AddUPTPropertyANGLE(self, table_reference_id: int, section_name: str, de
        """
        Add angle type to an defined UPT section.

        Parameters
        ----------
        table_reference_id : int
            The existing table number ID.
        section_name : str
            UPT section string name.
        depth_of_angle : float
            Depth of angle.
        width_of_angle : float
            Width of angle.
        flange_thickness : float
            Thickness of flange (TF).
        gyration_radius : float
            Radius of gyration about principal axis.
        shear_area_y: float
            Shear area in local y-axis. If zero, shear deformation is ignored ir
        shear_area_z: float
            Shear area in local z-axis. If zero, shear deformation is ignored ir

        Returns
        -------
        int
            Returns 0 if OK.\n
            Returns -6032 if the section with section_name cannot be added to th
            Returns -6045 if a section with the same section_name already exists

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> upt_num_id = staad_obj.Property.CreateUPTTable(3)
        >>> status = staad_obj.Property.AddUPTPropertyANGLE(upt_num_id, "UPT_Cha
        """
        return self._property.AddUPTPropertyANGLE(table_reference_id, section_na
```

[docs]

```
    def AddUPTPropertyDOUBLEANGLE(self, table_reference_id: int, section_name: 
        """
        Add double angle type to an defined UPT section.

        Parameters
        ----------
        table_reference_id : int
            The existing table number ID.
        section_name : str
            UPT section string name.
        depth_angle : float
            Depth of angle.
        width_angle : float
```

```
            Width of angle.
        flanges_thickness : float
            Thickness of flanges.
        distance_between_two_angles : float
            Distance between two angles.
        torsional_constant : float
            Torsional constant
        moi_y : float
            Moment of inertia about local y-axis.
        moi_z : float
            Moment of inertia about local z-axis.
        dist_z_top_section : float
            Distance from z axis to the top of section.
        shear_area_y : float
            Shear area in local y-axis. If zero, shear deformation is ignored in
        shear_area_z : float
            Shear area in local z-axis. If zero, shear deformation is ignored in

        Returns
        -------
        int
            Returns 0 if OK.\n
            Returns -6032 if the section with section_name cannot be added to th
            Returns -6045 if a section with the same section_name already exists

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> upt_num_id = staad_obj.Property.CreateUPTTable(4)
        >>> result = staad_obj.Property.AddUPTPropertyDOUBLEANGLE(upt_num_id, "V
        """
        return self._property.AddUPTPropertyDOUBLEANGLE(table_reference_id, sec
```

[docs]

```
    def AddUPTPropertyTEE(self, table_reference_id: int, section_name: str, cros
        """
        Add tee type to a defined UPT section.

        Parameters
        ----------
        table_reference_id : int
            The existing table number ID.
        section_name : str
            UPT section string name.
        cross_section_area : float
            Cross section area (AX).
        section_depth : float
            Depth of the section (D).
        top_flange_width : float
            Width of the top flange (WF).
        top_flange_thickness : float
            Thickness of top flange (TF).
        web_thickness : float
            Thickness of web (TW).
        torsional_constant : float
            Torsional constant (IZ).
        moi_y : float
            Moment of inertia about local y-axis (IY).
        moi_z : float
            Moment of inertia about local z-axis (IZ).
        dist_z_top_section : float
            Distance from z axis to the top of section.
        shear_area_y : float
            Shear area in local Y-axis. If zero, shear deformation is ignored ir
        shear_area_z : float
            Shear area in local Z-axis. If zero, shear deformation is ignored ir

        Returns
        -------
        int
            Returns 0 if OK.\n
            Returns -6032 if the section with section_name cannot be added to th
            Returns -6045 if a section with the same section_name already exists


        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> upt_num_id = staad_obj.Property.CreateUPTTable(5)
        >>> result = staad_obj.Property.AddUPTPropertyTEE(upt_num_id, "VJG20-2",
        """
        return self._property.AddUPTPropertyTEE(table_reference_id, section_name
```

[docs]

```
    def AddUPTPropertyPIPE(self, table_reference_id: int, section_name: str, ou
        """
        Add pipe type to a defined UPT section.

        Parameters
        ----------
        table_reference_id : int
            The existing table number ID.
        section_name : str
            UPT section string name.
        out_diameter : float
            Outer diameter (OD).
        in_diameter : float
            Inner diameter (ID).
        shear_area_y : float
            Shear area in local y-axis. If zero, shear deformation is ignored ir
        shear_area_z : float
            Shear area in local z-axis. If zero, shear deformation is ignored ir

        Returns
        -------
        int
            Return 0 if ok.\n
            Return -6032 if the section with section_name cannot be added to the
            Return -6045 if a section with the same section_name already exists

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> upt_num_id = staad_obj.Property.CreateUPTTable(6)
        >>> result = staad_obj.Property.AddUPTPropertyPIPE(upt_num_id, "VJG20-2'
        """
        return self._property.AddUPTPropertyPIPE(table_reference_id, section_nar
```

[docs]

```
    def AddUPTPropertyTUBE(self, table_reference_id: int, section_name: str, cr
        """
        Add tube type to an defined UPT section.

        Parameters
        ----------
        table_reference_id : int
            The existing table number ID.
        section_name : str
            UPT section string name.
        cross_section_area : float
            Cross section area (AX).
        section_depth : float
            Depth of the section (D).
        top_flange_width : float
            Width of the top flange (WF).
        top_flange_thickness : float
            Thickness of top flange (TF).
        torsional_constant : float
            Torsional constant (Iz).
        moi_y : float
            Moment of inertia about local y-axis (IY).
        moi_z : float
            Moment of inertia about local z-axis (IX).
        shear_area_y : float
            Shear area in local y-axis. If zero, shear deformation is ignored ir
        shear_area_z : float
            Shear area in local z-axis. If zero, shear deformation is ignored ir

        Returns
        -------
        int
            Returns 0 OK.\n
            Returns -6032 if unable to add section section_name in upt table_ret
            Returns -6045 if a section with the same section_name already exists

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> upt_num_id = staad_obj.Property.CreateUPTTable(7)
        >>> result = staad_obj.Property.AddUPTPropertyTUBE(upt_num_id, "VJG20-2'
        """
        return self._property.AddUPTPropertyTUBE(table_reference_id, section_nar
```

[docs]

```
    def AddUPTPropertyGENERAL(self, table_reference_id: float, section_name: flo
        """
        Add general type to an defined UPT section.

        Parameters
        ----------
        table_reference_id : int
            The existing table number ID.
        section_name : str
            UPT section string name.
        cross_section_area : float
            Cross section area (AX).
        section_depth : float
            Depth of the section (D).
        thickness_parallel_depth : float
            Thickness associated with section element parallel to depth (TD).
        width_of_section : float
            Width of section (B).
        thickness_parallel_flange : float
            Thickness associated with section element parallel to flange(TB).
        torsional_constant : float
            Torsional constant (IZ).
        moi_y : float
            Moment of inertia about local y-axis (IY).
        moi_z : float
            Moment of inertia about local z-axis (IZ).
        section_modulus_z : float
            Section modulus about local Z-axis (SZ).
        section_modulus_y : float
            Section modulus about local Y-axis (SY).
        shear_area_y : float
            Shear area for shear parallel to local Y-axis (AY).
        shear_area_z : float
            Shear area for shear parallel to local Z-axis (AZ).
        plastic_modulus_z : float
            Plastic modulus about local Z-axis (PZ).
        plastic_modulus_y : float
            Plastic modulus about local Y-axis (PY).
        warping_constant : float
            Warping constant for lateral torsional buckling calculations (HSS).
        depth_of_web : float
            Depth of web. For rolled sections, distance between fillets should b
        Returns
        -------
        int
            Returns 0 OK.\n
            Returns -6032 if the section with section_name cannot be added to th
            Returns -6045 if a section with the same section_name already exists

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> upt_num_id = staad_obj.Property.CreateUPTTable(8)
        >>> result = staad_obj.Property.AddUPTPropertyGENERAL(upt_num_id, "VJG2
        """
        return self._property.AddUPTPropertyGENERAL(table_reference_id, section_
```

[docs]

```
    def AddUPTPropertyISECTION(self, table_reference_id: int, section_name: str
        """
        Add I type to an defined UPT section.

        Parameters
        ----------
        table_reference_id : int
            The existing table number ID.
        section_name : str
            UPT section string name.
        depth_of_web : float
            Depth of section at start node(DWW).
        thickness_of_web : float
            Thickness of web(TWW).
        depth_of_web1 : float
            Depth of section at end node(DWW1).
        width_of_top_flange : float
            Width of top flange(BFF).
        thickness_of_top_flange : float
            Thickness of top flange(TFF).
        width_of_bottom_flange : float
            Width of bottom flange(BFF1).
        thickness_of_bottom_flange : float
            Thickness of bottom flange(TFF1).
        shear_area_y : float
            Shear area for shear parallel to Y-axis(AYF).
        shear_area_z : float
            Shear area for shear parallel to Z-axis(AZF).
        torsional_constant : float
            Torsional constant (XIF).

        Returns
        -------
        int
            Returns 0 OK.\n
            Returns -6032 if the section with section_name cannot be added to th
            Returns -6045 if a section with the same section_name already exists

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> upt_num_id = staad_obj.Property.CreateUPTTable(9)
        >>> result = staad_obj.Property.AddUPTPropertyISECTION(upt_num_id, "VJG2
        """
        return self._property.AddUPTPropertyISECTION(table_reference_id, sectio
```

[docs]

```
    def AddUPTPropertyPRISMATIC(self, table_reference_id: int, section_name: st
        """
        Add PRISMATIC type to an defined UPT section.

        Parameters
        ----------
        table_reference_id : int
            The existing table number ID.
        section_name : str
            UPT section string name.
        cross_section_area : float
            Cross section area (AX).
        torsional_constant : float
            Torsional constant (IZ).
        moment_of_inertia_y : float
            Moment of inertia about local y-axis (IY).
        moment_of_inertia_z : float
            Moment of inertia about local z-axis (IZ).
        shear_area_y : float
            Shear area for shear parallel to local Y-axis (AY).
        shear_area_z : float
            Shear area for shear parallel to local Z-axis (AZ).
        depth_y : float
            Depth of the section in the direction of the local Y-axis (YD).
        depth_z : float
            Depth of the section in the direction of the local Z-axis (ZD).

        Returns
        -------
        int
            Returns 0 OK.\n
            Returns -6032 if the section with section_name cannot be added to th
            Returns -6045 if a section with the same section_name already exists

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> upt_num_id = staad_obj.Property.CreateUPTTable(10)
        >>> result = staad_obj.Property.AddUPTPropertyPRISMATIC(upt_num_id, "VJG
        """
        return self._property.AddUPTPropertyPRISMATIC(table_reference_id, sectio
```

[docs]

```
    def RemovePropertyFromUPTTable(self, table_reference_id: int, section_name:
        """
        Remove a property from User Provided Table (UPT) if exist.

        Parameters
        ----------
        table_reference_id : int
            The existing table number ID.
        section_name : str
            UPT section string name.

        Returns
        -------
        int
            Returns 1 if successful.
            Returns 0 if general error.
            Returns -100 if invalid table number or section name.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> upt_num_id = staad_obj.Property.CreateUPTTable(7)
        >>> result = staad_obj.Property.RemovePropertyFromUPTTable(upt_num_id, '
        """
        return self._property.RemovePropertyFromUPTTable(table_reference_id, se
```

[docs]

```
    def CreateMemberAttribute(self, attribute_name: str, str_Value: str):
        """
        Create member attribute by name.

        Parameters
        ----------
        attribute_name : str
            Name of the attribute.
        str_Value : str
            A string value

        Returns
        -------
        int
            Returns 0 if successful else -1 if general error.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> result = staad_obj.Property.CreateMemberAttribute("MEMBTYPE", "BRACE
        """
        return self._property.CreateMemberAttribute(attribute_name, str_Value)
```

[docs]

```
    def AssignMemberAttribute(self, attribute_name: str, str_Value: str, member_
        """
        Assign member(s) to an attribute.

        Parameters
        ----------
        attribute_name : str
            Name of the attribute.
        str_Value : str
            A string value
        member_list : list of int or int
            Member number ID or ID List.

        Returns
        -------
        int
            Returns 0 if successful else -1 if general error.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> beamIdList = staad_obj.Geometry.GetBeamList()
        >>> result = staad_obj.Property.AssignMemberAttribute("MEMBTYPE", "BRACE
        >>> result = staad_obj.Property.AssignMemberAttribute("MEMBTYPE", "BRACE
        """
        if isinstance(member_list, int):
            member_list = [member_list]
        safe_member_list = make_safe_array_long_input(member_list)
        vt_member_list = make_variant_vt_ref(safe_member_list,  automation.VT_AR
        return self._property.AssignMemberAttribute(attribute_name, str_Value, v
```

[docs]

```
    def DeleteMemberAttribute(self, attribute_name: str, str_Value: str):
        """
        Delete the member attribute by name.

        Parameters
        ----------
        attribute_name : str
            Name of the attribute.
        str_Value : str
            A string value
        Returns
        -------
        int
            Returns 0 if successful else -1 if general error.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> result = staad_obj.Property.DeleteMemberAttribute("MemberAttribute1'
        """
        return self._property.DeleteMemberAttribute(attribute_name, str_Value)
```

[docs]

```
    def GetMemberCountByAttribute(self, attribute_name: str, str_Value: str):
        """
        Return the number of member(s) in specified attribute.

        Parameters
        ----------
        attribute_name : str
            Name of the attribute.
        str_Value : str
            A string value

        Returns
        -------
        int
            Returns 0 if successful else -1 if general error.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> result = staad_obj.Property.GetMemberCountByAttribute("MEMBTYPE", "E
        """
        return self._property.GetMemberCountByAttribute(attribute_name, str_Valu
```

[docs]

```
    def GetMemberListByAttribute(self, attribute_name: str, str_Value: str):
        """
        Get member list by attribute.

        Parameters
        ----------
        attribute_name : str
            Name of the attribute.
        str_Value : str
            A string value
        Returns
        -------
        List of int
            Returns a list for Member(s) number ID list.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> result = staad_obj.Property.GetMemberListByAttribute("MEMBTYPE", "BR
        """
        memberListCount = self._property.GetMemberCountByAttribute(attribute_nar
        safe_memberList = make_safe_array_long(memberListCount)
        vt_varMemberList = make_variant_vt_ref(safe_memberList,  automation.VT_/
        result = self._property.GetMemberListByAttribute(attribute_name, str_Va
        if (result != 0):
            raise_os_error_if_error_code(-1)
        return vt_varMemberList[0]
```

[docs]

```
    def CreateElementAttribute(self, attribute_name: str, str_value: str):
        """
        Create element attribute by name.

        Parameters
        ----------
        attribute_name : str
            Name of the attribute.
        str_value : str
            A string value

        Returns
        -------
        int
            Returns 0 if successful else -1 if general error.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> result = staad_obj.Property.CreateElementAttribute("MEMBTYPE", "BRAC
        """
        return self._property.CreateElementAttribute(attribute_name, str_value)
```

[docs]

```
    def AssignElementAttribute(self, attribute_name: str, str_Value: str, elemer
        """
        Assign element(s) to an attribute.
        Parameters
        ----------
        attribute_name : str
            Name of the attribute.
        str_Value : str
            A string value
        element_list : list of int or int
            Element(s) number ID list.

        Returns
        -------
        int
            Returns 0 if successful else -1 if general error.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> result = staad_obj.Property.AssignElementAttribute("MEMBTYPE", "BRAC
        >>> result = staad_obj.Property.AssignElementAttribute("MEMBTYPE", "BRAC
        """
        if isinstance(element_list, int):
            element_list = [element_list]
        safe_element_list = make_safe_array_long_input(element_list)
        vt_element_list = make_variant_vt_ref(safe_element_list, automation.VT_A
        return self._property.AssignElementAttribute(attribute_name, str_Value, 
```

[docs]

```
    def DeleteElementAttribute(self, attribute_name: str, str_value: str):
        """
        Delete the element attribute by name.

        Parameters
        ----------
        attribute_name : str
            Name of the attribute.
        str_value : str
            A string value

        Returns
        -------
        int
            Returns 0 if successful else -1 if general error.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> result = staad_obj.Property.DeleteElementAttribute("MEMBTYPE", "BRAC
        """
        return self._property.DeleteElementAttribute(attribute_name, str_value)
```

[docs]

```
def GetElementCountByAttribute(self, attribute_name: str, str_value: str):
    """
    Returns the number of element(s) in specified attribute.

    Parameters
    ----------
    attribute_name : str
        Name of the attribute.
    str_value : str
        A string value

    Returns
    -------
    int
        Returns number of elements in specified attribute if successful else

    Examples
    --------
    >>> from openstaadpy import os_analytical
    >>> staad_obj = os_analytical.connect()
    >>> count = staad_obj.Property.GetElementCountByAttribute("MEMBTYPE", "E
    """
    return self._property.GetElementCountByAttribute(attribute_name, str_va
```

[docs]

```
def GetElementListByAttribute(self, attribute_name: str, str_value: str):
    """
    Get element list by attribute.

    Parameters
    ----------
    attribute_name : str
        Name of the attribute.
    str_value : str
        A string value

    Returns
    -------
    List of int
        Returns an elements number ID list.

    Examples
    --------
    >>> from openstaadpy import os_analytical
    >>> staad_obj = os_analytical.connect()
    >>> result = staad_obj.Property.GetElementListByAttribute("MEMBTYPE", "E
    """
    elementListCount = self._property.GetElementCountByAttribute(attribute_
    safe_elementList = make_safe_array_long(elementListCount)
    vt_elementList = make_variant_vt_ref(safe_elementList,  automation.VT_I4
    result = self._property.GetElementListByAttribute(attribute_name, str_va
    if (result != 0):
        raise_os_error_if_error_code(-1)
    return vt_elementList[0]
```

[docs]

```
    def GetAssignedAttributeCount(self, member_id: int):
        """
        Gets the number of attributes associated with beam or plate having the s

        Parameters
        ----------
        member_id : int
            The number ID of member or plate.

        Returns
        -------
        int
            Returns the number of attributes associated with beam (if member wit

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> plateIds = staad_obj.Geometry.GetPlateList()
        >>> result = staad_obj.Property.GetAssignedAttributeCount(plateIds[0])
        """
        return self._property.GetAssignedAttributeCount(member_id)
```

[docs]

```
    def GetAssignedAttributeByIndex(self, attribute_index: int):
        """
        Gets assigned attribute at specified index

        Parameters
        ----------
        attribute_index : int
            The attribute index.

        Returns
        -------
        tuple : tuple<string, string>L
            Returns a tuple consisting of attribute name and a string value, res

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> beam_count = staad_obj.Geometry.GetBeamList()
        >>> if beam_count > 0:
```

```
    >>>     beamIdList = staad_obj.Geometry.GetBeamList()
    >>>     attributeCount = staad_obj.Property.GetAssignedAttributeCount(be
    >>>     attribute_name, str_value  = staad_obj.Property.GetAssignedAttri
    """
    attribute_name = make_safe_str()
    string_val = make_safe_str()
    ref_attribute_name = make_variant_vt_ref(attribute_name, automation.VT_
    ref_string_val = make_variant_vt_ref(string_val, automation.VT_BSTR)
    self._property.GetAssignedAttributeByIndex(attribute_index, ref_attribu
    return ref_attribute_name[0], ref_string_val[0]
```

[docs]

```
def RemoveAttribute(self, attribute_name: str, str_value: str, member_ids: 
    """
    Remove the member(s) from specified attribute.

    Parameters
    ----------
    attribute_name : str
        Name of the attribute.
    str_value : str
        A string value
    member_ids : list of int or int
        Member(s) number ID list.

    Returns
    -------
    int
        Returns 0 if successful else -1 if general error.

    Examples
    --------
    >>> from openstaadpy import os_analytical
    >>> staad_obj = os_analytical.connect()
    >>> beamIds = staad_obj.Geometry.GetBeamList()
    >>> result = staad_obj.Property.RemoveAttribute("MEMBTYPE", "BRACE", bea
    """
    if isinstance(member_ids, int):
        member_ids = [member_ids]
    safe_varMemberList = make_safe_array_long_input(member_ids)
    vt_varMemberList = make_variant_vt_ref(safe_varMemberList,  automation.
    return self._property.RemoveAttribute(attribute_name, str_value, vt_var
```

[docs]

```
def GetMemberSpecCode(self, member_id: int):
    """
    Get the type of specification attached to member with specified member 

    Parameters
    ----------
```

```
        member_id : int
            The member number ID.

        Returns
        -------
        int
            Returns value referring to type of member specification as per table
                +-------+-----------------------------+
                | Value | Type of Member Specification |
                +=======+=============================+
                | 0     | Truss Member                |
                +-------+-----------------------------+
                | 1     | Tension-only Member         |
                +-------+-----------------------------+
                | 2     | Compression-only Member     |
                +-------+-----------------------------+
                | 3     | Cable-only Member           |
                +-------+-----------------------------+
                | 4     | Joist Member                |
                +-------+-----------------------------+
                | -1    | Other                       |
                +-------+-----------------------------+

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> beamIds = staad_obj.Geometry.GetBeamList()
        >>> result = staad_obj.Property.GetMemberSpecCode(beamIds[0])
        """
        safe_SpecCode = make_safe_array_long(1)
        vt_SpecCode = make_variant_vt_ref(safe_SpecCode,  automation.VT_I4)
        self._property.GetMemberSpecCode(member_id, vt_SpecCode)
        return vt_SpecCode[0]
```

[docs]

```
    def GetPublishedProfileName(self, staad_profile_name: str, country_code: in
        """
        Get project published name by STAAD profile name.

        Parameters
        ----------
        staad_profile_name : str
            STAAD profile name.
        country_code : int
            The value for the specified country.

        Returns
        -------
        int
            Returns published profile name if successful.\n
            Returns NULL or empty string if unable to find any equivalent publis
        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> result = staad_obj.Property.GetPublishedProfileName("STAADProfile1",
        """
        return self._property.GetPublishedProfileName(staad_profile_name, countr
```

[docs]

```
    def GetSTAADProfileName(self, published_name: str, country_code: int):
        """
        Gets STAAD profile name by published profile name.

        Parameters
        ----------
        published_name : str
            Published profile name.
        country_code : int
            The value for the specified country.

        Returns
        -------
        int
            Returns STAAD profile name if successful.\n
            Returns NULL if Unable to find any equivalent STAAD name correspondi

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> result = staad_obj.Property.GetSTAADProfileName("PublishedProfile1",
        """
        return self._property.GetSTAADProfileName(published_name, country_code)
```

[docs]

```
    def GetSectionPropertyValues(self, prof_type: int):
        """
        Retrieve long member properties of the specified beam member.

        Parameters
        ----------
        prof_type : int
            Assign Profile Type:
                +-------------------+-------+
                | Prof Type         | Value |
                +===================+=======+
                | AssignAngle       | 0     |
                +-------------------+-------+
                | AssignDoubleAngle | 1     |
                +-------------------+-------+
                | AssignBeam        | 2     |
                +-------------------+-------+
                | AssignColumn      | 3     |
                +-------------------+-------+
                | AssignChannel     | 4     |
                +-------------------+-------+


        Returns
        -------
        tuple : Tuple (float, float, float, float, float, float, float, float, float, 
            Returns a Tuple consisting of Width of the section (WID), Depth of 

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> section_width, section_depth, cross_section_area, shear_area_y, shea
        """
        safe_varfWidth = make_safe_array_double(1)
        vt_varfWidth = make_variant_vt_ref(safe_varfWidth,  automation.VT_R8)
        safe_varfDepth = make_safe_array_double(1)
        vt_varfDepth = make_variant_vt_ref(safe_varfDepth,  automation.VT_R8)
        safe_varfAx = make_safe_array_double(1)
        vt_varfAx = make_variant_vt_ref(safe_varfAx,  automation.VT_R8)
        safe_varfAy = make_safe_array_double(1)
        vt_varfAy = make_variant_vt_ref(safe_varfAy,  automation.VT_R8)
        safe_varfAz = make_safe_array_double(1)
        vt_varfAz = make_variant_vt_ref(safe_varfAz,  automation.VT_R8)
        safe_varfIx = make_safe_array_double(1)
        vt_varfIx = make_variant_vt_ref(safe_varfIx,  automation.VT_R8)
        safe_varfIy = make_safe_array_double(1)
        vt_varfIy = make_variant_vt_ref(safe_varfIy,  automation.VT_R8)
        safe_varfIz = make_safe_array_double(1)
        vt_varfIz = make_variant_vt_ref(safe_varfIz,  automation.VT_R8)
        safe_varfTf = make_safe_array_double(1)
        vt_varfTf = make_variant_vt_ref(safe_varfTf,  automation.VT_R8)
        safe_varfTw = make_safe_array_double(1)
        vt_varfTw = make_variant_vt_ref(safe_varfTw,  automation.VT_R8)
        result = self._property.GetSectionPropertyValues(prof_type, vt_varfWidt
        if result < 0:
            raise_os_error_if_error_code(result)
        return vt_varfWidth[0], vt_varfDepth[0], vt_varfAx[0], vt_varfAy[0], vt_
```

[docs]

```
def GetSectionPropertyValuesEx(self, section_property_id: int):
    """
    Returns the section property Values of the specified beam.

    Parameters
    ----------
    section_property_id : int
        Section property ID.
    Returns
    -------
    tuple : Tuple(int, list)
        Returns a Tuple consisting of Number referring to below table, a flo

    Examples
    --------
    >>> from openstaadpy import os_analytical
    >>> staad_obj = os_analytical.connect()
    >>> nAssignedSectionPropID = staad_obj.Property.CreateBeamPropertyFromTa
    >>> result = staad_obj.Property.GetSectionPropertyValuesEx(nAssignedSect
    """
    safe_propType = make_safe_array_long(1)
    vt_propType = make_variant_vt_ref(safe_propType,  automation.VT_I4)
    safe_propValues = make_safe_array_double(24)
    vt_propValues = make_variant_vt_ref(safe_propValues,  automation.VT_ARRA
    result = self._property.GetSectionPropertyValuesEx(section_property_id,
    if result < 1:
        raise_os_error_if_error_code(-1)
    return vt_propType[0], list(vt_propValues[0])
```

[docs]

```
    def DeleteMemberReleaseSpec(self, beam_id: int, release_location: int):
        """
        Delete MEMBER RELEASE specification.

        Parameters
        ----------
        beam_id : int
            The beam number ID.
        release_location : int
            The Release location at START (= 0) or END (= 1) of the member.

        Returns
        -------
        bool
            Returns True if Delete Member Release Specification Successful.
            Returns False if Delete Member Release Specification failed

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> beamIds = staad_obj.Geometry.GetBeamList()
        >>> result = staad_obj.Property.DeleteMemberReleaseSpec(beamIds[0], 1)
        """
        return self._property.DeleteMemberReleaseSpec(beam_id, release_location
```

[docs]

```
    def GetBeamSectionPropertyValuesEx(self, beam_id: int):
        """
        Returns the section property Values of the specified beam.

        Parameters
        ----------
        beam_id : int
            The beam number ID.

        Returns
        -------
        tuple : Tuple(int, list)
            Returns a Tuple consisting of property type (proptype) number referr
                +----------------------------+----------+------------------------
                | Section Type               | propType | propValues
                +============================+==========+========================
                | BEAM ST                    | 610      | Ax D Bf Tf Tw Iz Iy Ix
                +----------------------------+----------+------------------------
                | BEAM D                     | 616      | D Bf Tf Tw Iz Iy Ix SP
                +----------------------------+----------+------------------------
                | BEAM TC                    | 613      | Ax D Bf Tf Tw Iz Iy Ix
                +----------------------------+----------+------------------------
                | BEAM BC                    | 614      | Ax D Bf Tf Tw Iz Iy Ix
                +----------------------------+----------+------------------------
                | BEAM TB                    | 615      | Ax D Bf Tf Tw Iz Iy Ix
                +----------------------------+----------+------------------------
                | BEAM T                     | 611      | Ax D Bf Tf Tw Iz Iy Ix
                +----------------------------+----------+------------------------
                | BEAM CM                    | 612      | Ax D Bf Tf Tw Iz Iy Ix
                +----------------------------+----------+------------------------
                | CHANNEL ST                 | 630      | Ax D Bf Tf Tw Iz Iy Ix
                +----------------------------+----------+------------------------
                | CHANNEL D                  | 631      | Ax D Bf Tf Tw Iz Iy Ix
                +----------------------------+----------+------------------------
                | CHANNEL FR                 | 633      | Ax D Bf Tf Tw Iz Iy Ix
                +----------------------------+----------+------------------------
                | ANGLE ST                   | 640      | Ax D B T Iz Iy Ix
                +----------------------------+----------+------------------------
                | ANGLE LD                   | 642      | Ax D B T Iz Iy Ix LD
                +----------------------------+----------+------------------------
                | ANGLE SD                   | 643      | Ax D B T Iz Iy Ix SD
                +----------------------------+----------+------------------------
                | ANGLE RA                   | 641      | Ax D B T Iz Iy Ix
                +----------------------------+----------+------------------------
                | ANGLE SA                   | 646      | Ax D B T Iz Iy Ix
                +----------------------------+----------+------------------------
                | PIPE ST                    | 660      | Ax OD Tw Iz Iy Ix
                +----------------------------+----------+------------------------
                | HSS RECTANGLE              | 654      | Ax D B T Iz Iy Ix
                +----------------------------+----------+------------------------
                | HSS ROUND                  | 655      | Ax OD Tw Iz Iy Ix
                +----------------------------+----------+------------------------
                | CASTEL ST                  | 656      | Ax D Bf Tf Tw Iz Iy Ix
                +----------------------------+----------+------------------------
                | TUBE ST                    | 650      | Ax D B T Iz Iy Ix
                +----------------------------+----------+------------------------
```

```
| TEE ST                    | 620      | Ax D Bf Tf Tw Iz Iy Ix
+---------------------------+----------+------------------------
| PLATE STRIP               | 666      | Ax D B Iz Iy Ix
+---------------------------+----------+------------------------
| ANGLE COLD ST             | 644      | Ax D B T Iz Iy Ix R Ay
+---------------------------+----------+------------------------
| ANGLE COLD ST WITH LIPS   | 645      | Ax D B T Iz Iy Ix R LIP
+---------------------------+----------+------------------------
| CHANNEL COLD ST           | 634      | Ax D Bf T R Iz Iy Ix Ay
+---------------------------+----------+------------------------
| CHANNEL COLD ST WITH LIPS | 635      | Ax D Bf T R Iz Iy Ix LI
+---------------------------+----------+------------------------
| ZEES COLD ST              | 662      | Ax D B T R Iz Iy Ix Ay
+---------------------------+----------+------------------------
| ZEES COLD ST WITH LIPS    | 663      | Ax D B T LIP LIP_Angle
+---------------------------+----------+------------------------
| HAT COLD ST               | 664      | Ax D B T BOT_F R Iz Iy
+---------------------------+----------+------------------------
| TAPER                     | 680      | F1 F2 F3 F4 F5 F6 F7
+---------------------------+----------+------------------------
| TAPERED TUBE              | 675      | | Ax Iz Iy Ix D1 D2
|                           |          | | TH SECTION_TYPE SECTI
+---------------------------+----------+------------------------
| PRISMATIC CIRCLE          | 671      | Ax Iz Iy Ix YD
+---------------------------+----------+------------------------
| PRISMATIC RECT            | 672      | Ax Iz Iy Ix YD ZD
+---------------------------+----------+------------------------
| PRISMATIC TRAP            | 674      | Ax Iz Iy Ix YD ZD ZB
+---------------------------+----------+------------------------
| PRISMATIC TEE             | 673      | Ax Iz Iy Ix YD ZD YB ZB
+---------------------------+----------+------------------------
| PRISMATIC GENERAL         | 676      | Ax Ay Az Ix Iy Iz YD ZD
+---------------------------+----------+------------------------
| SOLID ROUND               | 668      | Ax OD Tw Iz Iy Ix Z
+---------------------------+----------+------------------------
| UPT PRISMATIC             | 699      | Ax Iz Iy Ix Ay Az YD ZD
+---------------------------+----------+------------------------
| UPT GENERAL               | 697      | Ax D Td B Tb Iz Iy Ix S
+---------------------------+----------+------------------------
| UPT WIDE FLANGE           | 690      | Ax D Tw Wf Tf Iz Iy Ix
+---------------------------+----------+------------------------
| UPT CHANNEL               | 691      | Ax D Tw Wf Tf Iz Iy Ix
+---------------------------+----------+------------------------
| UPT ANGLE                 | 692      | Ax D Wf Tf R Ay Az Iz I
+---------------------------+----------+------------------------
| UPT DOUBLE ANGLE          | 693      | Ax D Wf Tf SP Iz Iy Ix
+---------------------------+----------+------------------------
| UPT TEE                   | 694      | Ax D Wf Tf Tw Iz Iy Ix
+---------------------------+----------+------------------------
| UPT PIPE                  | 695      | Ax OD ID Ay Az Iz Iy Ix
+---------------------------+----------+------------------------
| UPT TUBE                  | 696      | Ax D Wf Tf Iz Iy Ix Ay
+---------------------------+----------+------------------------
| UPT ISECTION              | 698      | Dww Tww Dww1 Bff Tff Bf
+---------------------------+----------+------------------------
```

```
        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> beamIds = staad_obj.Geometry.GetBeamList()
        >>> section_property_type, section_properties = staad_obj.Property.GetBe
        """
        safe_varPropType = make_safe_array_long(1)
        vt_varPropType = make_variant_vt_ref(safe_varPropType,  automation.VT_I4
        safe_varProperties = make_safe_array_double(24)
        vt_varProperties = make_variant_vt_ref(safe_varProperties,  automation.V
        result = self._property.GetBeamSectionPropertyValuesEx(beam_id, vt_varPr
        if not result:
            raise_os_error_if_error_code(-1)
        return vt_varPropType[0], list(vt_varProperties[0])
```

[docs]

```
    def GetSectionPropertyAssignedBeamCount(self, prof_type: int):
        """
        Get section  assigned beam count.

        Parameters
        ----------
        prof_type : int
            Assign Profile Type:
                +---------------------+-------+
                |      Prof Type      | Value |
                +=====================+=======+
                | AssignAngle         |   0   |
                +---------------------+-------+
                | AssignDoubleAngle   |   1   |
                +---------------------+-------+
                | AssignBeam          |   2   |
                +---------------------+-------+
                | AssignColumn        |   3   |
                +---------------------+-------+
                | AssignChannel       |   4   |
                +---------------------+-------+

        Returns
        -------
        int
            Returns the section table number if successful.\n
            Returns -3001 if cannot find member beam number.\n
            Returns -6004 if section not found in profile database.\n
            Returns -6022 if no property is attached to the member/element.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> result = staad_obj.Property.GetSectionPropertyAssignedBeamCount(1)
        """
        return self._property.GetSectionPropertyAssignedBeamCount(prof_type)
```

[docs]

```
    def GetSectionPropertyAssignedBeamList(self, prof_type: int):
        """
        Get section assigned beam list.

        Parameters
        ----------
        prof_type : int
            Assign Profile Type:
                +---------------------+-------+
                |      Prof Type      | Value |
                +=====================+=======+
                | AssignAngle         |   0   |
                +---------------------+-------+
                | AssignDoubleAngle   |   1   |
                +---------------------+-------+
                | AssignBeam          |   2   |
                +---------------------+-------+
                | AssignColumn        |   3   |
                +---------------------+-------+
                | AssignChannel       |   4   |
                +---------------------+-------+

        Returns
        -------
        list of int
            Returns a list of beam ids.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> result = staad_obj.Property.GetSectionPropertyAssignedBeamList(2)
        """
        beamListCount = self._property.GetSectionPropertyAssignedBeamCount(prof_
        safe_beamList = make_safe_array_long(beamListCount)
        vt_nBeamList = make_variant_vt_ref(safe_beamList,  automation.VT_ARRAY
        result = self._property.GetSectionPropertyAssignedBeamList(prof_type, v
        if result < 0:
            raise_os_error_if_error_code(result)
        return vt_nBeamList[0]
```

[docs]

```
    def GetIsotropicMaterialAssignedBeamCount(self, material_name: int):
        """
        Get isotropic material assigned beam count.
        Parameters
        ----------
        material_name : int
            Identification title of the material.

        Returns
        -------
        int
            Returns count of isotropic material assigned beams if successful.\n
            Else returns -6023 if Material not found in material database.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> material_name = "Material123"
        >>> count = staad_obj.Property.GetIsotropicMaterialAssignedBeamCount(mat
        """
        return self._property.GetIsotropicMaterialAssignedBeamCount(material_nar
```

[docs]

```
    def GetIsotropicMaterialAssignedBeamList(self, material_name: str):
        """
        Get isotropic material assigned beam list.

        Parameters
        ----------
        material_name : str
            Identification title of the material.

        Returns
        -------
        list of int
            Returns a list of beam ids.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> material_name = "Material123"
        >>> result = staad_obj.Property.GetIsotropicMaterialAssignedBeamList(mat
        """
        beamListCount = self._property.GetIsotropicMaterialAssignedBeamCount(mat
        safe_beamList = make_safe_array_long(beamListCount)
        vt_nBeamList = make_variant_vt_ref(safe_beamList,  automation.VT_ARRAY
        result = self._property.GetIsotropicMaterialAssignedBeamList(material_na
        if result < 0:
            raise_os_error_if_error_code(result)
        return vt_nBeamList[0]
```

[docs]

```
def CreatePropertyFromUserTable(self, section_name: str, table_no: int):
    """
    Create a section Property from User Table.

    Parameters
    ----------
    section_name : str
        Section name
    table_no : int
        Table Id

    Returns
    -------
    int
        Returns section property reference number if successful. Zero if not

    Examples
    --------
    >>> from openstaadpy import os_analytical
    >>> staad_obj = os_analytical.connect()
    >>> result = staad_obj.Property.CreatePropertyFromUserTable("H600X300X12
    """
    return self._property.CreatePropertyFromUserTable(section_name, table_no
```

[docs]

```
def GetBeamSectionPropertyRefNo(self, beam_id: int):
    """
    Returns the section property reference number of the specified beam.

    Parameters
    ----------
    beam_id : int
        The beam number ID

    Returns
    -------
    int
        Returns Section property ref number assigned to the  specified beam

    Examples
    --------
    >>> from openstaadpy import os_analytical
    >>> staad_obj = os_analytical.connect()
    >>> beamIds = staad_obj.Geometry.GetBeamList()
    >>> result = staad_obj.Property.GetBeamSectionPropertyRefNo(beamIds[0])
    """
    return self._property.GetBeamSectionPropertyRefNo(beam_id)
```

[docs]

```
    def GetUserProvidedTableCount(self):
        """
        Get the number of UPT tables.

        Returns
        -------
        int
            Returns the number of UPT tables.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> result = staad_obj.Property.GetUserProvidedTableCount()
        """
        return self._property.GetUserProvidedTableCount()
```

[docs]

```
    def GetSectionPropertyList(self):
        """
        Gets the list of Section Property Reference IDs.

        Returns
        -------
        list of int
            Returns a List of Section Property reference IDs.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> result = staad_obj.Property.GetSectionPropertyList()
        """
        sectionPropertyCount = self._property.GetSectionPropertyCount()
        safe_sectionProperty = make_safe_array_long(sectionPropertyCount)
        vt_nPropList = make_variant_vt_ref(safe_sectionProperty, automation.VT_
        result = self._property.GetSectionPropertyList(vt_nPropList)
        if not result:
            raise_os_error_if_error_code(-1)
        return vt_nPropList[0]
```

[docs]

```
    def RemovePropertyFromBeam(self, beam_id: int):
        """
        Remove property from beam.

        Parameters
        ----------
        beam_id : int
            The beam number ID.

        Returns
        -------
        int
            Returns 0 if successful else -1 if general error.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> result = staad_obj.Property.RemovePropertyFromBeam(1)
        """
        return self._property.RemovePropertyFromBeam(beam_id)
```

[docs]

```
    def DeleteProperty(self, property_id: int):
        """
        Delete property based on the property ID passed.

        Parameters
        ----------
        property_id : int
            Property ID.

        Returns
        -------
        int
            Returns true if successful else returns false if general error.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> result = staad_obj.Property.DeleteProperty(2)
        """
        return self._property.DeleteProperty(property_id)
```

[docs]

```
    def GetUserProvidedTableList(self):
        """
        Get the UPT table ID list.

        Returns
        -------
        List of int
            Returns UPT table ID list.
        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> result = staad_obj.Property.GetUserProvidedTableList()
        """
        UserProvidedTableCount = self._property.GetUserProvidedTableCount()
        safe_UserProvidedTable = make_safe_array_long(UserProvidedTableCount)
        vt_nTableListn = make_variant_vt_ref(safe_UserProvidedTable, automation
        result = self._property.GetUserProvidedTableList(vt_nTableListn)
        if not result:
            raise_os_error_if_error_code(-1)
        return vt_nTableListn[0]
```

[docs]

```
    def GetUserProvidedTableSectionCount(self, table_id: int):
        """
        Get the number of sections defined in specified User Provided Table (UP

        Parameters
        ----------
        table_id : int
            The User Provided Table (UPT) number ID.

        Returns
        -------
        int
            Returns number of section in a given UPT.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> result = staad_obj.Property.GetUserProvidedTableSectionCount(10)
        """
        return self._property.GetUserProvidedTableSectionCount(table_id)
```

[docs]

```
    def GetUserProvidedTableSectionList(self, table_id: int):
        """
        Get the list of section names in specified User Provided Table (UPT).

        Parameters
        ----------
        table_id : int
            The User Provided Table (UPT) number ID.

        Returns
        -------
        list of strings
```

```
            Returns a list of strings consisting of indexes and corresponding se

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> result = staad_obj.Property.GetUserProvidedTableSectionList(10)
        """
        UserProvidedTableSectionCount = self._property.GetUserProvidedTableSecti
        safe_UserProvidedTableSection = make_safe_array_string(UserProvidedTable
        vt_sectionList = make_variant_vt_ref(safe_UserProvidedTableSection,  au
        result = self._property.GetUserProvidedTableSectionList(table_id, vt_se
        if result < 0:
            raise_os_error_if_error_code(-1)
        return list(vt_sectionList[0])
```

[docs]

```
    def GetUserProvidedTableSectionProperties(self, table_id: int, section_name
        """
        Get the section type and section properties of specified UPT section.

        Parameters
        ----------
        table_id : int
            The User Provided Table (UPT) number ID.
        section_name : str
            UPT section string name given to this section property.
        property_count : int
            The number of properties present in section of UPT table (default is

        Returns
        -------
        tuple
            Returns a tuple consisting of UPT Section Type from the below table
                +-----------------------------+----------------------+----------
                |      User Table Type        |  UPT Section Type    |  propValue
                +=============================+======================+==========
                |    USER TABLE PRISMATIC     |         502          |  Ax Iz Iy
                +-----------------------------+----------------------+----------
                |     USER TABLE GENERAL      |         482          |  Ax D Td E
                +-----------------------------+----------------------+----------
                |   USER TABLE WIDE FLANGE    |         412          |  Ax D Tw W
                +-----------------------------+----------------------+----------
                |     USER TABLE CHANNEL      |         422          |  Ax D Tw W
                +-----------------------------+----------------------+----------
                |      USER TABLE ANGLE       |         432          |  D Wf Tf F
                +-----------------------------+----------------------+----------
                |   USER TABLE DOUBLE ANGLE   |         442          |  D Wf Tf S
                +-----------------------------+----------------------+----------
                |       USER TABLE TEE        |         452          |  Ax D Wf T
                +-----------------------------+----------------------+----------
                |       USER TABLE PIPE       |         462          |  OD ID Ay
                +-----------------------------+----------------------+----------
                |       USER TABLE TUBE       |         472          |  Ax D Wf 
                +-----------------------------+----------------------+----------
                |     USER TABLE ISECTION     |         492          |  Dww Tww D
                +-----------------------------+----------------------+----------

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> section_type, section_properties = staad_obj.Property.GetUserProvide
        """
        safe_sectionType = make_safe_array_long(1)
        vt_sectionType = make_variant_vt_ref(safe_sectionType,  automation.VT_I
        safe_propertyVals = make_safe_array_double(property_count)
        vt_propertyVals = make_variant_vt_ref(safe_propertyVals,  automation.VT_
        result = self._property.GetUserProvidedTableSectionProperties(table_id,
        if not result:
            raise_os_error_if_error_code(-1)
        return vt_sectionType[0], list(vt_propertyVals[0])
```

[docs]

```
def GetPropertyUniqueID(self, property_unique_id: int):
    """
    Get Property Unique ID.

    Parameters
    ----------
    property_unique_id : int
        Property number

    Returns
    -------
    int
        Returns property Unique ID.

    Examples
    --------
    >>> from openstaadpy import os_analytical
    >>> staad_obj = os_analytical.connect()
    >>> property_unique_id = staad_obj.Property.GetPropertyUniqueID(3)
    """
    return self._property.GetPropertyUniqueID(property_unique_id)
```

[docs]

```
def SetPropertyUniqueID(self, property_number: int, property_unique_id: str
    """
    Set Property Unique ID to specification property number.

    Parameters
    ----------
```

```
        property_number : int
            Property number
        property_unique_id : str
            Property Unique ID

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> result = staad_obj.Property.SetPropertyUniqueID(4, "EA8A58A7-FF56-4F
        """
        return self._property.SetPropertyUniqueID(property_number, property_uni
```

[docs]

```
    def DeleteMemberSpec(self, spec_id: int):
        """
        Delete specification based on the specification number passed.

        Parameters
        ----------
        spec_id : int
            The specification number.

        Returns
        -------
        int
            Returns true delete specification successful, else false if delete s

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> beamIds = staad_obj.Geometry.GetBeamList()
        >>> result = staad_obj.Property.DeleteMemberSpec(beamIds[0])
        """
        return self._property.DeleteMemberSpec(spec_id)
```

[docs]

```
    def RemoveMemberReleaseSpecFromBeam(self, beam_id: int, release_location: in
        """
        Removes the member specification from a particular member at the provide

        Parameters
        ----------
        beam_id : int
            The beam number ID.
        release_location : int
            The Release location at START (= 0) or END (= 1) of the member.

        Returns
        -------
        int
            Returns true if successful else false if it fails.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> beamIds = staad_obj.Geometry.GetBeamList()
        >>> staad_obj.Property.RemoveMemberReleaseSpecFromBeam(beamIds[0], 0)
        """
        return self._property.RemoveMemberReleaseSpecFromBeam(beam_id, release_
```

[docs]

```
    def RemoveMemberOffsetSpecFromBeam(self, beam_id: int, release_location: in
        """
        Removes the member offset specification from a particular member at the

        Parameters
        ----------
        beam_id : int
            The beam number ID.
        release_location : int
            The Release location at START (= 0) or END (= 1) of the member.

        Returns
        -------
        bool
            Returns true if successful else false if it fails.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> beamIds = staad_obj.Geometry.GetBeamList()
        >>> result = staad_obj.Property.RemoveMemberOffsetSpecFromBeam(beamIds[
        """
        return self._property.RemoveMemberOffsetSpecFromBeam(beam_id, release_l
```

[docs]

```
    def RemoveMemberTrussSpecFromBeam(self, beam_id: int):
        """
        Remove member truss specification from beam.

        Parameters
        ----------
        beam_id : int
            The beam number ID.

        Returns
        -------
        int
            Returns 0 if OK else -1 if general error.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> beamIds = staad_obj.Geometry.GetBeamList()
        >>> result = staad_obj.Property.RemoveMemberTrussSpecFromBeam(beamIds[0
        """
        return self._property.RemoveMemberTrussSpecFromBeam(beam_id)
```

[docs]

```
    def RemoveMemberInactiveSpecFromBeam(self, beam_id: int):
        """
        Remove member inactive specification from beam.

        Parameters
        ----------
        beam_id : int
            The beam number ID.

        Returns
        -------
        int
            Returns 0 if OK else -1 if general error.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> beamIds = staad_obj.Geometry.GetBeamList()
        >>> staad_obj.Property.RemoveMemberInactiveSpecFromBeam(beamIds[0])
        """
        return self._property.RemoveMemberInactiveSpecFromBeam(beam_id)
```

[docs]

```
    def RemoveMemberTensionSpecFromBeam(self, beam_id: int):
        """
        Remove member tension specification from beam.

        Parameters
        ----------
        beam_id : int
            The beam number ID.

        Returns
        -------
        int
```

```
            Returns 0 if OK else -1 if general error.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> beamIds = staad_obj.Geometry.GetBeamList()
        >>> result = staad_obj.Property.RemoveMemberTensionSpecFromBeam(beamIds
        """
        return self._property.RemoveMemberTensionSpecFromBeam(beam_id)
```

[docs]

```
    def RemoveMemberIgnoreStiffSpecFromBeam (self, beam_id:int):
        """
        Remove member ignore stiff specification from beam.

        Parameters
        ----------
        BeamNo : int
            The beam number ID.

        Returns
        -------
        int
            Returns 0 if OK else returns -1 if general error

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> beamIds = staad_obj.Geometry.GetBeamList()
        >>> result = staad_obj.Property.RemoveMemberTensionSpecFromBeam(beamIds
        """
        return self._property.RemoveMemberIgnoreStiffSpecFromBeam(beam_id)
```

[docs]

```
    def CreateBeamPropertyFromTableEx(self, country_code: int, section_name: st
        """
        Creates beam property from table.

        Parameters
        ----------
        country_code : int
            The value for the specified country::
                +--------------+----------------------+
                | Country Code | Country              |
                +==============+======================+
                | 1            | American             |
                +--------------+----------------------+
                | 2            | Australian           |
                +--------------+----------------------+
                | 3            | British              |
                +--------------+----------------------+
                | 4            | Canadian             |
                +--------------+----------------------+
                | 5            | Chinese              |
                +--------------+----------------------+
                | 6            | Dutch                |
                +--------------+----------------------+
                | 7            | European             |
                +--------------+----------------------+
                | 8            | French               |
                +--------------+----------------------+
                | 9            | German               |
                +--------------+----------------------+
                | 10           | Indian               |
                +--------------+----------------------+
                | 11           | Japanese             |
                +--------------+----------------------+
                | 12           | Russian              |
                +--------------+----------------------+
                | 13           | SouthAfrican         |
                +--------------+----------------------+
                | 14           | Spanish              |
                +--------------+----------------------+
                | 15           | Venezuelan           |
                +--------------+----------------------+
                | 16           | Korean               |
                +--------------+----------------------+
        section_name : str
            Name of the section.
        solid_shape_type : int
            The specification type number:
                +----------------+----------------------+
                | Solid Shape ID | The shape of section |
                +================+======================+
                | 1              | Plate Strip          |
                +----------------+----------------------+
                | 2              | Solid Rect           |
                +----------------+----------------------+
                | 3              | Solid Round          |
                +----------------+----------------------+
                | 4              | Round                |
                +----------------+----------------------+
                | 5              | Cable                |
                +----------------+----------------------+

        Returns
        -------
        int
            Returns the assigned section property ID else returns 0 if library e

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
```

```
        >>> nCountryCode = 6
        >>> strSectionName = "HE100A"
        >>> typeSolidShape = 1
        >>> result = staad_obj.Property.CreateBeamPropertyFromTableEx(nCountryCo
        """
        return self._property.CreateBeamPropertyFromTableEx(country_code, sectio
```

[docs]

```
    def RemoveMemberCompressionSpecFromBeam(self, beam_id: int):
        """
        Remove member compression specification from beam.

        Parameters
        ----------
        beam_id : int
            The beam number ID.

        Returns
        -------
        int
            Returns 0 if OK else returns -1 if general error

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> beamIds = staad_obj.Geometry.GetBeamList()
        >>> result = staad_obj.Property.RemoveMemberCompressionSpecFromBeam(bean
        """
        return self._property.RemoveMemberCompressionSpecFromBeam(beam_id)
```

[docs]

```
    def RemoveMemberCableSpecFromBeam(self, beam_id: int, tension_or_length: in
        """
        Removes the member cable specification from a particular member at the p

        Parameters
        ----------
        beam_id : int
            The beam number ID.
        tension_or_length : int
            The Cable location at Tension (= 0) or Length (= 1) of the member.

        Returns
        -------
        bool
            Returns 'True' if remove member cable specification succeeded else

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> beamIds = staad_obj.Geometry.GetBeamList()
        >>> result = staad_obj.Property.RemoveMemberCableSpecFromBeam(beamIds[0]
        """
        return self._property.RemoveMemberCableSpecFromBeam(beam_id, tension_or
```

[docs]

```
    def RemoveElementPlaneStressSpecFromPlate(self, plate_id: int):
        """
        Remove element plane stress specification from plate.

        Parameters
        ----------
        plate_id : int
            The plate number ID.

        Returns
        -------
        int
            Returns 0 if OK else returns -1 if general error.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> plateIds = staad_obj.Geometry.GetPlateList()
        >>> result = staad_obj.Property.RemoveElementPlaneStressSpecFromPlate(p
        """
        return self._property.RemoveElementPlaneStressSpecFromPlate(plate_id)
```

[docs]

```
    def RemoveElementIgnoreInplaneRotnSpecFromPlate(self, plate_id: int):
        """
        Remove element ignore in plane rotation specification from plate.

        Parameters
        ----------
        plate_id : int
            The plate number ID.

        Returns
        -------
        int
            Returns 0 if OK else returns -1 if general error.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> plateIds = staad_obj.Geometry.GetPlateList()
        >>> result = staad_obj.Property.RemoveElementIgnoreInplaneRotnSpecFromPl
        """
        return self._property.RemoveElementIgnoreInplaneRotnSpecFromPlate(plate_
```

[docs]

```
    def RemoveElementNodeReleaseSpecFromPlate(self, plate_id: int, node_id: int
        """
        Remove element node release specification from plate.

        Parameters
        ----------
        plate_id : int
            The plate number ID.
        node_id : int
            The node number ID to be released.

        Returns
        -------
        int
            Returns 0 if OK else returns -1 if general error.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> plateIds = staad_obj.Geometry.GetPlateList()
        >>> nodeIds = staad_obj.Geometry.GetNodeList()
        >>> result = staad_obj.Property.RemoveElementNodeReleaseSpecFromPlate(pl
        """
        return self._property.RemoveElementNodeReleaseSpecFromPlate(plate_id, no
```

[docs]

```
    def GetUserProvidedTableNo(self, table_index: int):
        """
        Get section user provided table number ID by user table index.

        Parameters
        ----------
        table_index : int
            User Provided Table (UPT) index.

        Returns
        -------
        int
            Returns User Provided Table (UPT) number ID else returns -1 if gener

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> upt_ids = staad_obj.Property.GetUserProvidedTableList()
        >>> result = staad_obj.Property.GetUserProvidedTableNo(upt_ids[0])
        """
        return self._property.GetUserProvidedTableNo(table_index)
```

[docs]

```
    def GetUserProvidedTableSectionType(self, table_id: int):
        """
        Get the user provided table section type in specified User Provided Tab

        Parameters
        ----------
        table_id : int
            User Provided Table (UPT) number ID.

        Returns
        -------
        int
            Returns an int for number referring to Section Type Code table:
                +-------------------------+---------------------+
                | User Table Type         | UPT Section Type    |
                +=========================+=====================+
                | USER TABLE PRISMATIC    | 502                 |
                +-------------------------+---------------------+
                | USER TABLE GENERAL      | 482                 |
                +-------------------------+---------------------+
                | USER TABLE WIDE FLANGE  | 412                 |
                +-------------------------+---------------------+
                | USER TABLE CHANNEL      | 422                 |
                +-------------------------+---------------------+
                | USER TABLE ANGLE        | 432                 |
                +-------------------------+---------------------+
                | USER TABLE DOUBLE ANGLE | 442                 |
                +-------------------------+---------------------+
                | USER TABLE TEE          | 452                 |
                +-------------------------+---------------------+
                | USER TABLE PIPE         | 462                 |
                +-------------------------+---------------------+
                | USER TABLE TUBE         | 472                 |
                +-------------------------+---------------------+
                | USER TABLE ISECTION     | 492                 |
                +-------------------------+---------------------+

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> upt_ids = staad_obj.Property.GetUserProvidedTableList()
        >>> result = staad_obj.Property.GetUserProvidedTableSectionType(7)
        """
        safe_sectionType = make_safe_array_long(1)
        vt_sectionType = make_variant_vt_ref(safe_sectionType,  automation.VT_I4
        self._property.GetUserProvidedTableSectionType(table_id, vt_sectionType)
        return vt_sectionType[0]
```

[docs]

```
    def GetMemberReleaseSpecEx(self, beam_id: int, release_spec_position: int):
        """
        Get releases for the specified member at the specified end.

        Parameters
        ----------
        beam_id : int
            Beam number ID.
        release_spec_position : int
            Member Start end (= 0); member End end (= 1).

        Returns
        -------
        tuple
            Returns a tuple consisting of following items respectively :
                0. Translational release list with 6 elements for 6 DOFs. Elemen
                1. Rotational releases list with 6 elements for 6 DOFs.
                2. Element values Spring value or partial moment factor in float
                3. Rotational releases list with 3 elements for 3 rotational DOF

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> upt_ids = staad_obj.Property.GetUserProvidedTableList()
        >>> beamIds = staad_obj.Geometry.GetBeamList()
        >>> trans_release_list, rot_release_list, spring_const_list, rot_spring_
        """
        safe_Releaselist = make_safe_array_long(6)
        vt_Releaselist= make_variant_vt_ref(safe_Releaselist, automation.VT_ARRA
        safe_SpringConstlist = make_safe_array_double(6)
        vt_SpringConstlist= make_variant_vt_ref(safe_SpringConstlist, automation
        safe_MPFactor = make_safe_array_double(1)
        vt_MPFactor = make_variant_vt_ref(safe_MPFactor,  automation.VT_R8)
        safe_MPFactorlist = make_safe_array_double(3)
        vt_MPFactorlist= make_variant_vt_ref(safe_MPFactorlist, automation.VT_AR
        result = self._property.GetMemberReleaseSpecEx(beam_id, release_spec_pos
        if (result < 1):
            raise_os_error_if_error_code(-1)
        return vt_Releaselist[0], vt_SpringConstlist[0], vt_MPFactor[0], vt_MPFa
```

[docs]

```
    def GetThicknessPropertyCount(self):
        """
        Get Thickness Property Count.

        Returns
        -------
        int
            Returns total thickness properties count.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> result = staad_obj.Property.GetThicknessPropertyCount()
        """
        return self._property.GetThicknessPropertyCount()
```

[docs]

```
    def GetThicknessPropertyList(self):
        """
        Get Thickness Property ID list

        Returns
        -------
        List of int
            Returns a list of Thickness Property ID list.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> result = staad_obj.Property.GetThicknessPropertyList()
        """
        count = self._property.GetThicknessPropertyCount()
        safe_PropList = make_safe_array_long(count)
        vt_PropList= make_variant_vt_ref(safe_PropList, automation.VT_ARRAY | a
        result = self._property.GetThicknessPropertyList(vt_PropList)
        if not result :
            raise_os_error_if_error_code(-1)
        return vt_PropList[0]
```

[docs]

```
    def GetThicknessPropertyAssignedPlateCount(self, property_reference_id: int
        """
        Gets the count of plates which are assigned with the specified Thicknes

        Parameters
        ----------
        property_reference_id : int
            Thickness Property reference ID.

        Returns
        -------
        int
            Returns count of plates which are assigned with the specified Thickn

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> thickness_props = staad_obj.Property.GetThicknessPropertyList()
        >>> result = staad_obj.Property.GetThicknessPropertyAssignedPlateCount(t
        """
        return self._property.GetThicknessPropertyAssignedPlateCount(property_re
```

[docs]

```
    def GetThicknessPropertyAssignedPlateList(self, property_reference_id: int)
        """
        Gets the list of plate numbers which are assigned with the specified Thi

        Parameters
        ----------
        property_reference_id : int
            The specific Thickness Property reference ID.

        Returns
        -------
        List of int
            Returns a list for plate number list.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> thickness_props = staad_obj.Property.GetThicknessPropertyList()
        >>> result = staad_obj.Property.Property.GetThicknessPropertyAssignedPla
        """
        count = self._property.GetThicknessPropertyAssignedPlateCount(property_r
        safe_PlateList = make_safe_array_long(count)
        vt_PlateList= make_variant_vt_ref(safe_PlateList, automation.VT_ARRAY |
        result = self._property.GetThicknessPropertyAssignedPlateList(property_r
        if not result :
            raise_os_error_if_error_code(-1)
        return vt_PlateList[0]
```

[docs]

```
    def GetThicknessPropertyValues(self, property_reference_id: int):
        """
        Get Thickness Property Values

        Parameters
        ----------
        property_reference_id : int
            The specific Thickness Property reference ID

        Returns
        -------
        List of floats
            Returns a list for thickness value list.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> thickness_props = staad_obj.Property.GetThicknessPropertyList()
        >>> result = staad_obj.Property.GetThicknessPropertyValues(thickness_pro
        """
        safe_ThkList = make_safe_array_double(4)
        vt_ThkList= make_variant_vt_ref(safe_ThkList, automation.VT_ARRAY | auto
        result = self._property.GetThicknessPropertyValues(property_reference_id
        if (result < 4):
            raise_os_error_if_error_code(-1)
        return vt_ThkList[0]
```

[docs]

```
    def GetPlateSectionPropertyRefNo(self, PlateNo: int):
        """
        Get the assigned section property ID of specified plate.

        Parameters
        ----------
        PlateNo : int
            The plate number ID.

        Returns
        -------
        int
            Returns the assigned section property ID if successful, else returns
                - -4001 : Cannot find plate.
                - -6022 : No property is attached to the plate.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> plate_list = staad_obj.Geometry.GetPlateList()
        >>> result = staad_obj.Property.GetPlateSectionPropertyRefNo(plate_list[
        """
        return self._property.GetPlateSectionPropertyRefNo(PlateNo)
```

[docs]

```
    def RemovePropertyFromPlate(self, plate_id: int):
        """
        Removes Thickness Property From the specific surface.

        Parameters
        ----------
        plate_id : int
            Plate Id for plate to remove thickness from

        Returns
        -------
        int
            Returns value of the assigned section property ID.\n
            Else returns '-4001' if cannot find plate with specified plate id (p
            Else returns '-6022' if no property is attached to the plate.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> result = staad_obj.Property.RemovePropertyFromPlate(2)
        """
        return self._property.RemovePropertyFromPlate(plate_id)
```

[docs]

```
    def GetIsotropicMaterialAssignedPlateCount(self, material_name: int):
        """
        Gets the count of plates assigned with the specific isotropic material.

        Parameters
        ----------
        material_name : str
            Material Name.

        Returns
        -------
        int
            Returns count of plates assigned with the specific isotropic materia

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> result = staad_obj.Property.GetIsotropicMaterialAssignedPlateCount('
        """
        return self._property.GetIsotropicMaterialAssignedPlateCount(material_na
```

[docs]

```
    def GetIsotropicMaterialAssignedPlateList(self, material_name: str):
        """
        Gets the list of plate numbers which are assigned with the specified iso
        Parameters
        ----------
        material_name : str
            Material Name.

        Returns
        -------
        List of int
            Returns a list for plate id of plates which have material assigned t

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> result = staad_obj.Property.GetIsotropicMaterialAssignedPlateList("C
        """
        count = self._property.GetIsotropicMaterialAssignedPlateCount(material_n
        safe_PlateList = make_safe_array_long(count)
        vt_PlateList= make_variant_vt_ref(safe_PlateList, automation.VT_ARRAY |
        result = self._property.GetIsotropicMaterialAssignedPlateList(material_n
        if not result :
            raise_os_error_if_error_code(-1)
        return vt_PlateList[0]
```

[docs]

```
    def AssignMaterialToSolid(self, material_name: str, solid_ids: list):
        """
        Assign material to solid.

        Parameters
        ----------
        material_name : str
            Identification title of material.
        solid_ids : list
            List of integers containing solid numbers.

        Returns
        -------
        int
            Returns '1' if True else '0' if False.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> solid_ids = staad_obj.Geometry.GetSolidList()
        >>> result = staad_obj.Property.AssignMaterialToSolid("CONCRETE1", solid
        """
        safe_SolidNo = make_safe_array_long_input(solid_ids)
        vt_solid_ids = make_variant_vt_ref(safe_SolidNo, automation.VT_ARRAY | a
        return self._property.AssignMaterialToSolid(material_name, vt_solid_ids)
```

[docs]

```
def RemoveMaterialFromSolid(self, solid_id_list: list):
    """
    Remove Material From the specific Solids.

    Parameters
    ----------
    solid_id_list : list of int
        List of Solids IDs

    Returns
    -------
    int
        Returns 'True' if it succeeds in removing material from solids else

    Examples
    --------
    >>> from openstaadpy import os_analytical
    >>> staad_obj = os_analytical.connect()
    >>> result = staad_obj.Property.RemoveMaterialFromSolid([8, 5, 10, 3])
    """
    safe_SolidNoList = make_safe_array_long_input(solid_id_list)
    vt_solid_ids = make_variant_vt_ref(safe_SolidNoList, automation.VT_ARRAY
    return self._property.RemoveMaterialFromSolid(vt_solid_ids)
```

[docs]

```
def GetSolidMaterialName(self, solid_id: int):
    """
    Get the material name of the specified solid.

    Parameters
    ----------
    solid_id : int
        The Solid number ID.

    Returns
    -------
    str
        Returns material name of the specified solid.

    Examples
    --------
    >>> from openstaadpy import os_analytical
    >>> staad_obj = os_analytical.connect()
    >>> solid_ids = staad_obj.Geometry.GetSolidList()
    >>> result = staad_obj.Property.GetSolidMaterialName(solid_ids[0])
    """
    return self._property.GetSolidMaterialName(solid_id)
```

[docs]

```
def GetIsotropicMaterialAssignedSolidCount(self, material_name: str):
    """
    Get the count of solids assigned with the specified isotropic material.

    Parameters
    ----------
    material_name : str
        Identification title of the material.

    Returns
    -------
    int
        Returns count of solids assigned with the specified isotropic materi

    Examples
    --------
    >>> from openstaadpy import os_analytical
    >>> staad_obj = os_analytical.connect()
    >>> result = staad_obj.Property.GetIsotropicMaterialAssignedSolidCount('
    """
    return self._property.GetIsotropicMaterialAssignedSolidCount(material_na
```

[docs]

```
def GetIsotropicMaterialAssignedSolidList(self, material_name: str):
    """
    Get isotropic material assigned solid list.

    Parameters
    ----------
    material_name : str
        Identification title of the material.

    Returns
    -------
    List of int
        Returns a list of int for list of solid.

    Examples
    --------
    >>> from openstaadpy import os_analytical
    >>> staad_obj = os_analytical.connect()
    >>> result = staad_obj.Property.GetIsotropicMaterialAssignedSolidList("S
    """
    count = self._property.GetIsotropicMaterialAssignedSolidCount(material_
    safe_nSolidList = make_safe_array_long(count)
    vt_nSolidList= make_variant_vt_ref(safe_nSolidList, automation.VT_ARRAY
    result = self._property.GetIsotropicMaterialAssignedSolidList(material_
    if not result :
        raise_os_error_if_error_code(-1)
    return vt_nSolidList[0]
```

[docs]

```
def CreateIsotropicMaterialPropertiesEx(self, material_name: str, elasitici
    """
    Creates isotropic material property extended.

    Parameters
    ----------
    material_name : str
        Identification title of material of material.
    elasiticity : float
        Modulus of elasticity (E).
    poisson : float
        Poisson's ratio (POI).
    shear_modulus : float
        Shear modulus (G).
    density : float
        Weight density (DEN).
    alpha : float
        Coefficient of thermal expansion (ALP).
    damping_ratio : float
        Damping ratio (DAMP).
    fy : float
        Yield stress (Fy)
    fu : float
        Tensile strength (Fu).
    ry : float
        Yield strength ratio (Ry).
    rt : float
        Tensile strength ratio (Rt).
    fcu : float
        Compressive strength (Fcu).

    Returns
    -------
    int
        Status code:
            - 1 : Material is updated as a material with that name was alrea
            - 0 : Material is created.
            - -1 : General Error

    Examples
    --------
    >>> from openstaadpy import os_analytical
    >>> staad_obj = os_analytical.connect()
    >>> result = staad_obj.Property.CreateIsotropicMaterialPropertiesEx("STE
    """
    return self._property.CreateIsotropicMaterialPropertiesEx(material_name
```

[docs]

```
def GetIsotropicMaterialPropertiesEx(self, material_number: int):
    """
    Get the properties for the specified isotropic material number.

    Parameters
    ----------
    material_number : int
        Zero based index of the material

    Returns
    -------
    tuple : Tuple(str, float, float, float, float, float, float, float, floa
        Returns a Tuple consisting of Modulus of elasticity (E), Poisson's r

    Examples
    --------
    >>> from openstaadpy import os_analytical
    >>> staad_obj = os_analytical.connect()
    >>> material_name, elasticity, poisson_ratio, shear_modulus, density, co
    """
    safe_Elasiticity = make_safe_array_double(1)
    vt_Elasiticity = make_variant_vt_ref(safe_Elasiticity,  automation.VT_R8
    safe_Poisson = make_safe_array_double(1)
    vt_Poisson = make_variant_vt_ref(safe_Poisson,  automation.VT_R8)
    safe_ShearModulus = make_safe_array_double(1)
    vt_ShearModulus = make_variant_vt_ref(safe_ShearModulus,  automation.VT_
    safe_Density = make_safe_array_double(1)
    vt_Density = make_variant_vt_ref(safe_Density,  automation.VT_R8)
    safe_Alpha = make_safe_array_double(1)
    vt_Alpha = make_variant_vt_ref(safe_Alpha,  automation.VT_R8)
    safe_CrDamp = make_safe_array_double(1)
    vt_CrDamp = make_variant_vt_ref(safe_CrDamp,  automation.VT_R8)
    safe_Fy = make_safe_array_double(1)
    vt_Fy = make_variant_vt_ref(safe_Fy,  automation.VT_R8)
    safe_Fu = make_safe_array_double(1)
    vt_Fu = make_variant_vt_ref(safe_Fu,  automation.VT_R8)
    safe_Ry = make_safe_array_double(1)
    vt_Ry = make_variant_vt_ref(safe_Ry,  automation.VT_R8)
    safe_Rt = make_safe_array_double(1)
    vt_Rt = make_variant_vt_ref(safe_Rt,  automation.VT_R8)
    safe_Fcu = make_safe_array_double(1)
    vt_Fcu = make_variant_vt_ref(safe_Fcu,  automation.VT_R8)
    material_name = self._property.GetIsotropicMaterialPropertiesEx(material
    return material_name, vt_Elasiticity[0], vt_Poisson[0], vt_ShearModulus
```

[docs]

```
def GetMaterialPropertyEx(self, material_name: str):
    """
    Get the properties for the specified isotropic material Name.

    Parameters
```

```
        ----------
        material_name : str
            Material name

        Returns
        -------
        tuple : Tuple(float, float, float, float, float, float, float, float, fl
            Returns a tuple consisting of Modulus of elasticity (E), Poisson's r

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> elasticity, poisson_ratio, shear_modulus, density, coef_thermal_exp,
        """
        safe_Elasiticity = make_safe_array_double(1)
        vt_Elasiticity = make_variant_vt_ref(safe_Elasiticity,  automation.VT_R8
        safe_Poisson = make_safe_array_double(1)
        vt_Poisson = make_variant_vt_ref(safe_Poisson,  automation.VT_R8)
        safe_Density = make_safe_array_double(1)
        vt_Density = make_variant_vt_ref(safe_Density,  automation.VT_R8)
        safe_Alpha = make_safe_array_double(1)
        vt_Alpha = make_variant_vt_ref(safe_Alpha,  automation.VT_R8)
        safe_CrDamp = make_safe_array_double(1)
        vt_CrDamp = make_variant_vt_ref(safe_CrDamp,  automation.VT_R8)
        safe_Fy = make_safe_array_double(1)
        vt_Fy = make_variant_vt_ref(safe_Fy,  automation.VT_R8)
        safe_Fu = make_safe_array_double(1)
        vt_Fu = make_variant_vt_ref(safe_Fu,  automation.VT_R8)
        safe_Ry = make_safe_array_double(1)
        vt_Ry = make_variant_vt_ref(safe_Ry,  automation.VT_R8)
        safe_Rt = make_safe_array_double(1)
        vt_Rt = make_variant_vt_ref(safe_Rt,  automation.VT_R8)
        safe_Fcu = make_safe_array_double(1)
        vt_Fcu = make_variant_vt_ref(safe_Fcu,  automation.VT_R8)
        result = self._property.GetMaterialPropertyEx(material_name, vt_Elasiti
        if not result:
            raise_os_error_if_error_code(-1)
        return vt_Elasiticity[0], vt_Poisson[0], vt_Density[0], vt_Alpha[0], vt_
```

[docs]

```
    def CreateUPTTableEx(self, table_ref_id: int, table_type: int):
        """
        Create User Provided Table (UPT) specified by table number ID and Table

        Parameters
        ----------
        table_ref_id : int
            A new table number ID.
        table_type : int
            Type of the table:
                +------+----------------------------+
                |  No. |         Table Type         |
                +======+=============================+
                |  1   | scUserTableWideFlangeTitle  |
                +------+-----------------------------+
                |  2   | scUserTableChannelTitle     |
                +------+-----------------------------+
                |  3   | scUserTableAngleTitle       |
                +------+-----------------------------+
                |  4   | scUserTableDoubleAngleTitle |
                +------+-----------------------------+
                |  5   | scUserTableTeeTitle         |
                +------+-----------------------------+
                |  6   | scUserTablePipeTitle        |
                +------+-----------------------------+
                |  7   | scUserTableTubeTitle        |
                +------+-----------------------------+
                |  8   | scUserTableGeneralTitle     |
                +------+-----------------------------+
                |  9   | scUserTableIsectionTitle    |
                +------+-----------------------------+
                |  10  | scUserTablePrismaticTitle   |
                +------+-----------------------------+

        Returns
        -------
        int
            Returns table number ID if successful else '0' if create new User Pr

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> result = staad_obj.Property.CreateUPTTableEx(6, 7) // Create User Pr
        """
        return self._property.CreateUPTTableEx(table_ref_id, table_type)
```

[docs]

```
    def GetShapeCode(self, country_code: int, section_name: str):
        """
        Get the Shape Code with specific Country and specific Section Name.

        Parameters
        ----------
        country_code : int
            Country id according to the table below:
                +--------------+---------------------+
                | Country Code | Country             |
                +==============+=====================+
                | 1            | American            |
                +--------------+---------------------+
                | 2            | Australian          |
                +--------------+---------------------+
                | 3            | British             |
                +--------------+---------------------+
                | 4             | Canadian              |
                +---------------+-----------------------+
                | 5             | Chinese               |
                +---------------+-----------------------+
                | 6             | Dutch                 |
                +---------------+-----------------------+
                | 7             | European              |
                +---------------+-----------------------+
                | 8             | French                |
                +---------------+-----------------------+
                | 9             | German                |
                +---------------+-----------------------+
                | 10            | Indian                |
                +---------------+-----------------------+
                | 11            | Japanese              |
                +---------------+-----------------------+
                | 12            | Russian               |
                +---------------+-----------------------+
                | 13            | SouthAfrican          |
                +---------------+-----------------------+
                | 14            | Spanish               |
                +---------------+-----------------------+
                | 15            | Venezuelan            |
                +---------------+-----------------------+
                | 16            | Korean                |
                +---------------+-----------------------+

        section_name : str
            Section Name.

        Returns
        -------
        int
            Returns the Shape Code according to the table below else '-1' if it
                +------------------------------+--------------------------------
                | Country                      | Shape Code
                +==============================+================================
                | American                     | | 1 for "Wshape",
                |                              | | 2 for "MShape",
                |                              | | 3 for "SShape",
                |                              | | 4 for "HPShape",
                |                              | | 5 for "BShape",
                |                              | | 6 for "Channel",
                |                              | | 7 for "MCChannel",
                |                              | | 8 for "Angle",
                |                              | | 9 for "Tube",
                |                              | | 10 for "Pipe",
                |                              | | 11 for "HSSRectangle",
                |                              | | 12 for "HSSRound",
                |                              | | 13 for "CastellatedNonCompBea
                |                              | | 14 for "CastellatedCompBeam",
                |                              | | 15 for "RodShape",
                |                              | | 16 for "CableShape",
                |                              | | 23 for "HSSRectangleA1085",
                |                              | | 24 for "HSSRoundA1085"
                +------------------------------+--------------------------------
```

```
| Mexican                        | | 1 for "IEShape",
|                                | | 2 for "IRShape",
|                                | | 3 for "ISShape",
|                                | | 4 for "CEChannel",
|                                | | 5 for "LDAngle",
|                                | | 6 for "LIAngle",
|                                | | 7 for "OCPipe",
|                                | | 8 for "ORTube",
|                                | | 9 for "ORTubeR"
+--------------------------------+--------------------------------
| Australian                     | | 1 for "UBShape",
|                                | | 2 for "UCShape",
|                                | | 3 for "WBShape",
|                                | | 4 for "WCShape",
|                                | | 5 for "Channel",
|                                | | 6 for "Angle"
+--------------------------------+--------------------------------
| British                        | | 1 for "UBShape",
|                                | | 2 for "UCShape",
|                                | | 3 for "UPShape",
|                                | | 4 for "JOShape",
|                                | | 5 for "Channel",
|                                | | 6 for "Angle",
|                                | | 7 for "Tube",
|                                | | 8 for "Pipe"
+--------------------------------+--------------------------------
| Canadian                       | | 1 for "Wshape",
|                                | | 2 for "MShape",
|                                | | 3 for "SShape",
|                                | | 4 for "HPShape",
|                                | | 5 for "WWShape",
|                                | | 6 for "Channel",
|                                | | 7 for "MCChannel",
|                                | | 8 for "Angle",
|                                | | 9 for "Tube",
|                                | | 10 for "Pipe",
|                                | | 11 for "HSSRect",
|                                | | 12 for "HSSRound"
+--------------------------------+--------------------------------
| Chinese                        | | 1 for "IShape",
|                                | | 2 for "Channel",
|                                | | 3 for "Angle",
|                                | | 4 for "Tube",
|                                | | 5 for "Pipe",
|                                | | 6 for "TShape",
|                                | | 7 for "HShape"
+--------------------------------+--------------------------------
| Dutch                          | | 1 for "HEShape",
|                                | | 2 for "IPEShape",
|                                | | 3 for "IPNShape",
|                                | | 4 for "UPNChannel",
|                                | | 5 for "Angle",
|                                | | 6 for "Tube",
|                                | | 7 for "Pipe",
|                                | | 8 for "PlateStrip",
|                                | | 9 for "SolidRound",
|                              | | 10 for "SolidSquare"
+------------------------------+-------------------------------
| European                     | | 1 for "IPEShape",
|                              | | 2 for "HEShape",
|                              | | 3 for "DILShape",
|                              | | 4 for "IPNShape",
|                              | | 5 for "UChannel",
|                              | | 6 for "UPNChannel",
|                              | | 7 for "Angle",
|                              | | 8 for "Tube",
|                              | | 9 for "Pipe",
|                              | | 10 for "BulbFlat",
|                              | | 11 for "FlatBar",
|                              | | 12 for "HDShape",
|                              | | 13 for "HLShape",
|                              | | 14 for "HPShape",
|                              | | 15 for "SolidSquare",
|                              | | 16 for "UPEChannel",
|                              | | 17 for "UAPChannel",
|                              | | 18 for "Rhs",
|                              | | 19 for "Shs",
|                              | | 20 for "Chs"
+------------------------------+-------------------------------
| French                       | | 1 for "IPEShape",
|                              | | 2 for "HEShape",
|                              | | 3 for "IPNShape",
|                              | | 4 for "Channel",
|                              | | 5 for "Angle",
|                              | | 6 for "Tube",
|                              | | 7 for "Pipe"
+------------------------------+-------------------------------
| German                       | | 1 for "IPEShape",
|                              | | 2 for "HEShape",
|                              | | 3 for "IShape",
|                              | | 4 for "UChannel",
|                              | | 5 for "Angle",
|                              | | 6 for "Tube",
|                              | | 7 for "Pipe"
+------------------------------+-------------------------------
| Indian                       | | 1 for "SShape",
|                              | | 2 for "IShape",
|                              | | 3 for "MShape",
|                              | | 4 for "WShape",
|                              | | 5 for "TShape",
|                              | | 6 for "Channel",
|                              | | 7 for "Angle",
|                              | | 8 for "Tube",
|                              | | 9 for "Pipe",
|                              | | 10 for "WPBShape",
|                              | | 11 for "NPBShape"
+------------------------------+-------------------------------
| Brazilian                    | | 1 for "IShape",
|                              | | 2 for "WHShape",
|                              | | 3 for "WIShape",
|                              | | 4 for "TShape",
|                              | | 5 for "Channel",
|                             | | 6 for "Angle",
|                             | | 7 for "Rhs",
|                             | | 8 for "Shs",
|                             | | 9 for "Chs",
|                             | | 10 for "Pipe",
|                             | | 11 for "Cs",
|                             | | 12 for "Cvs",
|                             | | 13 for "Vs",
|                             | | 14 for "SShape"
+-----------------------------+---------------------------------
| Japanese                    | | 1 for "HShape",
|                             | | 2 for "IShape",
|                             | | 3 for "TShape",
|                             | | 4 for "Channel",
|                             | | 5 for "Angle",
|                             | | 6 for "Tube",
|                             | | 7 for "Pipe",
|                             | | 8 for "Rhs",
|                             | | 9 for "Shs",
|                             | | 10 for "Chs",
|                             | | 11 for "CTShape",
|                             | | 51 for "HShapeOld",
|                             | | 52 for "TShapeOld"
+-----------------------------+---------------------------------
| Russian                     | | 1 for "BShape",
|                             | | 2 for "SHShape",
|                             | | 3 for "KShape",
|                             | | 4 for "IShape",
|                             | | 5 for "Channel",
|                             | | 6 for "Angle",
|                             | | 7 for "Tube",
|                             | | 8 for "Pipe"
+-----------------------------+---------------------------------
| SouthAfrican                | | 1 for "IShape",
|                             | | 2 for "HShape",
|                             | | 3 for "PGShape",
|                             | | 4 for "CChannel",
|                             | | 5 for "Angle",
|                             | | 6 for "Tube",
|                             | | 7 for "Pipe"
+-----------------------------+---------------------------------
| Spanish                     | | 1 for "IPEShape",
|                             | | 2 for "HEShape",
|                             | | 3 for "IPNShape",
|                             | | 4 for "Channel",
|                             | | 5 for "Angle",
|                             | | 6 for "Tube",
|                             | | 7 for "Pipe"
+-----------------------------+---------------------------------
| Venezuelan                  | | 1 for "Beam",
|                             | | 2 for "Channel",
|                             | | 3 for "Angle",
|                             | | 4 for "Tube",
|                             | | 5 for "Pipe"
+-----------------------------+---------------------------------
| Korean                      | | 1 for "WShape",
|                             | | 2 for "HShape",
|                             | | 3 for "IShape",
|                             | | 4 for "WTShape",
|                             | | 5 for "Channel",
|                             | | 6 for "Angle",
|                             | | 7 for "Pipe",
|                             | | 8 for "Tube"
+-----------------------------+---------------------------------
| Aluminum                    | | 1 for "AAStandardIBeams",
|                             | | 2 for "HBeam",
|                             | | 3 for "ArmyNavyIBeam",
|                             | | 4 for "AmericanStandardIBeam",
|                             | | 5 for "IBeam",
|                             | | 6 for "AAStandardChannel",
|                             | | 7 for "Channel",
|                             | | 8 for "ArmyNavyChannel",
|                             | | 9 for "SpecialChannel",
|                             | | 10 for "AmericanStandardChanne
|                             | | 11 for "EqualLegAngle",
|                             | | 12 for "SquareEndEqualLegAngle
|                             | | 13 for "UnequalLegAngle",
|                             | | 14 for "SquareEndUnequalLegAng
|                             | | 15 for "SquareTube",
|                             | | 16 for "RectangularTube",
|                             | | 17 for "RoundTube"
+-----------------------------+---------------------------------
| UserTable                   | | 1 for "WideFlange",
|                             | | 2 for "Channel",
|                             | | 3 for "Angle",
|                             | | 4 for "DblAngle",
|                             | | 5 for "Tee",
|                             | | 6 for "Pipe",
|                             | | 7 for "Tube",
|                             | | 8 for "General",
|                             | | 9 for "ISection",
|                             | | 10 for "Prismatic"
+-----------------------------+---------------------------------
| AmericanColdFormed          | | 1 for "Angle",
|                             | | 2 for "AngleS",
|                             | | 3 for "Channel",
|                             | | 4 for "ChannelS",
|                             | | 5 for "Zee",
|                             | | 6 for "ZeeS",
|                             | | 7 for "Hat",
|                             | | 8 for "Pipe",
|                             | | 9 for "Tube"
+-----------------------------+---------------------------------
| RCecoColdFormed             | | 1 for "Angle",
| (Reserved)                  | | 2 for "AngleS",
|                             | | 3 for "Channel",
|                             | | 4 for "ChannelS",
|                             | | 5 for "Zee",
|                             | | 6 for "ZeeS",
|                             | | 7 for "Hat",
|                             | | 8 for "Pipe",
|                             | | 9 for "Tube",
|                             | | 10 for "EaveStrut"(Reserved)
+-----------------------------+---------------------------------
| Lysaght                     | | 4 for "ChannelS",
|                             | | 6 for "ZeeS"
+-----------------------------+---------------------------------
| IndianColdFormed            | | 1 for "Angle",
|                             | | 2 for "AngleS",
|                             | | 3 for "Channel",
|                             | | 4 for "ChannelS",
|                             | | 5 for "Zee",
|                             | | 6 for "ZeeS",
|                             | | 7 for "Hat"
+-----------------------------+---------------------------------
| BritishColdFormed           | | 1 for "Angle",
|                             | | 2 for "AngleS",
|                             | | 3 for "Channel",
|                             | | 4 for "ChannelS",
|                             | | 5 for "Zee",
|                             | | 6 for "ZeeS",
|                             | | 7 for "Hat",
|                             | | 8 for "Pipe",
|                             | | 9 for "Tube"
+-----------------------------+---------------------------------
| AustralianColdFormed        | | 1 for "RHS",
|                             | | 2 for "SHS",
|                             | | 3 for "CHS"
+-----------------------------+---------------------------------
| EuropeanColdFormed          | | 1 for "RHS",
|                             | | 2 for "SHS",
|                             | | 3 for "CHS"
+-----------------------------+---------------------------------
| KingspanColdFormed          | | 1 for "Angle",
|                             | | 2 for "AngleS",
|                             | | 3 for "Channel",
|                             | | 4 for "ChannelS",
|                             | | 5 for "Zee",
|                             | | 6 for "ZeeS",
|                             | | 7 for "Hat",
|                             | | 8 for "Pipe",
|                             | | 9 for "Tube"
+-----------------------------+---------------------------------
| JapaneseColdFormed          | | 11 for "BCP",
|                             | | 12 for "BCPT",
|                             | | 13 for "BCR"
+-----------------------------+---------------------------------
| RusColdFormed               | | 8 for "Pipe"
+-----------------------------+---------------------------------
| AITC-Timber                 | | 1 for "GluedLaminatedTimber",
|                             | | 2 for "Aspen",
|                             | | 3 for "BalsamFir",
|                             | | 4 for "BeechBirchHickory",
|                             | | 5 for "CoastSitkaSpruce",
|                             | | 6 for "Cottonwood",
|                             | | 7 for "DouglasFirLarch",
|                             | | 8 for "DouglasFirLarchNorth",
|                             | | 9 for "DouglasFirLarchSouth",
|                             | | 10 for "EasternHemlock",
|                             | | 11 for "EasternHemlockTamarack
|                             | | 12 for "EasternHemlockTamarack
|                             | | 13 for "EasternSoftwoods",
|                             | | 14 for "EasternSpruce",
|                             | | 15 for "EasternWhitePine",
|                             | | 16 for "HemFir",
|                             | | 17 for "HemFirNorth",
|                             | | 18 for "MixedMaple",
|                             | | 19 for "MixedOak",
|                             | | 20 for "MixedSouthernPine",
|                             | | 21 for "MountainHemlock",
|                             | | 22 for "NorthernPine",
|                             | | 23 for "NorthernRedOak",
|                             | | 24 for "NorthernSpecies",
|                             | | 25 for "NorthernWhiteCedar",
|                             | | 26 for "PonderosaPine",
|                             | | 27 for "RedMaple",
|                             | | 28 for "RedOak",
|                             | | 29 for "RedPine",
|                             | | 30 for "Redwood",
|                             | | 31 for "SitkaSpruce",
|                             | | 32 for "SouthernPine",
|                             | | 33 for "SprucePineFir",
|                             | | 34 for "SprucePineFirSouth",
|                             | | 35 for "WesternCedars",
|                             | | 36 for "WesternCedarsNorth",
|                             | | 37 for "WesternHemlock",
|                             | | 38 for "WesternHemlockNorth",
|                             | | 39 for "WesternWhitePine",
|                             | | 40 for "WesternWoods",
|                             | | 41 for "WhiteOak",
|                             | | 42 for "YellowPoplar"
+-----------------------------+-------------------------------
| American Steel Joist        | | 1 for "Kjoist",
|                             | | 2 for "KCSJoist",
|                             | | 3 for "LHJoist",
|                             | | 4 for "DLHJoist",
|                             | | 5 for "JoistGirder"
+-----------------------------+-------------------------------
| Generic                     | | 1 for "WShape",
|                             | | 2 for "TShape",
|                             | | 3 for "Channel",
|                             | | 4 for "Angle",
|                             | | 5 for "Tube",
|                             | | 6 for "Pipe",
|                             | | 7 for "Rectangle",
|                             | | 8 for "Round",
|                             | | 9 for "Zee",
|                             | | 20 for "General"
+-----------------------------+-------------------------------
| Canadian Timber             | | 1 for "GluedLaminatedTimber",
|                             | | 2 for "DouglasFirLarch",
|                             | | 3 for "HemFir",
|                             | | 4 for "NorthernSpecies",
|                             | | 5 for "SprucePineFir"
```

```
                +------------------------------+--------------------------------
                | Butler                       | | 4 for "EaveStrut",
                |                              | | 6 for "ZeePurlin",
                |                              | | 9 for "BoxStrut",
                |                              | | 10 for "WideFlange",
                |                              | | 11 for "TaperedWideFlange",
                |                              | | 12 for "SolidRound"
                +------------------------------+--------------------------------
                | Jindal                       | | 1 for "UBShape",
                |                              | | 2 for "HEShape",
                |                              | | 3 for "IPEShape",
                |                              | | 4 for "UCShape",
                |                              | | 5 for "ISMCShape",
                |                              | | 6 for "WPBShape",
                |                              | | 7 for "NPBShape"
                +------------------------------+--------------------------------
                | Tata Structura               | | 1 for "Rhs",
                |                              | | 2 for "Shs",
                |                              | | 3 for "Chs"
                +------------------------------+--------------------------------
                | APL Apollo Tubes             | | 1 for "Rhs",
                |                              | | 2 for "Shs",
                |                              | | 3 for "Chs"
                +------------------------------+--------------------------------

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> result = staad_obj.Property.GetShapeCode(5, "HW200X200")
        """
        return self._property.GetShapeCode(country_code, section_name)
```

[docs]

```
    def GetRecordForSection(self, country_code: int, section_name: str):
        """
        Get the Record No (Record No in Section database) on table with specifi

        Parameters
        ----------
        country_code : int
            Country id. (Refer OsProperty.CreateBeamPropertyFromTable for Countr
        section_name : str
            Section Name(Type: String).

        Returns
        -------
        int
            Returns record number for specific section if successful else -1 if

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> result = staad_obj.Property.GetRecordForSection(2, "HW200X200")
        """
        return self._property.GetRecordForSection(country_code, section_name)
```

[docs]

```
    def GetMemberAttributeCount(self):
        """
        Get the Count of Member Attribute.

        Returns
        -------
        int
            Returns Member Attribute Count

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> result = staad_obj.Property.GetMemberAttributeCount()
        """
        return self._property.GetMemberAttributeCount()
```

[docs]

```
    def GetMemberAttributeList(self):
        """
        Get member attribute list.

        Returns
        -------
        tuple : Tuple(list, list, int)
            Returns a tuple consisting of attribute name list, the correspondin

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> attribute_name_list, attribute_value_list, attribute_count = stead_
        """
        count = self._property.GetMemberAttributeCount()
        safe_attributeNameList = make_safe_array_string(count)
        vt_attributeNameList= make_variant_vt_ref(safe_attributeNameList, autom
        safe_attributeValueList = make_safe_array_string(count)
        vt_attributeValueList= make_variant_vt_ref(safe_attributeValueList, aut
        count = self._property.GetMemberAttributeList(vt_attributeNameList, vt_
        return vt_attributeNameList[0], vt_attributeValueList[0], count
```

[docs]

```
    def GetUserProvidedTableSectionPropertyCount(self, upt_table_id: int, secti
        """
        Get the user provided table section property count in specified User Pro

        Parameters
        ----------
        upt_table_id : int
            The User Provided Table (UPT) number ID.
        section_name : str
             UPT section string name given to this section property.

        Returns
        -------
        int
            Returns the number of section(s) in given UPT.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> result = staad_obj.Property.GetUserProvidedTableSectionPropertyCount
        """
        return self._property.GetUserProvidedTableSectionPropertyCount(upt_table
```

[docs]

```
    def CreateBeamPropertyFromTableComposite(self, country_code: int, section_na
        """
        Creates beam property from table composite.

        Parameters
        ----------
        country_code : int
            The value for the specified country
        section_name : str
            Name of the section.
        spec_type : int
            The specification type number:
                +-------+----------------------+
                | Index |      Spec Type       |
                +=======+======================+
                |  -1   | Define               |
                +-------+----------------------+
                |   0   | ST                   |
                +-------+----------------------+
                |   1   | RA                   |
                +-------+----------------------+
                |   2   | D                    |
                +-------+----------------------+
                |   3   | LD                   |
                +-------+----------------------+
                |   4   | SD                   |
                +-------+---------------------+
                |   5   | T (for aluminum)    |
                +-------+---------------------+
                |   6   | CM                  |
                +-------+---------------------+
                |   7   | TC                  |
                +-------+---------------------+
                |   8   | BC                  |
                +-------+---------------------+
                |   9   | TB                  |
                +-------+---------------------+
                |  10   | BA (for aluminum)   |
                +-------+---------------------+
                |  11   | FR                  |
                +-------+---------------------+
                |  12   | SA (for aluminum)   |
                +-------+---------------------+
        additional_spec_list : List
            List of additional specification values:
                +-------------+----------------------------------------------+
                | Spec Value  | Specification Description                    |
                +============+==============================================+
                | WP TH       | for TC and BC                                |
                +-------------+----------------------------------------------+
                | WP TH BW BT | for TB / WP TH for TB                        |
                +-------------+----------------------------------------------+
                | CT FC       | for CM                                       |
                +-------------+----------------------------------------------+
                | SP          | for D, BA and FR                             |
                +-------------+----------------------------------------------+
                | SP          | for LD and SD                                |
                +-------------+----------------------------------------------+
                | TH WT DT    | for Tube define                              |
                +-------------+----------------------------------------------+
                | OD ID       | for Pipe define                              |
                +-------------+----------------------------------------------+

        Returns
        -------
        int
            Returns the assigned section property ID if successful else returns
                - 0 : Library Error: Unable to create property.
                - -6004 : Section is not found in profile database.
                - -6005 : Section data for a section is not found.
                - -6006 : Invalid section type.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> result = staad_obj.Property.reateBeamPropertyFromTableComposite(10,
        """
        safe_AddSpeclist = make_safe_array_double_input(additional_spec_list)
        vt_AddSpeclist = make_variant_vt_ref(safe_AddSpeclist, automation.VT_ARI
        return self._property.CreateBeamPropertyFromTableComposite(country_code
```

[docs]

```
def CreateBeamPropertyFromTableWithCoverPlates(self, country_code: int, sec
    """
    Creates beam property from table with cover plates.

    Parameters
    ----------
    country_code : int
        The value for the specified country
    section_name : str
        Name of the section.
    spec_type : int
        The specification type number:
            +-------+----------------------+
            | Index |      Spec Type       |
            +=======+======================+
            |  -1   | Define               |
            +-------+----------------------+
            |   0   | ST                   |
            +-------+----------------------+
            |   1   | RA                   |
            +-------+----------------------+
            |   2   | D                    |
            +-------+----------------------+
            |   3   | LD                   |
            +-------+----------------------+
            |   4   | SD                   |
            +-------+----------------------+
            |   5   | T (for aluminum)     |
            +-------+----------------------+
            |   6   | CM                   |
            +-------+----------------------+
            |   7   | TC                   |
            +-------+----------------------+
            |   8   | BC                   |
            +-------+----------------------+
            |   9   | TB                   |
            +-------+----------------------+
            |  10   | BA (for aluminum)    |
            +-------+----------------------+
            |  11   | FR                   |
            +-------+----------------------+
            |  12   | SA (for aluminum)    |
            +-------+----------------------+
    additional_spec_list : List
        list of additional specification values:
            +------------+--------------------------------------------------+
            | Spec Value | Specification Description                        |
            +============+==================================================+
            | WP TH      | for TC and BC                                    |
            +------------+--------------------------------------------------+
            | WP TH BW BT | for TB / WP TH for TB                           |
            +-------------+---------------------------------------------+
            | CT FC       | for CM                                      |
            +-------------+---------------------------------------------+
            | SP          | for D, BA and FR                            |
            +-------------+---------------------------------------------+
            | SP          | for LD and SD                               |
            +-------------+---------------------------------------------+
            | TH WT DT    | for Tube define                             |
            +-------------+---------------------------------------------+
            | OD ID       | for Pipe define                             |
            +-------------+---------------------------------------------+

    Returns
    -------
    int
        Returns the assigned section property ID if successful else returns
            - 0 : Library Error: Unable to create property.
            - -6004 : Section is not found in profile database.
            - -6005 : Section data for a section is not found.
            - -6006 : Invalid section type.

    Examples
    --------
    >>> from openstaadpy import os_analytical
    >>> staad_obj = os_analytical.connect()
    >>> result = staad_obj.Property.CreateBeamPropertyFromTableWithCoverPlat
    """
    safe_AddSpeclist = make_safe_array_double_input(additional_spec_list)
    vt_AddSpeclist = make_variant_vt_ref(safe_AddSpeclist, automation.VT_ARI
    return self._property.CreateBeamPropertyFromTableWithCoverPlates(country
```

[docs]

```
    def AddUPTPropertyWIDEFLANGEUNEQUAL(self, table_reference_id: int, section_
        """
        Add unequal wide flange to a defined UPT section.

        Parameters
        ----------
        table_reference_id : int
            The existing table number ID.
        section_name : str
            UPT section string name.
        profile_spec_list : List
            Profile specification list which consists of the following correspo
                +-------+-----------------+
                | Index | Prop Spec Value |
                +=======+=================+
                |   0   | Ax              |
                +-------+-----------------+
                |   1   | D               |
                +-------+-----------------+
                |   2   | TW              |
                +-------+-----------------+
                |   3   | WF               |
                +-------+------------------+
                |   4   | TF               |
                +-------+------------------+
                |   5   | IZ               |
                +-------+------------------+
                |   6   | IY               |
                +-------+------------------+
                |   7   | IX               |
                +-------+------------------+
                |   8   | AY               |
                +-------+------------------+
                |   9   | AZ               |
                +-------+------------------+
                |  10   | WF1              |
                +-------+------------------+
                |  11   | TF1              |
                +-------+------------------+

        Returns
        -------
        bool
            Returns 'True' if add unequal wide flange successful.\n
            Returns 'False' if it encounters generate error.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> upt_num_id = staad_obj.Property.CreateUPTTable(1)
        >>> result = staad_obj.Property.AddUPTPropertyWIDEFLANGEUNEQUAL(upt_num_
        """
        safe_PropSpeclist = make_safe_array_double_input(profile_spec_list)
        vt_PropSpeclist = make_variant_vt_ref(safe_PropSpeclist, automation.VT_
        return self._property.AddUPTPropertyWIDEFLANGEUNEQUAL(table_reference_i
```

[docs]

```
    def AddUPTPropertyWIDEFLANGECOMPOSITE(self, table_reference_id: int, sectio
        """
        Add wide flange type with additional composite and bottom steel plate to

        Parameters
        ----------
        table_reference_id : int
            The existing table number ID.
        section_name : str
            UPT section string name.
        profile_spec_list : List
            Float list consisting of Profile Specifications data of size 12 (wit
                +-------+-----------------------------------------------------------
                | Index | Data
                +=======+===========================================================
                |   0   | Cross section area (AX).
                +-------+-----------------------------------------------------------
                |   1   | Depth of the section (D).
                +-------+-----------------------------------------------------------
                |   2   | Thickness of web (TW).
                +-------+-----------------------------------------------------------
                |   3   | Width of the top flange (WF).
                +-------+-----------------------------------------------------------
                |   4   | Thickness of top flange (TF).
                +-------+-----------------------------------------------------------
                |   5   | Torsional constant (IZ).
                +-------+-----------------------------------------------------------
                |   6   | Moment of inertia about local y-axis (IY).
                +-------+-----------------------------------------------------------
                |   7   | Moment of inertia about local z-axis (IX).
                +-------+-----------------------------------------------------------
                |   8   | Shear area in local y-axis. If zero, shear deformatio
                +-------+-----------------------------------------------------------
                |   9   | Shear area in local z-axis. If zero, shear deformatio
                +-------+-----------------------------------------------------------
                |  10   | Width of the bottom flange (WF1).
                +-------+-----------------------------------------------------------
                |  11   | Thickness of bottom flange (TF1).
                +-------+-----------------------------------------------------------
                |  12   | (for additional composite flange) Width of the composi
                +-------+-----------------------------------------------------------
                |  13   | (for additional composite flange) Width of the composi
                +-------+-----------------------------------------------------------
                |  14   | (for additional composite flange) Thickness of the co
                +-------+-----------------------------------------------------------
                |  15   | (for additional composite flange) Modular ratio of the
                +-------+-----------------------------------------------------------
                |  16   | (for additional bottom plate) Width of the additional
                +-------+-----------------------------------------------------------
                |  17   | (for additional bottom plate) Width of the additional
                +-------+-----------------------------------------------------------
                |  18   | (for additional bottom plate) Thickness of the additio
                +-------+-----------------------------------------------------------

        Returns
        -------
        bool
            Returns 'True' if OK, else 'False' if Error.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> upt_num_id = staad_obj.Property.CreateUPTTable(1)
        >>> result = staad_obj.Property.AddUPTPropertyWIDEFLANGECOMPOSITE(upt_nu
        """
        safe_varPropSpeclist = make_safe_array_double_input(profile_spec_list)
        vt_PropSpeclist = make_variant_vt_ref(safe_varPropSpeclist, automation.V
        return self._property.AddUPTPropertyWIDEFLANGECOMPOSITE(table_reference_
```

[docs]

```
def CreateTeePropertyFromTable(self, country_code: int, section_name: str, s
    """
    Creates Tee property from database.

    Parameters
    ----------
    country_code : int
        The value for the specified country.
    section_name : str
        Name of the section.
    spec_type : int
        The specification type number:
            +-------+----------------------+
            | Index | Spec Type            |
            +=======+======================+
            |  -1   | Define               |
            +-------+----------------------+
            |   0   | ST                   |
            +-------+----------------------+
            |   5   | T From Wide Flange   |
            +-------+----------------------+

    Returns
    -------
    int
        Returns the assigned section property ID if successful else returns
            - 0 : Library error: unable to create property.
            - -6004 : Section is not found in profile database.
            - -6005 : Section data for a section is not found.
            - -6006 : Invalid section type.

    Examples
    --------
    >>> from openstaadpy import os_analytical
    >>> staad_obj = os_analytical.connect()
    >>> result = staad_obj.Property.CreateTeePropertyFromTable(7, "ISNT20",
    """
    return self._property.CreateTeePropertyFromTable(country_code, section_
```

[docs]

```
def SetTypeToIsotropicMaterial(self, material_name: str, material_type: int
    """
    Set Type To the specified Isotropic Material.

    Parameters
    ----------
    material_name : str
        Identification title of the material.
    material_type : int
        Material Type.
    Returns
    -------
    bool
        Returns 'True' if Set Type to Material successful else 'False' if ge

    Examples
    --------
    >>> from openstaadpy import os_analytical
    >>> staad_obj = os_analytical.connect()
    >>> result = staad_obj.Property.SetTypeToIsotropicMaterial("TestMat", 4)
    """
    return self._property.SetTypeToIsotropicMaterial(material_name, materia
```

[docs]

```
    def GetTypeForIsotropicMaterial(self, material_name: str):
        """
        Get Type For the specified Isotropic Material.

        Parameters
        ----------
        material_name : str
            Identification title of the material.

        Returns
        -------
        int
            Returns an int for Material Type:
                +-------+-------------------------------+
                |  No.  | Material Type                 |
                +=======+===============================+
                |   0   | Not Specified                 |
                +-------+-------------------------------+
                |   1   | Steel                         |
                +-------+-------------------------------+
                |   2   | Concrete                      |
                +-------+-------------------------------+
                |   3   | Aluminum                      |
                +-------+-------------------------------+
                |   4   | Timber                        |
                +-------+-------------------------------+

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> result = staad_obj.Property.GetTypeForIsotropicMaterial(strInput)
        """
        safe_MatType = make_safe_array_long(1)
        vt_MatType = make_variant_vt_ref(safe_MatType,  automation.VT_I4)
        result = self._property.GetTypeForIsotropicMaterial(material_name, vt_Ma
        if not result:
            raise_os_error_if_error_code(-1)
        return vt_MatType[0]
```

[docs]

```
def CreatePropertyFromUPTTable(self, table_id: int, section_name: str):
    """
    Creates a section property from User Provided Table (UPT).

    Parameters
    ----------
    table_id : int
        The existing table number ID.
    section_name : str
        UPT section string name.

    Returns
    -------
    int
        Returns section property number ID if successful else -1 if general

    Examples
    --------
    >>> from openstaadpy import os_analytical
    >>> staad_obj = os_analytical.connect()
    >>> result = staad_obj.Property.CreatePropertyFromUPTTable(2, "UPT VJG5
    """
    return self._property.CreatePropertyFromUPTTable(table_id, section_name
```

[docs]

```
def CreateParametricSurfaceThicknessProperty(self, node_thickness_list: lis
    """
    Creates plate uniform or nonuniform thickness property.

    Parameters
    ----------
    node_thickness_list : List
        List consisting of thickness for all nodes.

    Returns
    -------
    int
        Returns the assigned section property ID if successful,\n
        Else returns status code from below:
            - -106 : node_thickness_list gives dimensional array error.
            - -6003 : Library error being unable to create property.

    Examples
    --------
    >>> from openstaadpy import os_analytical
    >>> staad_obj = os_analytical.connect()
    >>> result = staad_obj.Property.CreateParametricSurfaceThicknessProperty
    """
```

```
        safe_Thickness = make_safe_array_double_input(node_thickness_list)
        vt_Thickness = make_variant_vt_ref(safe_Thickness, automation.VT_ARRAY
        return self._property.CreateParametricSurfaceThicknessProperty(vt_Thick
```

[docs]

```
    def GetUptGeneralProfilePointsCount(self, table_reference_id: int, section_
        """
        Get profile points count from user provided general section table (UPT)

        Parameters
        ----------
        table_reference_id : int
            The existing table number ID.
        section_name : str
            UPT section string name.

        Returns
        -------
        tuple
            Returns a Tuple consisting of count of outer profile points and cou

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> result = staad_obj.Property.GetUptGeneralProfilePointsCount(1, "AAA
        """
        safe_CountOfOuter = make_safe_array_long(1)
        vt_CountOfOuter = make_variant_vt_ref(safe_CountOfOuter,  automation.VT
        safe_CountOfInner = make_safe_array_long(1)
        vt_CountOfInner = make_variant_vt_ref(safe_CountOfInner,  automation.VT
        result = self._property.GetUptGeneralProfilePointsCount(table_reference
        if not result:
            raise_os_error_if_error_code(-1)
        return vt_CountOfOuter[0], vt_CountOfInner[0]
```

[docs]

```
    def GetUptGeneralProfileBoundaryPoints(self, table_number_id: int, section_
        """
        Get Profile Points coordinate from User Provided general section Table 

        Parameters
        ----------
        table_number_id : int
            The existing table number ID.
        section_name : str
            UPT section string name.
        is_inner : bool
            (Reserved for inner points, set it to false)
        Returns
        -------
        Tuple of float
            Returns a tuple consisting of profile points coordinate list in Z ar

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> point_cord_z_list, point_cord_y_list = staad_obj.Property.GetUptGene
        """
        safe_CountOfOuter = make_safe_array_long(1)
        vt_CountOfOuter = make_variant_vt_ref(safe_CountOfOuter,  automation.VT_
        safe_CountOfInner = make_safe_array_long(1)
        vt_CountOfInner = make_variant_vt_ref(safe_CountOfInner,  automation.VT_
        count = self._property.GetUptGeneralProfilePointsCount(table_number_id,
        safe_varZP = make_safe_array_double(int(vt_CountOfOuter[0]))
        vt_varZP= make_variant_vt_ref(safe_varZP, automation.VT_ARRAY | automat
        safe_varYP = make_safe_array_double(int(vt_CountOfOuter[0]))
        vt_varYP= make_variant_vt_ref(safe_varYP, automation.VT_ARRAY | automat
        count = self._property.GetUptGeneralProfileBoundaryPoints(table_number_
        if (count == 0):
            raise_os_error_if_error_code(-1)
        return vt_varZP[0], vt_varYP[0]
```

[docs]

```
    def GetUptGeneralStressLocationPoints(self, table_reference_id: int, sectio
        """
        Stress Location in local coordinate from User Provided general section T

        Parameters
        ----------
        TableRef : int
            The existing table number ID.
        SectionName : str
            UPT section string name.

        Returns
        -------
        Tuple of list: Tuple(list, list)
            Returns a tuple consisting of list (of size 4) consisting of stress

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> point_cord_z_list, point_cord_y_list = staad_obj.Property.GetUptGene
        """
        safe_ZP = make_safe_array_double(4)
        vt_ZP= make_variant_vt_ref(safe_ZP, automation.VT_ARRAY | automation.VT_
        safe_YP = make_safe_array_double(4)
        vt_YP= make_variant_vt_ref(safe_YP, automation.VT_ARRAY | automation.VT_
        count = self._property.GetUptGeneralStressLocationPoints(table_reference
        if (count == 0):
            raise_os_error_if_error_code(-1)
        return (vt_ZP[0], vt_YP[0])
```

[docs]

```
    def GetInactiveMemberCount(self):
        """
        Returns the total number of inactive members in the current model.

        Returns
        -------
        int
            Returns the total number of inactive members.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> count = staad_obj.Property.GetInactiveMemberCount()
        """
        return self._property.GetInactiveMemberCount()
```

[docs]

```
    def GetInactiveMemberList(self):
        """
        Populates a list of the member ids of all the inactive members in the cu

        Returns
        -------
        List of int
            Returns a list for list of member number ids of inactive members.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> inactive_member_list = staad_obj.Property.GetInactiveMemberList()
        """
        count = self._property.GetInactiveMemberCount()
        safe_InactiveMemList = make_safe_array_long(count)
        vt_InactiveMemList= make_variant_vt_ref(safe_InactiveMemList, automatio
        self._property.GetInactiveMemberList(vt_InactiveMemList)
        return list(vt_InactiveMemList[0])
```

[docs]

```
    def GetAlphaAngleForSection(self, spec_property_id: int):
        """
        Gets the angle between the principal axis and geometric axis of the sect

        Parameters
        ----------
        spec_property_id : int
            The specified property ID.

        Returns
        -------
        float
            Returns a float for alpha angle (in Radian).

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> count = staad_obj.Property.GetAlphaAngleForSection(7)
        """
        safe_dAlpha = make_safe_array_double(1)
        vt_dAlpha = make_variant_vt_ref(safe_dAlpha,  automation.VT_R8)
        result = self._property.GetAlphaAngleForSection(spec_property_id, vt_dA
        if not result:
            raise_os_error_if_error_code(-1)
        return float(vt_dAlpha[0])
```

[docs]

```
    def GetCentroidLocationForSection(self, property_id: int):
        """
        Gets the location of the Centroid of the specified section.

        Parameters
        ----------
        property_id : int
            The specified property ID.

        Returns
        -------
        Tuple : tuple(int, int)
            Returns a tuple consisting of offset value of centroid along Y axis

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> section_list = staad_obj.Property.GetSectionPropertyList()
        >>> for section_id in section_list:
        >>>     y_axis_offset, z_axis_offset = staad_obj.Property.GetCentroidLo
        """
        safe_Cey = make_safe_array_double(1)
        vt_Cey = make_variant_vt_ref(safe_Cey,  automation.VT_R8)
        safe_Cez = make_safe_array_double(1)
        vt_Cez = make_variant_vt_ref(safe_Cez,  automation.VT_R8)
        result = self._property.GetCentroidLocationForSection(property_id, vt_C
        if not result:
            raise_os_error_if_error_code(-1)
        return vt_Cey[0], vt_Cez[0]
```

[docs]

```
    def DeleteAllControlDependentRelations(self):
        """
        Deletes all control/dependent joint specifications from model.

        Returns
        -------
        int
            Returns '0' if OK successful deleted else '1' if ERROR delete unsucc

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> status = staad_obj.Property.DeleteAllControlDependentRelations()
        """
        return self._property.DeleteAllControlDependentRelations()
```

[docs]

```
    def CreateWideFlangePropertyFromTable(self, country_code: int, section_name
        """
        Creates wide flange member property from table with data for all specs.

        Parameters
        ----------
        country_code : int
            The country CODE:
                +--------------+----------------------+
                | Country Code | Country              |
                +==============+======================+
                | 1            | American             |
                +--------------+----------------------+
                | 2            | Australian           |
                +--------------+----------------------+
                | 3            | British              |
                +--------------+----------------------+
                | 4            | Canadian             |
                +--------------+----------------------+
                | 5            | Chinese              |
                +--------------+----------------------+
                | 6            | Dutch                |
                +--------------+----------------------+
                | 7            | European             |
                +--------------+----------------------+
                | 8            | French               |
                +--------------+----------------------+
                | 9            | German                |
                +--------------+-----------------------+
                | 10           | Indian                |
                +--------------+-----------------------+
                | 11           | Japanese              |
                +--------------+-----------------------+
                | 12           | Russian               |
                +--------------+-----------------------+
                | 13           | SouthAfrican          |
                +--------------+-----------------------+
                | 14           | Spanish               |
                +--------------+-----------------------+
                | 15           | Venezuelan            |
                +--------------+-----------------------+
                | 16           | Korean                |
                +--------------+-----------------------+
        section_name : str
            Name of the section.
        spec_type : str
            The specification type number:
                +---------------+---------------+
                | Specification | Specification |
                | Type          | Number        |
                +===============+===============+
                | ST            | 0             |
                +---------------+---------------+
                | D             | 2             |
                +---------------+---------------+
                | T             | 5             |
                +---------------+---------------+
                | CM            | 6             |
                +---------------+---------------+
                | TC            | 7             |
                +---------------+---------------+
                | BC            | 8             |
                +---------------+---------------+
                | TB            | 9             |
                +---------------+---------------+
        specs_list : list
            The specification values corresponding to type shown in the table be
                +-------+---------------+---------------------------------------
                | Array |     Spec      |
                | Index |     Type      |                                   Desc
                +=======+===============+=======================================
                |   0   |    SP/CT/WP   | - SP: Spacing for double-I, double-C,
                |       |               | - CT: Conc. thickness for composite-I
                |       |               | - WP: Width of top cover plate for TC,
                +-------+---------------+---------------------------------------
                |   1   |    FC/TH      | - FC: Concrete grade for composite-I
                |       |               | - TH: Thickness of top cover plate for
                +-------+---------------+---------------------------------------
                |   2   |    CW/BW      | - CW: Concrete width for composite-I
                |       |               | - BW: Width of bottom cover plate for
                +-------+---------------+---------------------------------------
                |   3   |    CD/BT      | - CD: Concrete density for composite-I
                |       |               | - Thickness of bottom cover plate for
```

```
                +-------+--------------+----------------------------------------

        Returns
        -------
        int
            Returns the assigned section property ID if successful else -1 if ge

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> status = staad_obj.Property.CreateWideFlangePropertyFromTable(7, "HP
        """
        if (specs_list is None) or (len(specs_list) == 0):
            specs_list = [0]
        safe_SpecsList = make_safe_array_double_input(specs_list)
        vt_SpecsList = make_variant_vt_ref(safe_SpecsList, automation.VT_ARRAY
        return self._property.CreateWideFlangePropertyFromTable(country_code, se
```

[docs]

```
    def CreateIsotropicMaterialSteel(self, name: str, elasticity_mod: float, po
        """
        Creates isotropic material steel.

        Parameters
        ----------
        name : str
            Identification title of material.
        elasticity_mod : float
            Modulus of elasticity (E).
        poisson_ratio : float
            Poisson's ratio (POI).
        shear_modulus : float
            Shear modulus (G).
        density : float
            Weight density (DEN).
        thermal_expansion : float
            Coefficient of thermal expansion (ALP).
        damping_ratio : float
            Damping ratio (DAMP).
        tensile_strength : float
            Tensile strength (Fu).
        yield_strength : float
            Yield stress (Fy).
        tensile_ratio : float
            Tensile strength ratio (Rt).
        yield_ratio : float
            Yield strength ratio (Ry).
        is_physical : int
            Identifies if the material is for physical member (flag/int).

        Returns
        -------
        int
            Status code from below:
                - 1 : Material is updated as a material with that name was alrea
                - 0 : Material is created.
                - -1 : General Error

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> status = staad_obj.Property.CreateIsotropicMaterialSteel("STEEL1", 1
        """
        return self._property.CreateIsotropicMaterialSteel(name, elasticity_mod
```

[docs]

```
    def CreateIsotropicMaterialConcrete(self, name: str, elasticity: float, poi
        """
        Create isotropic concrete material.

        Parameters
        ----------
        name : str
            Material name.
        elasticity : float
            Modulus of elasticity (E).
        poisson : float
            Poisson's ratio.
        shear_modulus : float
            Shear modulus (G).
        density : float
            Weight density.
        alpha : float
            Coefficient of thermal expansion.
        damping_ratio : float
            Damping ratio.
        compressive_strength : float
            Compressive strength (Fcu).
        physical : int
            Flag indicating physical-member material (nonzero = physical).

        Returns
        -------
        int
            Status code:
                - 1 : Material updated as a material with that name was already
                - 0 : Material created
                - -1 : General error

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> result = staad_obj.Property.CreateIsotropicMaterialConcrete("CONCRET
        """
        return self._property.CreateIsotropicMaterialConcrete(name, elasticity,
```

[docs]

```
    def CreateIsotropicMaterialAluminum(self, material_name: str, elasticity_mod
        """
        Creates isotropic aluminum material.

        Parameters
        ----------
        material_name : str
            Material name.
        elasticity_mod : float
            Modulus of elasticity (E).
        poisson : float
            Poisson's ratio.
        shear_mod : float
            Shear modulus (G).
        density : float
            Weight density.
        thermal_exp : float
            Coefficient of thermal expansion.
        damping_ratio : float
            Damping ratio.
        physical_flag : int
            Flag indicating physical-member material (nonzero = physical).

        Returns
        -------
        int
            Status code:
                - 1 : Material updated (name existed).
                - 0 : Material created.
                - -1 : General error.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> result = staad_obj.Property.CreateIsotropicMaterialAluminum("Aluminu
        """
        return self._property.CreateIsotropicMaterialAluminum(material_name, ela
```

[docs]

```
    def CreateIsotropicMaterialTimber(self, material_name: str, elasticity: floa
        """
        Creates isotropic timber material.

        Parameters
        ----------
        material_name : str
            Identification title of the material.
        elasticity : float
            Modulus of elasticity (E).
        poisson : float
            Poisson's ratio (POI).
        shear_modulus : float
            Shear modulus (G).
        density : float
            Weight density (DEN).
        thermal_expansion : float
            Coefficient of thermal expansion (ALP).
        damping_ratio : float
            Damping ratio (DAMP).
        physical_flag : int
            Flag indicating if the material is for physical members (nonzero = p

        Returns
        -------
        int
            Status code:
                - 1 : Material updated (a material with that name already existe
                - 0 : Material created.
                - -1 : General error.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> output = staad_obj.Property.CreateIsotropicMaterialTimber("TIMBER1",
        """
        return self._property.CreateIsotropicMaterialTimber(material_name, elas
```

[docs]

```
    def RemoveAllElementNodeReleaseSpec(self):
        """
        Remove all element node release specification from the model.

        Returns
        -------
        int
            Returns '1' if OK else '0' if no element release specification prese

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> output = staad_obj.Property.RemoveAllElementNodeReleaseSpec()
        """
        return self._property.RemoveAllElementNodeReleaseSpec()
```

[docs]

```
    def CreateElementOffsetSpec(self, offset_direction: int, plate_node_index: 
        """
        Create ELEMENT OFFSET specification.

        Parameters
        ----------
        offset_direction : int
            The offset direction at Local (= 0) or Global (= 1) of the element.
        plate_node_index : int
            The Node index at which the offset is to be applied for local and gl
        x_offset : float
            The offset x coordinate.
        y_offset : float
            The offset y coordinate.
        z_offset : float
            The offset z coordinate.

        Returns
        -------
        int
            Returns the assigned specification number ID if successful else stat

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> output = staad_obj.Property.CreateElementOffsetSpec(7, 3, 9.9, 4.6, 
        """
        return self._property.CreateElementOffsetSpec(offset_direction, plate_no
```

[docs]

```
    def CreateElementLocalZOffsetSpec(self, node1_localz_offset: float, node2_lo
        """
        Create ELEMENT OFFSET specification (Z-Offset).

        Parameters
        ----------
        node1_localz_offset : float
            The offset at Node 1 for local-Z offset.
        node2_localz_offset : float
            The offset at Node 2 for local-Z offset.
        node3_localz_offset : float
            The offset at Node 3 for local-Z offset.
        node4_localz_offset : float
            The offset at Node 4 for local-Z offset.

        Returns
        -------
        int
            Returns the assigned specification number ID if successful else -60
        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> output = staad_obj.Property.CreateElementLocalZOffsetSpec(3.6, 17.0,
        """
        return self._property.CreateElementLocalZOffsetSpec(node1_localz_offset,
```

[docs]

```
    def GetElementLocalOffset(self, plate_id: int, plate_node_index: int):
        """
        Get element offsets in all three local directions.

        Parameters
        ----------
        plate_id : int
            The plate number ID.
        plate_node_index : int
            The Node Index at which the offset is to be applied (1/2/3/4).

        Returns
        -------
        tuple : Tuple(float, float, float)
            Returns a tuple consisting of the offset x coordinate, the offset y

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> plate_ids = staad_obj.Geometry.GetPlateIDs()
        >>> offset_x, offset_y, offset_z = staad_obj.Property.GetElementLocalOff
        """
        safe_OffsetX = make_safe_array_double(1)
        vt_OffsetX = make_variant_vt_ref(safe_OffsetX,  automation.VT_R8)
        safe_OffsetY = make_safe_array_double(1)
        vt_OffsetY = make_variant_vt_ref(safe_OffsetY,  automation.VT_R8)
        safe_OffsetZ = make_safe_array_double(1)
        vt_OffsetZ = make_variant_vt_ref(safe_OffsetZ,  automation.VT_R8)
        result = self._property.GetElementLocalOffset(plate_id, plate_node_inde
        if result < 0:
            raise_os_error_if_error_code(result)
        return vt_OffsetX[0], vt_OffsetY[0], vt_OffsetZ[0]
```

[docs]

```
    def GetElementGlobalOffSet(self, plate_id: int, plate_node_index: int):
        """
        Get element offsets in all three local directions.

        Parameters
        ----------
        plate_id : int
            The plate number ID.
        plate_node_index : int
            The Node Index at which the offset is to be applied (1/2/3/4).

        Returns
        -------
        tuple : Tuple(float, float, float)
            Returns a tuple consisting of the offset x coordinate (global), the

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> plate_ids = staad_obj.Geometry.GetPlateIDs()
        >>> offset_x, offset_y, offset_z = staad_obj.Property.GetElementGlobalOf
        """
        safe_OffsetX = make_safe_array_double(1)
        vt_OffsetX = make_variant_vt_ref(safe_OffsetX,  automation.VT_R8)
        safe_OffsetY = make_safe_array_double(1)
        vt_OffsetY = make_variant_vt_ref(safe_OffsetY,  automation.VT_R8)
        safe_OffsetZ = make_safe_array_double(1)
        vt_OffsetZ = make_variant_vt_ref(safe_OffsetZ,  automation.VT_R8)
        result = self._property.GetElementGlobalOffSet(plate_id, plate_node_inde
        if result < 0:
            raise_os_error_if_error_code(result)
        return vt_OffsetX[0], vt_OffsetY[0], vt_OffsetZ[0]
```

[docs]

```
    def GetElementOffSetSpec(self, plate_id: int, plate_node_index: int):
        """
        Get Element offsets in all three global directions.

        Parameters
        ----------
        plate_id : int
            The plate number ID.
        plate_node_index : int
            The Node Index at which the offset is to be applied (1/2/3/4).

        Returns
        -------
        tuple : Tuple(int, float, float, float)
            Returns a list consisting of the offset direction at Local (= 0) or

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> plate_ids = staad_obj.Geometry.GetPlateIDs()
        >>> direction, offset_x, offset_y, offset_z = staad_obj.Property.GetEler
        """
        safe_Direction = make_safe_array_long(1)
        vt_Direction = make_variant_vt_ref(safe_Direction,  automation.VT_I4)
        safe_OffsetX = make_safe_array_double(1)
        vt_OffsetX = make_variant_vt_ref(safe_OffsetX,  automation.VT_R8)
        safe_OffsetY = make_safe_array_double(1)
        vt_OffsetY = make_variant_vt_ref(safe_OffsetY,  automation.VT_R8)
        safe_OffsetZ = make_safe_array_double(1)
        vt_OffsetZ = make_variant_vt_ref(safe_OffsetZ,  automation.VT_R8)
        result = self._property.GetElementOffSetSpec(plate_id, plate_node_index
        if result < 0:
            raise_os_error_if_error_code(result)
        return (vt_Direction[0], vt_OffsetX[0], vt_OffsetY[0], vt_OffsetZ[0])
```

[docs]

```
    def GetCountofSectionPropertyValuesEx(self):
        """
        Returns the total count of Section Property values.

        Returns
        -------
        int
            Returns the total count of Section Property values.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> count = staad_obj.Property.GetCountofSectionPropertyValuesEx()
        """
        return self._property.GetCountofSectionPropertyValuesEx()
```

[docs]

```
    def CreateMemberCableSpecEx(self, tension_or_unstressed_len: int, spec_valu
        """
        Create MEMBER CABLE specification.

        Parameters
        ----------
        tension_or_unstressed_len : int
            Specify additional information about the cable:
                - 0 = Initial TENSION of Value in the cable to be considered.
                - 1 = Unstressed LENGTH of Value to be considered.
        spec_value : float
            Value for TENSION or Unstressed LENGTH
        tension_end_node_indicator : int
            Initial tension end for TENSION. To be used for Advanced Cable Analy
                - 0 = cable start or end node will not be considered.
                - 1 = cable start node to be considered.
                - 2 = cable end node to be considered.
        self_weight_factor_x : float
            Multiplying factor on self weight component applied in the global X
        self_weight_factor_y : float
            Multiplying factor on self weight component applied in the global Y
        self_weight_factor_z : float
            Multiplying factor on self weight component applied in the global Z

        Returns
        -------
        int
            Returns the assigned specification number id if successful, else ret

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> plate_ids = staad_obj.Geometry.GetPlateIDs()
        >>> specification_id = staad_obj.Property.CreateMemberCableSpecEx(1, 16
        """
        return self._property.CreateMemberCableSpecEx(tension_or_unstressed_len
```

[docs]

```
    def GetElementOffsetSpecCount(self):
        """
        Returns the total number of element offset specifications in the current

        Returns
        -------
        int
            Returns the total number of element offset specifications.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> count = staad_obj.Property.GetElementOffsetSpecCount()
        """
        return self._property.GetElementOffsetSpecCount()
```

[docs]

```
    def RemoveAllElementOffsetSpec(self):
        """
        Removes all element node offset specifications from the model.

        Returns
        -------
        int
            Returns 1 if OK else 0 if no element offset specifications present.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> output = staad_obj.Property.RemoveAllElementOffsetSpec()
        """
        return self._property.RemoveAllElementOffsetSpec()
```

[docs]

```
    def UpdatePropertiesToDesignSection(self):
        """
        Updates all the section properties that have been designed with a SELEC

        Returns
        -------
        int
            Returns 1 if assignment is successful else 0 if assignment is unsucc

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> count = staad_obj.Property.UpdatePropertiesToDesignSection()
        """
        return self._property.UpdatePropertiesToDesignSection()
```

[docs]

```
    def GetFireProofedBeamCount(self):
        """
        Returns count of beams which are fire proofed.

        Returns
        -------
        int
            Returns the total number of fire proofed beams in the current model

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> count = staad_obj.Property.GetFireProofedBeamCount()
        """
        return self._property.GetFireProofedBeamCount()
```

[docs]

```
    def GetFireProofedBeamList(self):
        """
        Returns a list of the member ids of all the fire proofed members in the

        Returns
        -------
        List of int
            Returns for list of member number ids of all the members that are fi

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> fire_proofed_beam_list = staad_obj.Property.GetFireProofedBeamList()
        """
        count = self._property.GetFireProofedBeamCount()
        safe_FireProofedBeamList = make_safe_array_long(count)
        vt_FireProofedBeamList= make_variant_vt_ref(safe_FireProofedBeamList, au
        result = self._property.GetFireProofedBeamList(vt_FireProofedBeamList)
        if result == 0:
            raise_os_error_if_error_code(-1)
        return vt_FireProofedBeamList[0]
```

[docs]

```
    def GetFireProofDataForBeam(self, beam_id: int):
        """
        Get fire proofing data for beam.

        Parameters
        ----------
        beam_id : int
            The beam number.

        Returns
        -------
        tuple: Tuple(int, float, float)
            Returns a tuple consisting of type of fire proof [1 for BFP, 2 for C

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> beam_ids = staad_obj.Property.GetBeamList()
        >>> count = staad_obj.Property.GetFireProofDataForBeam(beam_ids[0])
        """
        safe_FireProofType = make_safe_array_long(1)
        vt_FireProofType = make_variant_vt_ref(safe_FireProofType,  automation.V
        safe_Thickness = make_safe_array_double(1)
        vt_Thickness = make_variant_vt_ref(safe_Thickness,  automation.VT_R8)
        safe_Density = make_safe_array_double(1)
        vt_Density = make_variant_vt_ref(safe_Density,  automation.VT_R8)
        result = self._property.GetFireProofDataForBeam(beam_id, vt_FireProofTy
        if result == 0:
            raise_os_error_if_error_code(-1)
        return vt_FireProofType[0], vt_Thickness[0], vt_Density[0]
```

[docs]

```
def GetFireProofingSpecCount(self):
    """
    Returns the count of different fire proofing specifications in the model

    Returns
    -------
    int
        Returns the total number of fire proofing specification.

    Examples
    --------
    >>> from openstaadpy import os_analytical
    >>> staad_obj = os_analytical.connect()
    >>> count = staad_obj.Property.GetFireProofingSpecCount()
    """
    return self._property.GetFireProofingSpecCount()
```

[docs]

```
def GetFireProofingSpecDetails(self, index: int):
    """
    Get the details for the specified fire proofing specification number.

    Parameters
    ----------
    Index : int
        Non-zero based index of the fire proofing specification.

    Returns
    -------
    tuple : Tuple(int, float, float, int)
        Returns a tuple consisting of type of fire proof, thickness of fire

    Examples
    --------
    >>> from openstaadpy import os_analytical
    >>> staad_obj = os_analytical.connect()
    >>> count = staad_obj.Property.GetFireProofingSpecDetails(1)
    """
    safe_FireProofType = make_safe_array_long(1)
    vt_FireProofType = make_variant_vt_ref(safe_FireProofType,  automation.V
    safe_Thickness = make_safe_array_double(1)
    vt_Thickness = make_variant_vt_ref(safe_Thickness,  automation.VT_R8)
    safe_Density = make_safe_array_double(1)
    vt_Density = make_variant_vt_ref(safe_Density,  automation.VT_R8)
    safe_AssignCount = make_safe_array_long(1)
    vt_AssignCount= make_variant_vt_ref(safe_AssignCount, automation.VT_I4)
    result = self._property.GetFireProofingSpecDetails(index, vt_FireProofTy
    if result == 0:
        raise_os_error_if_error_code(-1)
    return vt_FireProofType[0], vt_Thickness[0], vt_Density[0], vt_AssignCo
```

[docs]

```
def GetFireProofingSpecAssignedBeamCount(self, index: int):
    """
    Get the count of beams assigned with a particular fire proofing specifi

    Parameters
    ----------
    index : int
        Non-zero based index of the fire proofing specification.

    Returns
    -------
    int
        Returns the number of beams assigned with a particular fire proofing

    Examples
    --------
    >>> from openstaadpy import os_analytical
    >>> staad_obj = os_analytical.connect()
    >>> count = staad_obj.Property.GetFireProofingSpecAssignedBeamCount(5)
    """
    return self._property.GetFireProofingSpecAssignedBeamCount(index)
```

[docs]

```
def GetFireProofingSpecAssignedBeamList(self, index: int):
    """
    Populates a list of the member ID(s) of all the members assigned to a pa

    Parameters
    ----------
    index : int
        Non-zero based index of the fire proofing specification.

    Returns
    -------
    List
        Returns for list of member numbers IDs of all the members that are t

    Examples
    --------
    >>> from openstaadpy import os_analytical
    >>> staad_obj = os_analytical.connect()
    >>> count = staad_obj.Property.GetFireProofingSpecAssignedBeamList(5)
    """
    count = self._property.GetFireProofingSpecAssignedBeamCount(index)
    safe_FireProofedBeamList = make_safe_array_long(count)
    vt_FireProofedBeamList= make_variant_vt_ref(safe_FireProofedBeamList, a
    result = self._property.GetFireProofingSpecAssignedBeamList(index, vt_F
    if result == 0:
        raise_os_error_if_error_code(-1)
    return vt_FireProofedBeamList[0]
```

[docs]

```
    def CreateMemberFireProofingSpec(self, fire_proof_type: int, thickness_value
        """
        Create MEMBER FIREPROOFING specification.

        Parameters
        ----------
        fire_proof_type : int
            Specify type of fire proofing:
                - 1 = BFP Block Fireproofing.
                - 2 = CFP Contour Fireproofing.
        thickness_value : float
            Thickness of the Fireproofing
        density : float
            Density of the Fireproofing material

        Returns
        -------
        int
            Returns zero based index for the newly created specification if succ

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> count = staad_obj.Property.CreateMemberFireProofingSpec(1, 3.5, 6.4)
        """
        return self._property.CreateMemberFireProofingSpec(fire_proof_type, thic
```

[docs]

```
    def RemoveMemberFireProofingSpecFromBeam(self, beam_id: int):
        """
        Remove member fire proofing specification from beam.

        Parameters
        ----------
        beam_id : int
            The beam number ID.

        Returns
        -------
        int
            Returns 1 if fire proofing specification removed from beam else 0 if

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> count = staad_obj.Property.RemoveMemberFireProofingSpecFromBeam(1)
        """
        return self._property.RemoveMemberFireProofingSpecFromBeam(beam_id)
```

[docs]

```
    def GetBeamSectionDisplayName(self, beam_id: int):
        """
        This function returns the display section name of the specified beam.

        Parameters
        ----------
        beam_id : int
            The beam number ID

        Returns
        -------
        str
            Returns the section string name if successful else empty string if t
                +----------+-------------------------------------------+--------
                | Sl No.   | Section Type                              | In STD
                +==========+===========================================+========
                | 1        | Standard Section from Steel Database      | | TABLE
                |          |                                           | | TABLE
                |          |                                           | | 5 TAB
                +----------+-------------------------------------------+--------
                | 2        | Pipe and Tube definition                  | | 8 TAB
                |          |                                           | | 8 TAB
                +----------+-------------------------------------------+--------
                | 3        | Prismatic                                 | | 3 PRI
                |          |                                           | | 8 PRI
                +----------+-------------------------------------------+--------
                | 4        | Tapered                                   | 3 TAPER
                +----------+-------------------------------------------+--------
                | 5        | Assign Profile                            | 3 ASSIG
                +----------+-------------------------------------------+--------
                | 6        | User Provided Table                       | 14 TO 2
                +----------+-------------------------------------------+--------

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> beam_ids = staad_obj.Geometry.GetBeamList()
        >>> count = staad_obj.Property.GetBeamSectionDisplayName(beam_ids[0])
        """
        return self._property.GetBeamSectionDisplayName(beam_id)
```

[docs]

```
    def SetStandardProfileDBFolder(self, folder_name:str ):
        """
        Sets standard profile database folder path.

        Parameters
        ----------
        folder_name : str
            Path of the folder.

        Returns
        -------
        int
            Returns 0 if successful else -1 if error (If path is empty or does r

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> count = staad_obj.Property.SetStandardProfileDBFolder("C:\Staad.Pro
        """
        return self._property.SetStandardProfileDBFolder(folder_name)
```

[docs]

```
    def GetStandardProfileDBFolder(self):
        """
        Gets standard profile default database folder path.

        Returns
        -------
        str
            Returns the standard profile database folder path.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> count = staad_obj.Property.GetStandardProfileDBFolder()
        """
        return self._property.GetStandardProfileDBFolder()
```

[docs]

```
    def GetDefaultStandardProfileDBFolder(self):
        """
        Gets standard profile default database folder path.

        Returns
        -------
        str
            Returns the standard profile default database folder path.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> folder_path = staad_obj.Property.GetDefaultStandardProfileDBFolder()
        """
        return self._property.GetDefaultStandardProfileDBFolder()
```

[docs]

```
    def IsStandardDatabaseSection(self, section_reference_id: int):
        """
        Checks if the specified section property reference number is from standa

        Parameters
        ----------
        section_reference_id : int
            The section property reference ID.

        Returns
        -------
        bool
            Returns 'True' if section source is standard database else 'False' i

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> count = staad_obj.Property.IsStandardDatabaseSection(1)
        """
        return self._property.IsStandardDatabaseSection(section_reference_id)
```

[docs]

```
    def GetStandardSectionDatabaseName(self, section_property_id: int):
        """
        Gets standard section database name for the specified section property r

        Parameters
        ----------
        section_property_id : int
            The section property reference ID.

        Returns
        -------
        str
            Returns <Non-Empty-String> if the standard section database name if

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> count = staad_obj.Property.GetStandardSectionDatabaseName(4)
        """
        return self._property.GetStandardSectionDatabaseName(section_property_i
```

[docs]

```
    def GetStandardSectionTableName(self, section_reference_id: int):
        """
        Get the section name from the standard section database and table for th

        Parameters
        ----------
        section_reference_id : int
            The section property reference ID.

        Returns
        -------
        str
            Returns <Non-Empty-String> if the standard section database name if

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> count = staad_obj.Property.GetStandardSectionTableName(1)
        """
        return self._property.GetStandardSectionTableName(section_reference_id)
```

[docs]

```
    def GetStandardSectionName(self, section_reference_id: int):
        """
        Get the section name from the standard section database and table for th

        Parameters
        ----------
        section_reference_id : int
            The section property reference ID.

        Returns
        -------
        int
            Returns <Non-Empty-String> if the standard section database name if

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> count = staad_obj.Property.GetStandardSectionName()
        """
        return self._property.GetStandardSectionName(section_reference_id)
```