```
#--------------------------------------------------------------------------------
# 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 client
from comtypes import CoInitialize
from .oserrors import *
```

---

[docs]

```
class OSSupport:
    CoInitialize()
```

[docs]

```
    def __init__(self, staadObj):
        self._staad = staadObj
        self._support = self._staad.Support

        self._functions= [
            "AssignSupportToNode",
            "CreateSupportFixed",
            "CreateSupportPinned",
            "CreateSupportFixedBut",
            "GetSupportCount",
            "GetSupportNodes",
            "GetSupportType",
            "GetSupportInformation",
            "GetSupportUniqueID",
            "SetSupportUniqueID",
            "RemoveSupportFromNode",
            "DeleteSupport",
            "GetSupportName",
            "GetSupportInformationEx",
            "CreateInclinedSupport",
            "CreateElasticMat",
            "GetCountOfElasticMat",
            "GetElasticMatDetail",
            "GetElasticMatAssignmentList",
            "RemoveElasticMat",
            "RemoveElasticMatFromNode",
            "AssignSupportToEntityList",
            "CreatePlateMat",
            "GetCountOfPlateMat",
            "GetPlateMatSupportId",
            "GetPlateMatDetail",
            "GetPlateMatAssignmentList",
            "RemovePlateMat",
            "RemovePlateMatFromPlate",
            "CreateElasticFooting",
            "GetCountOfElasticFooting",
            "GetElasticFootingDetail",
            "GetElasticFootingAssignmentList",
            "RemoveElasticFooting",
```

```
            "RemoveElasticFootingFromNode"
        ]

        for function_name in self._functions:
            self._support._FlagAsMethod(function_name)



    ## SUPPORT FUNCTIONS



                                                                                [docs]
    def AssignSupportToNode(self, NodeIDs: list|int, SupportID: int):
        """
        Assign a support to one or more nodes.

        Parameters
        ----------
        NodeIDs : list of int or int
            List of node numbers or a single node number to assign the support t
        SupportID : int
            Support reference number ID.

        Returns
        -------
        None

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> fixed_id = staad_obj.Support.CreateSupportFixed()
        >>> staad_obj.Support.AssignSupportToNode([1, 2, 3], fixed_id) // Passir
        >>> staad_obj.Support.AssignSupportToNode(5, fixed_id) //Passing support
        """
        if (isinstance(NodeIDs, int)):
            NodeIDs = [NodeIDs]
        safe_list = make_safe_array_long_input(NodeIDs)
        self._support.AssignSupportToNode(safe_list, SupportID)




                                                                                [docs]
    def CreateSupportFixed(self):
        """
        Creates a fully fixed support.

        Returns
        -------
        int
            Support reference number ID.
            -1 indicates General Error.

        Examples
        --------
```

```
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> count = staad_obj.Support.CreateSupportFixed()
        """
        return self._support.CreateSupportFixed()
```

[docs]

```
    def CreateSupportPinned(self):
        """
        Creates a pinned support (i.e., free to rotate about local y and z axis,

        Returns
        -------
        int
            Support reference number ID.
            -1 indicates General Error.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> count = staad_obj.Support.CreateSupportPinned()
        """
        return self._support.CreateSupportPinned()
```

[docs]

```
    def CreateSupportFixedBut(self, ReleaseSpec:list, SpringSpec:list):
        """
        Creates fixed support with releases in specified directions or a spring

        Returns
        -------
        int
            Support reference number ID.
            -1 indicates General Error.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> count = staad_obj.Support.CreateSupportFixedBut(ReleaseSpec, SpringS
        """
        release = make_safe_array_double_input(ReleaseSpec)
        spring = make_safe_array_double_input(SpringSpec)
        release_vt = make_variant_vt_ref(release, automation.VT_ARRAY | automat
        spring_vt = make_variant_vt_ref(spring, automation.VT_ARRAY | automatio
        return self._support.CreateSupportFixedBut(release_vt, spring_vt)
```

[docs]

```
def GetSupportCount(self):
    """
    Get the total number of supported nodes in the current structure.

    Returns
    -------
    int
        Number of supported nodes.

    Examples
    --------
    >>> from openstaadpy import os_analytical
    >>> staad_obj = os_analytical.connect()
    >>> count = staad_obj.Support.GetSupportCount()
    """
    return self._support.GetSupportCount()
```

[docs]

```
def GetSupportNodes(self):
    """
    Get all supported node numbers.

    Returns
    -------
    list of int
        List of supported node numbers.

    Examples
    --------
    >>> from openstaadpy import os_analytical
    >>> staad_obj = os_analytical.connect()
    >>> nodes = staad_obj.Support.GetSupportNodes()
    """
    count = self.GetSupportCount()
    safe_list = make_safe_array_long(count)
    node_list = make_variant_vt_ref(safe_list, automation.VT_ARRAY | automa
    self._support.GetSupportNodes(node_list)
    return node_list[0]
```

[docs]

```
def GetSupportType(self, nodeNo: int):
    """
    Get the support type for the specified node.

    Parameters
    ----------
    nodeNo : int
        Node number.
```

```
        Returns
        -------
        int
            Support type code.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> staad_obj.Support.GetSupportType(1)
        """
        return self._support.GetSupportType(nodeNo)
```

[docs]

```
    def GetSupportInformation(self, nodeNo: int):
        """
        Get support information for the specified node.

        Parameters
        ----------
        nodeNo : int
            Node number.

        Returns
        -------
        tuple
            Returns a tuple consisting of support_type, list of release specific

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> stype, release, spring = staad_obj.Support.GetSupportInformation(1)
        """
        release = make_safe_array_long(6)
        spring = make_safe_array_double(6)
        release_vt = make_variant_vt_ref(release, automation.VT_ARRAY | automat
        spring_vt = make_variant_vt_ref(spring, automation.VT_ARRAY | automatio
        stype = self._support.GetSupportInformation(nodeNo, release_vt, spring_
        return stype, list(release_vt[0]), list(spring_vt[0])
```

[docs]

```
    def GetSupportUniqueID(self, supportNo: int):
        """
        Get unique ID GUID string for a support item.

        Parameters
        ----------
        supportNo : int
```

```
            Support item number.

        Returns
        -------
        str
            GUID string.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> staad_obj.Support.GetSupportUniqueID(2)
        """
        return self._support.GetSupportUniqueID(supportNo)
```

[docs]

```
    def SetSupportUniqueID(self, supportNo: int, guid: str):
        """
        Set unique ID for a support item.

        Parameters
        ----------
        supportNo : int
            Support item number.
        guid : str
            GUID string.

        Returns
        -------
        None

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> staad_obj.Support.SetSupportUniqueID(2, "supportId")
        """
        self._support.SetSupportUniqueID(supportNo, guid)
```

[docs]

```
    def RemoveSupportFromNode(self, NodeIDs: list):
        """
        Remove support from one or more nodes.

        Parameters
        ----------
        NodeIDs : list of int
            List of node numbers from which to remove the support.

        Returns
```

```
        -------
        None

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> staad_obj.Support.RemoveSupportFromNode([1, 2, 3])
        """
        safe_list = make_safe_array_long_input(NodeIDs)
        NodeIDs_vt = make_variant_vt_ref(safe_list, automation.VT_ARRAY | automa
        self._support.RemoveSupportFromNode(NodeIDs_vt)
```

[docs]

```
    def DeleteSupport(self, supportNo: int):
        """
        Remove a support item from the model.

        Parameters
        ----------
        supportNo : int
            Support item number.

        Returns
        -------
        bool
            True if successful.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> status = staad_obj.Support.DeleteSupport(2)
        >>> print(status)
        """
        return self._support.DeleteSupport(supportNo)
```

[docs]

```
    def GetSupportName(self, supportNo: int):
        """
        Get support string name.

        Parameters
        ----------
        supportNo : int
            Support item number.

        Returns
        -------
        str
```

```
            Support name.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> staad_obj.Support.GetSupportName(2)
        """
        return self._support.GetSupportName(supportNo)




                                                                    [docs]
    def GetSupportInformationEx(self, nodeNo: int):
        """
        Get extended support information for the specified node.

        Parameters
        ----------
        nodeNo : int
            Node number.

        Returns
        -------
        tuple
            (supportNo, supportType, releaseSpec, springSpec)

            - supportNo
                Support item number.

            - supportType
                Support type code.

            - releaseSpec
                List of release specifications. ( = 1) or Fixed ( = 0) or Spring

            - springSpec
                List of spring specifications. [KFX, KFY, KFZ, KMX, KMY, KMZ]

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> staad_obj.Support.GetSupportInformationEx(1)
        """
        supportNo = make_variant_vt()
        supportNo_ref = make_variant_vt_ref(supportNo,  automation.VT_I4)

        supportType = make_variant_vt()
        supportType_ref = make_variant_vt_ref(supportType,  automation.VT_I4)

        release = make_safe_array_long(6)
        spring = make_safe_array_double(6)
        release_vt = make_variant_vt_ref(release, automation.VT_ARRAY | automat
        spring_vt = make_variant_vt_ref(spring, automation.VT_ARRAY | automatio
```

```
        retval = self._support.GetSupportInformationEx(nodeNo, supportNo_ref, su
        if not bool(retval):
            raise_os_error_if_error_code(-1)
        return (supportNo_ref[0], supportType_ref[0], list(release_vt[0]), list
```

[docs]

```
    def CreateInclinedSupport(self, inclinedType:int, refType:int, refNode:int,
        """
        Create an inclined support.

        Parameters
        ----------
        inclinedType : int
            Type of the inclined support:
                +-------+---------------+
                | Value | Inclined Type |
                +=======+===============+
                | 1     | Pinned        |
                +-------+---------------+
                | 2     | Fixed         |
                +-------+---------------+
                | 3     | FixedBut      |
                +-------+---------------+
                | 4     | Enforced      |
                +-------+---------------+
                | 5     | EnforcedBut   |
                +-------+---------------+

        refType : int
            Type of the reference point:
                +-------+-----------------------------------------------------
                | Value | Table Type
                +=======+=====================================================
                | 0     |  fRefX,  fRefY,  fRefY global distances from the joint
                +-------+-----------------------------------------------------
                | 1     |  fRefX,  fRefY,  fRefY global coordinates of the refer
                +-------+-----------------------------------------------------
                | 2     | a joint number ( vaRefNode) whose x, y, z global coord
                +-------+-----------------------------------------------------

        refNode : int
            Reference node number.
        coord : list of float
            Reference coordinates. [X, Y, Z]
        releaseSpec : list of float
            Release specification. Fixed (= 0) or Release (= 1) [FX, FY, FZ, MX,
        springSpec : list of float
            Spring specification. [KFX, KFY, KFZ, KMX, KMY, KMZ]

        Returns
        -------
        int
            Support reference number ID.
```

```
        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> support_id = staad_obj.Support.CreateInclinedSupport(2, 2, 1, [0,0,0
        >>> print(support_id)
        """
        coord_vt = make_safe_array_double_input(coord)
        release_vt = make_safe_array_double_input(releaseSpec)
        spring_vt = make_safe_array_double_input(springSpec)
        result = self._support.CreateInclinedSupport(inclinedType, refType, refN
        if result < 0:
            raise_os_error_if_error_code(result)
        return result
```

[docs]

```
    def CreateElasticMat(self, direction, subgrade, printFlag, springType):
        """
        Create an elastic mat support.

        Parameters
        ----------
        direction : int
            Direction.
        subgrade : float
            Subgrade modulus.
        printFlag : int
            Print flag.
        springType : int
            Spring type.

        Returns
        -------
        int
            Support reference number ID.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> support_id = staad_obj.Support.CreateElasticMat(5, 20, 0, 1)
        >>> print(support_id)
        """
        return self._support.CreateElasticMat(direction, subgrade, printFlag, sp
```

[docs]

```
    def GetCountOfElasticMat(self):
        """
        Get the total number of ElasticMat supports.
```

```
        Returns
        -------
        int
            Number of ElasticMat supports.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> support_count = staad_obj.Support.GetCountOfElasticMat()
        >>> print(support_count)
        """
        return self._support.GetCountOfElasticMat()
```

[docs]

```
    def GetElasticMatDetail(self, supportid):
        """
        Get elastic mat support information for a specific support Id.

        Parameters
        ----------
        supportid : int
            Support reference ID.

        Returns
        -------
        tuple
            (direction, subgrade, printFlag, springType, nodesCount)

            - direction : int
                +-------+-------------------+
                | Value | Inclined Type     |
                +=======+===================+
                | 0     | X Direction       |
                +-------+-------------------+
                | 1     | Y Direction       |
                +-------+-------------------+
                | 2     | Z Direction       |
                +-------+-------------------+
                | 3     | X Only Direction  |
                +-------+-------------------+
                | 4     | Y Only Direction  |
                +-------+-------------------+
                | 5     | Z Only  Direction |
                +-------+-------------------+

            - subgrade : float
                Subgrade modulus.

            - printFlag : bool
                Print flag. True if checked, False if not.
```

```
            - springType : int
                +-------+------------------+
                | Value | Spring Type      |
                +=======+==================+
                | 0     | None             |
                +-------+------------------+
                | 1     | Compression only |
                +-------+------------------+
                | 2     | Multi-linear     |
                +-------+------------------+

            - nodesCount : int
                Number of nodes assigned to this support.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> details = staad_obj.Support.GetElasticMatDetail(4)
        >>> print(details)
        """
        dir = make_variant_vt()
        dir_ref = make_variant_vt_ref(dir, automation.VT_I4)
        subgrade = make_variant_vt()
        subgrade_ref = make_variant_vt_ref(subgrade, automation.VT_R8)
        printFlag = make_variant_vt()
        printFlag_ref = make_variant_vt_ref(printFlag, automation.VT_I4)
        springType = make_variant_vt()
        springType_ref = make_variant_vt_ref(springType, automation.VT_I4)
        nodesCount = make_variant_vt()
        nodesCount_ref = make_variant_vt_ref(nodesCount, automation.VT_I4)
        self._support.GetElasticMatDetail(supportid, dir_ref, subgrade_ref, pri
        return dir_ref[0], subgrade_ref[0], bool(printFlag_ref[0]), springType_
```

[docs]

```
    def GetElasticMatAssignmentList(self, supportid):
        """
        Get elastic mat support entity list for a specific support Id.

        Parameters
        ----------
        supportid : int
            Support reference ID.

        Returns
        -------
        list of int
            List of node numbers.

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

```
        >>> node_list = staad_obj.Support.GetElasticMatAssignmentList(4)
        >>> print(node_list)
        """
        # Get node count first
        _, _, _, _, nodesCount = self.GetElasticMatDetail(supportid)
        if nodesCount == 0:
            return []
        safe_list = make_safe_array_long(nodesCount)
        node_list = make_variant_vt_ref(safe_list, automation.VT_ARRAY | automa
        retval = self._support.GetElasticMatAssignmentList(supportid, node_list
        if not bool(retval):
            return []
        return node_list[0]
```

[docs]

```
    def RemoveElasticMat(self, supportid):
        """
        Remove elastic mat support for a specific support Id.

        Parameters
        ----------
        supportid : int
            Support reference ID.

        Returns
        -------
        bool
            True if successful.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> status = staad_obj.Support.RemoveElasticMat(4)
        >>> print(status)
        """
        return self._support.RemoveElasticMat(supportid)
```

[docs]

```
    def RemoveElasticMatFromNode(self, nodeid):
        """
        Remove elastic mat support from a specific node.

        Parameters
        ----------
        nodeid : int
            Node number.

        Returns
        -------
```

```
        bool
            True if successful.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> status = staad_obj.Support.RemoveElasticMatFromNode(7)
        >>> print(status)
        """
        return bool(self._support.RemoveElasticMatFromNode(nodeid))
```

[docs]

```
    def AssignSupportToEntityList(self, supportid, entitylist):
        """
        Assign the specified support to an entity list.

        Parameters
        ----------
        supportid : int
            Support reference ID.
        entitylist : list of int
            List of node or plate numbers.

        Returns
        -------
        bool
            True if successful.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> status = staad_obj.Support.AssignSupportToEntityList(3, [41, 42])
        >>> print(status)
        """
        safe_list = make_safe_array_long_input(entitylist)
        retval = self._support.AssignSupportToEntityList(supportid, safe_list)
        return bool(retval)
```

[docs]

```
    def CreatePlateMat(self, direction:int, subgrades, printFlag:bool, springTy
        """
        Create a plate mat support.

        Parameters
        ----------
        direction : int
            +-------+-------------------+
            | Value | Inclined Type     |
```

```
        +=======+====================+
        | 0     | X Direction        |
        +-------+--------------------+
        | 1     | Y Direction        |
        +-------+--------------------+
        | 2     | Z Direction        |
        +-------+--------------------+
        | 3     | X Only Direction   |
        +-------+--------------------+
        | 4     | Y Only Direction   |
        +-------+--------------------+
        | 5     | Z Only  Direction  |
        +-------+--------------------+
        | 6     | All  Direction     |
        +-------+--------------------+
```

```
    subgrades : list of float or float
        Subgrade modulus value(s).
    printFlag : bool
        Print flag.
    springType : int
        Spring type.

    Returns
    -------
    int
        Support reference number ID. / 0 if error.

    Examples
    --------
    >>> from openstaadpy import os_analytical
    >>> staad_obj = os_analytical.connect()
    >>> support_id = staad_obj.Support.CreatePlateMat(2, [20, 30, 40], False
    >>> print(support_id)
    """
    if isinstance(subgrades, float):
        subgrades = [subgrades]
    subgrades_vt = make_safe_array_double_input(subgrades)
    return self._support.CreatePlateMat(direction, subgrades_vt, int(printF
```

[docs]

```
def GetCountOfPlateMat(self):
    """
    Get the total number of plate mat supports.

    Returns
    -------
    int
        Number of plate mat supports.

    Examples
    --------
    >>> from openstaadpy import os_analytical
```

```
        >>> staad_obj = os_analytical.connect()
        >>> support_count = staad_obj.Support.GetCountOfPlateMat()
        >>> print(support_count)
        """
        return self._support.GetCountOfPlateMat()
```

[docs]

```
    def GetPlateMatSupportId(self, plateMatIndex):
        """
        Get the plate mat support ID.

        Parameters
        ----------
        plateMatIndex : int
            PlateMat index (starting from 0).

        Returns
        -------
        int
            Plate mat support ID.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> support_id = staad_obj.Support.GetPlateMatSupportId(1)
        >>> print(support_id)
        """
        retval = self._support.GetPlateMatSupportId(plateMatIndex)
        if retval < 0:
            raise_os_error_if_error_code(-1)
        return retval
```

[docs]

```
    def GetPlateMatDetail(self, plateMatNo):
        """
        Get plate mat support information for a specific support Id.

        Parameters
        ----------
        plateMatNo : int
            Plate mat support ID.

        Returns
        -------
        tuple
            (direction, subgrade1, subgrade2, subgrade3, printFlag, springType,

        Examples
        --------
```

```
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> details = staad_obj.Support.GetPlateMatDetail(1)
        >>> print(details)
        """
        direction = make_safe_array_int(1)
        direction_ref = make_variant_vt_ref(direction, automation.VT_I4)
        subgrade1 = make_safe_array_double(1)
        subgrade1_ref = make_variant_vt_ref(subgrade1, automation.VT_R8)
        subgrade2 = make_safe_array_double(1)
        subgrade2_ref = make_variant_vt_ref(subgrade2, automation.VT_R8)
        subgrade3 = make_safe_array_double(1)
        subgrade3_ref = make_variant_vt_ref(subgrade3, automation.VT_R8)
        printFlag = make_safe_array_int(1)
        printFlag_ref = make_variant_vt_ref(printFlag, automation.VT_I4)
        springType = make_safe_array_int(1)
        springType_ref = make_variant_vt_ref(springType, automation.VT_I4)
        nAssignedPlateCount = make_safe_array_int(1)
        nAssignedPlateCount_ref = make_variant_vt_ref(nAssignedPlateCount, autor
        retval = self._support.GetPlateMatDetail(plateMatNo, direction_ref, subg
        if not bool(retval):
            raise_os_error_if_error_code(-1)
        return (direction_ref[0], subgrade1_ref[0], subgrade2_ref[0], subgrade3_
```

[docs]

```
    def GetPlateMatAssignmentList(self, plateMatNo):
        """
        Get plate mat support entity list for a specific support Id.

        Parameters
        ----------
        plateMatNo : int
            Plate mat support ID.

        Returns
        -------
        list of int
            List of plate numbers.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> plate_list = staad_obj.Support.GetPlateMatAssignmentList(1)
        >>> print(plate_list)
        """
        # Get plate count first
        _, _, _, _, _, _, nAssignedPlateCount = self.GetPlateMatDetail(plateMatN
        if nAssignedPlateCount == 0:
            return []
        safe_list = make_safe_array_long(nAssignedPlateCount)
        plate_list = make_variant_vt_ref(safe_list, automation.VT_ARRAY | automa
        retval = self._support.GetPlateMatAssignmentList(plateMatNo, plate_list
```

```
        if not bool(retval):
            return []
        return plate_list[0]
```

[docs]

```
    def RemovePlateMat(self, supportId):
        """
        Remove plate mat support for a specific support Id.

        Parameters
        ----------
        supportId : int
            Plate mat support ID.

        Returns
        -------
        bool
            True if successful.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> status = staad_obj.Support.RemovePlateMat(4)
        >>> print(status)
        """
        return bool(self._support.RemovePlateMat(supportId))
```

[docs]

```
    def RemovePlateMatFromPlate(self, plateNo:int):
        """
        Remove plate mat support from a specific plate.

        Parameters
        ----------
        plateNo : int
            Plate number.

        Returns
        -------
        bool
            True if successful.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> status = staad_obj.Support.RemovePlateMatFromPlate(56)
        >>> print(status)
```

```
        """
        return bool(self._support.RemovePlateMatFromPlate(plateNo))
```

[docs]

```
    def CreateElasticFooting(self, length, width, direction, subgrade):
        """
        Create an elastic footing support.

        Parameters
        ----------
        length : float
            Length of footing.
        width : float
            Width of footing.
        direction : int
            Direction.
        subgrade : float
            Subgrade modulus.

        Returns
        -------
        int
            Support reference number ID.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> support_id = staad_obj.Support.CreateElasticFooting(5, 6, 2, 20)
        >>> print(support_id)
        """
        return self._support.CreateElasticFooting(length, width, direction, subg
```

[docs]

```
    def GetCountOfElasticFooting(self):
        """
        Get the total number of elastic footing supports.

        Returns
        -------
        int
            Number of elastic footing supports.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> count = staad_obj.Support.GetCountOfElasticFooting()
        >>> print(count)
```

```
        """
        return self._support.GetCountOfElasticFooting()
```

[docs]

```
    def GetElasticFootingDetail(self, supportid):
        """
        Get elastic footing support information for a specific support Id.

        Parameters
        ----------
        supportid : int
            Support reference ID.

        Returns
        -------
        tuple
            (length, width, direction, subgrade, nodesCount)

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> details = staad_obj.Support.GetElasticFootingDetail(2)
        >>> print(details)
        """
        length = make_safe_array_double(1)
        length_ref = make_variant_vt_ref(length, automation.VT_R8)
        width = make_safe_array_double(1)
        width_ref = make_variant_vt_ref(width, automation.VT_R8)
        direction = make_safe_array_int(1)
        direction_ref = make_variant_vt_ref(direction, automation.VT_I4)
        subgrade = make_safe_array_double(1)
        subgrade_ref = make_variant_vt_ref(subgrade, automation.VT_R8)
        nodesCount = make_safe_array_int(1)
        nodesCount_ref = make_variant_vt_ref(nodesCount, automation.VT_I4)
        retval = self._support.GetElasticFootingDetail(supportid, length_ref, w
        if retval == 0:
            raise_os_error_if_error_code(-1)
        return (length_ref[0], width_ref[0], direction_ref[0], subgrade_ref[0],
```

[docs]

```
    def GetElasticFootingAssignmentList(self, supportid):
        """
        Get list of assigned node Ids for a specific elastic footing support Id

        Parameters
        ----------
        supportid : int
            Support reference ID.
```

```
        Returns
        -------
        list of int
            List of node numbers.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> node_list = staad_obj.Support.GetElasticFootingAssignmentList(3)
        >>> print(node_list)
        """
        _, _, _, _, nodesCount = self.GetElasticFootingDetail(supportid)
        if nodesCount == 0:
            return []
        safe_list = make_safe_array_long(nodesCount)
        node_list = make_variant_vt_ref(safe_list, automation.VT_ARRAY | automa
        retval = self._support.GetElasticFootingAssignmentList(supportid, node_
        if retval == 0:
            return []
        return node_list[0]
```

[docs]

```
    def RemoveElasticFooting(self, supportid):
        """
        Remove elastic footing support for a specific support Id.

        Parameters
        ----------
        supportid : int
            Support reference ID.

        Returns
        -------
        bool
            True if successful.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> status = staad_obj.Support.RemoveElasticFooting(3)
        >>> print(status)
        """
        return self._support.RemoveElasticFooting(supportid)
```

[docs]

```
    def RemoveElasticFootingFromNode(self, nodeid):
        """
        Remove elastic footing support from a specific node.
```

```
        Parameters
        ----------
        nodeid : int
            Node number.

        Returns
        -------
        bool
            True if successful.

        Examples
        --------
        >>> from openstaadpy import os_analytical
        >>> staad_obj = os_analytical.connect()
        >>> status = staad_obj.Support.RemoveElasticFootingFromNode(2)
        >>> print(status)
        """
        return self._support.RemoveElasticFootingFromNode(nodeid)
```