USER MANUAL

LIBRARY : Toolkit_WorkPackage

This python library contains wrapper functions to the Sir3S_Toolkit.dll

This package provides 2 different classes, SIR3S_Model() and SIR3S_View() to access functions within dll. SIR3S_Model() deals with all the functions directly dealing with general operation performed on a model(including creating new model). On the other side SIR3S_View() offers wrapped function to deal with visual representation on the model open. This can be used to add visual elements and modify their properties and it is recommended and advised to use this class in the context of SIRGraf(with the help of python console), for clear visualization.

In this document we will understand each function provided by both classes and their usage techniques and limitations. Although source code exposes enough information about the functions available in the class, it will be in everyone's interest to read and understand the user manual for better usage.

A. Class: SIR3S_Model

1. StartTransaction(SessionName:str)

Summary: If Modifications on a Model are intended, it is recommended to make a Call of StartTransaction(), then do all the Modifications you need. And then call EndTransaction() as soon as you are finished with Modifications. This helps the Software to keep Track of Modifications, so the User can Undo/Redo them on the main UI (SirGraf)

Input Parameters:
SessionName(string) - give a meaningful name to start a transaction; Empty string or None will lead to error

Output Parameters:
None

2. EndTransaction()

Summary: If Modifications on a Model are intended, it is recommended to make a Call of StartTransaction(), then do all the Modifications you need. Ant then call EndTransaction() as soon as you are finished with Modifications. This helps the Software to keep Track of Modifications, so the User can Undo/Redo them on the main UI (SirGraf)

Input Parameters:

None

Output Parameters:

None

3. **StartEditSession(SessionName:str)**

Summary: Recommended for fast bulk Changes (e.g. Changing the Values of 40 thousand Nodes in a single Task). Similar to StartTransaction(), EndEditSession() should be called after the Caller is done with all his bulk Changes.

Input Parameters:

SessionName(string) - give a meaningful name to start a transaction; Empty string or None will lead to error

Output Parameters:

None

4. **EndEditSession()**

Summary: Closes an already started EditSession. Should always be called after a Call of StartEditSession() and all the Modifications applied.

Input Parameters:

None

Output Parameters:

None

5. **GetCurrentTimeStamp()**

Summary: This is a wrapper method to access the get method for the property CurrentTimestamp from toolkit

Input Parameters:
None

Output Parameters:
None

6. **SetCurrentTimeStamp(timestamp:str)**

   Summary: This is a wrapper method to access the set method for the property CurrentTimestamp from toolkit

   Input parameters:

   Timestamp(string) - Time stamp value to be set

   Output parameters:

   None

7. **GetValue(Tk:str, propertyName:str)**

   Summary: Reads the Value of the Property of an Element and also return the Type name [string/float/double/int/bool] of that Property

   Input Parameters:

   Tk – Tk of the element

   propertyName - The Name of the Property

   Output Parameters:

   value – returns the expected value

8. **SetValue (Tk:str, propertyName:str, value:str)**

   Summary: Sets the Value of the Property of an Element

   Input Parameters:

   Tk – Tk of the element

   propertyName - The Name of the Property

   value – value to be set

   Output Parameters:

   None

9. **OpenModelXml (Path:str, SaveCurrentModel:bool)**

   Summary: Opens a SIR 3S Modell from a XML File. Eventually currently open Model will be closed before

Input Parameters:

Path – The full Path to the XML File

SaveCurrentModel - If TRUE, an eventually currently open Model will be saved before closing

Output Parameters:

None

10. **OpenModel(dbName: str, providerType: Interfaces.SirDBProviderType, Mid: str, saveCurrentlyOpenModel: bool, namedInstance: str, userID: str, password: str)**

Summary: General Method for opening Model in XML File, Access DB, ORACLE DB, SQLServer DB, and SQLite DB

Input Parameters:

dbName – For ACCESS, XML and SQLITE the Full Path to the Database File,

For SQLServer, the Database Name (eg. 'ModelDB').

For Oracle the Instance Name or Alias (eg. 'localhost/XE', 'ProdDB')

providerType - Provider Type (Enumeration)

should be one of the following:

'SirDBProviderType.XMLFILE' for XML File,

'SirDBProviderType.ACCESS' for ACCESS-Database,

'SirDBProviderType.ORACLE' for ORACLE-Database,

'SirDBProviderType.SQLite' for SQLite-Database,

'SirDBProviderType.SQLSERVER' for SQL Server-Database,

In Case of SQLExpress, we favourite mounting SQLExpress Data Files into a SQLExpress

Database Instance and handle it similar as regular SQL Server Database.

Mid – Model Identifier

saveCurrentlyOpenModel - If TRUE, an eventually currently open Model will be saved before closing

namedInstance - Named Instance, only required for SQLServer.

userID - User Id for Authentication, only needed for ORACLE and for SQLServer only if SQLServer Authentication is required.

Password - Password for Authentication, only needed for ORACLE and for SQLServer only if SQLServer Authentication is required.

For SQLServer, just imput an empty String if Server accepts trusted Connection

Output Parameters:

None

11. **ExecCalculation(waitForSirCalcToExit: bool)**

    Summary: Executes Model Calculation with SirCalc.

    Input Parameters:

    waitForSirCalcToExit – If this Parameter is set to True, the Program Flow would

    wait until the Sircal-Process exits before continuing

    Output Parameters:

    None

12. **GetTimeStamps ()**

    Summary: Gets an Array with all available Timestamps as ISO formatted Strings

    Input Parameters:

    None

    Output Parameters:

    timestamps - Array with all available Timestamps as ISO formatted Strings

    tsStat - Timestamp for stationary Results

    tsMin - Timestamp for minimum Results

    tsMax - Timestamp for maximum Results

13. **GetTksofElementType (ElementType: Interfaces.Sir3SObjectTypes)**

    Summary: Retrieves a List with all TK (if ElementType has any, otherwise PK)

    Input Parameters:

    ElementType - The Type of Element in the Repository

    Output Parameters:

    Tk_list - list of all Tk's belonging to the elements of type 'ElementType'

14. **GetNetworkType ()**

    Summary: Gets the Network Type of the current Model

    Input Parameters:

    None

    Output Parameters:

    Networktype - Network type defined in the enum is returned

15. **IsMainContainer(fkCont: str)**

    Summary: Tests if the provided Key (TK) is the Key of the main Container of the Model

    Input Parameters:

    fkCont - the provided Key

    Output Parameters:

    isMainContainer - boolean value is returned

16. **GetMainContainer ()**

    Summary: Finds the Main Container of the Model and returns its Key (TK)

    Input Parameters:

    None

    Output Parameters:

    Tk(type: boolean) - Tk of the main container is returned

    objType(type: Enum Interfaces.Sir3SObjectTypes) - Object type is returned

17. **GetElementInfo(Tk: str)**

    Summary: Gets a short ToolTip Text for a SIR 3S Element

    Input Parameters:

    Tk – Key of the element

    Output Parameters:

    None

18. **GetNumberOfElements(ElementType:Interfaces.Sir3SObjectTypes)**

Summary: Get the Number of Elements in the Repository of a provided Type
Input Parameters:

ElementType– Type of the element

Output Parameters:

NumberofElements(type: integer) - Total number of elements of type 'ElementType'

19. **GetPropertiesofElementType(ElementType: Interfaces.Sir3SObjectTypes)**

Summary: Retrieve a List of all Propertynames defined on a SIR 3S Object Type
Input Parameters:

ElementType– Type of the element

Output Parameters:

Property_list(type: list) - list of all properties belonging to the element of type 'ElementType'

20. **GetObjectTypeof_Key(Key: str)**

Summary: Retrieves the Object Type of any random key.
Example: On a Node, we have fkFstf ="5455551997615855".
So what is the type of that fk?
It has to be "Sir3SObjectTypes.FluidThermalPropertyGroup"

Input Parameters:

Key – Key of the element

Output Parameters:

objecttype(type: Enum Interfaces.Sir3SObjectTypes) - Returns the type of object the input Key belongs to

21. **DeleteElement(Tk: str)**

Summary: Deletes the Element identified by its Key (tk/pk).
Privileges for deleting are checked.

Input Parameters:

Tk – Key of the element

Output Parameters:

None

22. **InsertElement(ElementType: Interfaces.Sir3SObjectTypes)**

    Summary: Inserts a new Object/Element of a certain Type and returns its Key (PK or TK)

    Input Parameters:

    ElementType– Type of the element

    Output Parameters:

    Tk(type: string) - Tk of the element inserted

23. **NewModel(dbName: str, providerType: Interfaces.SirDBProviderType, netType:Interfaces.NetworkType, modelDescription: str, namedInstance: str, userID: str, password: str)**

    Summary: Creates a new Model of specified Network Type, opens it and returns the Model Identifier (eg. 'M-1-0-1') of the new Model.

    Input Parameters:

    dbName(type: string) - Full Path to the Database File

    providerType(type: Enum Interfaces.SirDBProviderType) - Provider type from the enum

    netType(type: Interfaces.NetworkType) - Network type

    modelDescription(type: string) - Description of the model to be created

    namedInstance(type: string) - Instance Name of the SQL Server

    userID(type: string) - User Id for Authentication, only needed for ORACLE and for SQLServer only if SQLServer Authentication is required

    password(type: string) - Password for Authentication, only needed for ORACLE and for SQLServer only if SQLServer Authentication is required

    Output Parameters:

    Tk(type: string) - Tk of the element inserted

24. **ConnectConnectingElementWithNodes(Tk:str, keyOfNodeI:str, keyOfNodeK: str)**

Summary: Totally connects a Link (Link with 2 Connectors or Pipe) with 2 Nodes. Privileges for editing technical Data are checked. The Link and both the I- and K-Side Node must reside within the same Container (View/fkCont)

Input Parameters:

Tk(type: String) - Tk of the connecting object

keyOfNodeI(type: string) - Tk of node I(one of the element that needs to be connected)

keyOfNodeK(type: string) - Tk of node K(other element that needs to be connected)

Output Parameters:

None

25. **ConnectBypassElementWithNode(Tk: str, keyOfNodeI: str)**

Summary: Connects a Bypass-Element with a Node. Privileges for editing technical Data are checked. The Bypass and the Node must reside within the same Container (View/fkCont)

Input Parameters:

Tk(type: String) - The Key (TK) of the Bypass-Element

keyOfNodeI(type: string) - Tk of node I(one of the element that needs to be connected)

Output Parameters:

None

26. **SaveChanges()**

Summary: Saves all Changes done on the Model back into Database (or XML File)

Input Parameters:

None

Output Parameters:

None

27. **AddTableRow(tablePkTk: str)**

    Summary: Inserts a new Table Row into the Rows Collection of a provided Table and returns the PK (TK if any) of the new inserted Row and also the Element Type of that new Row

    Input Parameters:

    tablePkTk(type: string) - Key of the table

    Output Parameters:

    Tk(type: string) - Tk of the inserted row

    objectType(type: Enum Interfaces.SirDBProviderType)

28. **GetTableRows(tablePkTk: str)**

    Summary: Gets a List of all Rows of a SIR 3S Table

    Input Parameters:

    tablePkTk(type: string) - Key of the table

    Output Parameters:

    Tk_list(type: list) - List of Tk's of all rows of the table

    objectType(type: Enum Interfaces.SirDBProviderType)

29. **RefreshViews()**

    Summary: Causes the Presenter to rebuild the Element Cell List and refresh all open Views.

    Input Parameters:

    None

    Output Parameters:

    None

30. **SetInsertPoint(elementKey: str, x:np.float64, y:np.float64)**

    Summary: Set the Insert Point of a Symbol-Object (eg. Node, Valve, Tank, etc...). The Insertpoint of a Symbol-Object is the Position the on which Object is placed in the View.

    Input Parameters:

    elementKey(type: string) - Key of the symbol-object

    x(type: double) - x co-ordinate for the object

    y(type: double) - y co-ordinate for the object

    Output Parameters:

    None

31. **SetElementColor_RGB(elementKey: str, red: int, green: int, blue: int, fillOrLineColor: bool)**

    Summary: Sets the Color of an Element in RGB.

    Input Parameters:

    elementKey(type: string) - Key of the symbol-object

    red(type: integer) - The R-Part of the Color (0...255)

    green(type: integer) - The G-Part of the Color (0...255)

    blue(type: integer) - The B-Part of the Color (0...255)

    fillOrLineColor(type: boolean) - True if the Filling Color is to be set, False if only the Line Color is to be set.

    Output Parameters:

    None

32. **SetElementColor(elementKey: str, color: int, fillOrLineColor: bool)**

    Summary: Sets the Color of an Element.

    Input Parameters:

    elementKey(type: string) - Key of the symbol-object

    color(type: integer) - The RGB Integer Representation of the Color (COLORREF in GDI)

fillOrLineColor(type: boolean) - True if the Filling Color is to be set, False if only the Line Color is to be set

Output Parameters:

None

33. AlignElement(elementKey: str)

Summary: Aligns an Element, so its graphical Position (Insert Point) shall be equal to its technical Data Position (Xkor, Ykor)

Input Parameters:

elementKey(type: string) - Key of the symbol-object

Output Parameters:

None

34. GetResultValue(elementKey: str, propertyName: str)

Summary: Gets the Result Value of an element

Input Parameters:

elementKey(type: string) - Key of the symbol-object

propertyName(type: string) - The Name of the Result Property

Output Parameters:

value(type: string) - Value for the given elements property
valueType(type: string) - Type of the value returned

35. GetResultProperties_from_elementType(elementType: Interfaces.Sir3SObjectTypes, onlySelectedVectors: bool)

Summary: Get a List of all possible Result Propety Names of an Element Type

Input Parameters:

elementType(type: Interfaces.Sir3SObjectTypes) - The Element Type

onlySelectedVectors(type: boolean) - If this Parameter is set to TRUE, only the Names of selected Vector Channels for this ElementType shall be returned, otherwise all possible Results Property Names for this Element Type shall be returned.

Output Parameters:

Result_list(type: list of strings) - List of Result Propety Names of an Element Type

36. **GetResultProperties_from_elementKey(elementKey: str)**

Summary: Get a List of all Result Properties Names of an Element.
The Names of selected Vector Channels for the Type of this Element
and also all individually selected Results for this Element shall be returned.

Input Parameters:

elementKey(type: string) - The Element key

Output Parameters:

Result_list(type: list of strings) -List of all Result Propeties Names of an Element

37. **GetMinResult(elementType:Interfaces.Sir3SObjectTypes, propertyName: str)**

Summary: Gets the minimal Result Value of an Element Type and also the Key (tk/pk) of the corresponding Element

Input Parameters:

elementKey(type: string) - The Element key

propertyName(type: string) - The Name of the Result Property

Output Parameters:

MinResult(type: string) - the minimal Result Value of an Element Type
tkElement(type: string) - The Key (pk/tk) of the corresponding Element
valueType(type: string) - The Data tyte of the Result

38. **GetMaxResult(elementType:Interfaces.Sir3SObjectTypes, propertyName: str)**

Summary: Gets the maximal Result Value of an Element Type and also the Key (tk/pk) of the corresponding Element
Input Parameters:

elementKey(type: string) - The Element key

propertyName(type: string) - The Name of the Result Property

Output Parameters:

Maxsult(type: string) - the maximal Result Value of an Element Type
tkElement(type: string) - The Key (pk/tk) of the corresponding Element
valueType(type: string) - The Data tyte of the Result

39. **GetMinResult_for_timestamp(timestamp:str,elementType: Interfaces.Sir3SObjectTypes, propertyName: str)**

Summary: Gets the minimal Result Value of an Element Type and also the Key (tk/pk) of the corresponding Element at a particular Timestamp

Input Parameters:

timestamp(type: string) - the timestamp for which result is needed

elementKey(type: string) - The Element key

propertyName(type: string) - The Name of the Result Property

Output Parameters:

MinResult(type: string) - the minimal Result Value of an Element Type at a particular Timestamp

tkElement(type: string) - The Key (pk/tk) of the corresponding Element

valueType(type: string) - The Data tyte of the Result

40. **GetMaxResult_for_timestamp(timestamp:str,elementType: Interfaces.Sir3SObjectTypes, propertyName: str)**

Summary: Gets the maximal Result Value of an Element Type and also the Key (tk/pk) of the corresponding Element at a particular Timestamp

Input Parameters:

timestamp(type: string) - the timestamp for which result is needed

elementKey(type: string) - The Element key

propertyName(type: string) - The Name of the Result Property

Output Parameters:

MaxResult(type: string) - the maximal Result Value of an Element Type at a particular Timestamp

tkElement(type: string) - The Key (pk/tk) of the corresponding Element

valueType(type: string) - The Data tyte of the Result

41. **AddNewNode(tkCont: str, name: str, typ: str, x: np.float64, y: np.float64, z: np.float32, qm_PH:np.float32, symbolFactor: np.float64, description: str, idRef: str, kvr: int)**

    Summary: Comfortable Method for inserting a new Node

    Input Parameters:

    tkCont(type: string) - The TK of the Container (View) in which the new Object shall be inserted. Entering a Value of "-1" means the Main View of the Model.

    name(type: string) - Name of the new node

    typ(type. string) - type of the new node

    x(type: float64) - X Coordinate

    y(type: float64) - Y Coordinate

    z(type: float32) - Geodetic Height

    qm_PH(type: float32) - Value for Extraction/Feeding (in Case QKON) or Pressure (in Case PKON or PKQN)

    symbolfactor(type: float64) - The Symbol Factor of the new Node.

    description(type: string) - Description

    idRef(type: string) - ID in Reference System

    kvr(type: integer) - SL/RL Flag. Should be 0 (undefined), 1 (SL) or 2 (RL)

    Output Parameters:

    Tk_new(type: string) - The Key (TK) of the added Node, otherwise '-1' if something went wrong

42. **AddNewPipe(tkCont: str, tkFrom: str, tkTo: str,L:np.float32, linestring: str, material: str, dn: str, roughness: np.float32, idRef: str, description: str, kvr: int)**

    Summary: Comfortable Method for inserting a new Pipe

    Input Parameters:

    tkCont(type: string) - The TK of the Container (View) in which the new Object shall be inserted. Entering a Value of "-1" means the Main View of the Model.

    tkFrom(type: string) -Tk (Key) of the Start Node

tkTo(type. string) - Tk (Key) of the End Node

L(type: float32) - The Pipe Length, mandatory for Computation

linestring(type: string) - An optional string with intermediate Points for Geometry formatted like 'LINESTRING (120 76, 500 300, 620 480)'. The Insert Points of From and To will be added on both Ents of the Geometry

material(type: string) - Name or Tk (key) of the Pipe Diameter Table

dn(type: string) - The Nominal Diameter or the Tk of the Nominal Diameter

roughness(type: float32) - Roughness of Pipe

description(type: string) - Description

idRef(type: string) - ID in Reference System

kvr(type: integer) - SL/RL Flag. Should be 0 (undefined), 1 (SL) or 2 (RL)

Output Parameters:

Tk_new(type: string) - The Key (TK) of the added pipe, otherwise '-1' if something went wrong

43. **AddNewConnectingElement(tkCont: str, tkFrom: str, tkTo: str, x: np.float64, y: np.float64, z: np.float32, elementType: Interfaces.Sir3SObjectTypes, dn: np.float32, symbolfactor: np.float64, angleDegree:np.float32, idRef: str, description: str)**

Summary: Comfortable Method for inserting a new Connecting Element

Input Parameters:

tkCont(type: string) - The TK of the Container (View) in which the new Object shall be inserted. Entering a Value of "-1" means the Main View of the Model.

tkFrom(type: string) -Tk (Key) of the Start Node

tkTo(type. string) - Tk (Key) of the End Node

x(type: float64) - X Coordinate

y(type: float64) - Y Coordinate

z(type: float32) - Z Coordinate

elementType(type: Sir3SObjectTypes) - Element type

dn(type: string) - The Nominal Diameter or the Tk of the Nominal Diameter

symbolfactor(type: float64) - The Symbol Factor of the new Node.

angleDegree(type: float32) - The Symbol Angle in Degree

idRef(type: string) - ID in Reference System

description(type: string) – Description

Output Parameters:

Tk_new(type: string) - The Key (TK) of the added connecting element, otherwise '-1' if something went wrong

44. **AddNewBypassElement(tkCont: str, tkFrom: str, x: np.float64, y: np.float64, z: np.float32,symbolfactor: np.float64, elementType: Interfaces.Sir3SObjectTypes, idRef: str, description: str)**

Summary: Comfortable Method for inserting a new Bypass Element

Input Parameters:

tkCont(type: string) - The TK of the Container (View) in which the new Object shall be inserted. Entering a Value of "-1" means the Main View of the Model.

tkFrom(type: string) -Tk (Key) of the Start Node

x(type: float64) - X Coordinate

y(type: float64) - Y Coordinate

z(type: float32) - Z Coordinate

elementType(type: Sir3SObjectTypes) - Element type

symbolfactor(type: float64) - The Symbol Factor of the new Node.

idRef(type: string) - ID in Reference System

description(type: string) – Description

Output Parameters:

Tk_new(type: string) - The Key (TK) of the added Bypass element, otherwise '-1' if something went wrong

B. Class: SIR3S_View

1. GetMainContainer ()

Summary: Finds the Main Container of the Model and returns its Key (TK)

Input Parameters:

None

Output Parameters:

Tk(type: boolean) - Tk of the main container is returned

objType(type: Enum Interfaces.Sir3SObjectTypes) - Object type is returned

2. AddExternalPolyline(xArray: list, yArray: list, iColor: int, lineWidthMM: np.float64, dashedLine: bool, containerTK: str)

Summary: Insert a new External Polyline and returns its Key. External Objects are not persisted in the Database

Input Parameters:

xArray - Array with X-Coordinates of the Polyline

yArray - Array with Y-Coordinates of the Polyline

iColor - The Line Color of the Polyline

dashedLine - True if line is dashed

containerTK - The Key of the Container (View) to store the new Polyline into.

Output Parameters:

Tk(type: boolean) - Tk of the line added

3. AddExternalPolylinePoint(Tk: str, x: np.float64, y: np.float64)

Summary: Appends a further Point to an existing external Polyline. External Objects are not persisted in the Database

Input Parameters:

Tk – Key of the polyline

x - X-Coordnate of the new Point

y - Y-Coordnate of the new Point

Output Parameters:

None

4. **SetExternalPolyLineWidthAndColor(Tk: str, lineWidthMM: np.float64, iColor: int)**

   Summary: Sets some major Properties on an external Polyline

   Input Parameters:

   Tk - The Key of the Polyline

   widthMM - Line Width in mm

   iColor - Color of the Polyline

   Output Parameters:

   None

5. **AddExternalPolygon(xArray: list, yArray: list, lineColor: int, fillColor: int, lineWidthMM: np.float64, isFilled: bool, containerTK: str)**

   Summary: Insert a new External Polygon and returns its Key. External Objects are not persisted in the Database

   Input Parameters:

   xArray - Array with X-Coordinates of the Polygon

   yArray - Array with Y-Coordinates of the Polygon

   lineColor - The Line Color of the Polygon

   fillColor - The filling Col,or of the Polygon (if its set to be filled)

   lineWidthMM - The Line Width of the Polygon in mm

   isFilled - True if the Polygon shall be filled

   containerTK - The Key of the Container (View) to store the new Polygon into.

   Output Parameters:

   Tk(type: boolean) - Tk of the polygon added

6. **AddExternalPolygonPoint(Tk: str, x: np.float64, y: np.float64)**

   Summary: Appends a further Point to an existing external Polygon. External Objects are not persisted in the Database

   Input Parameters:

   Tk – Key of the polyline

   x - X-Coordnate of the new Point

   y - Y-Coordnate of the new Point

   Output Parameters:

   None

7. **SetExternalPolygonProperties(Tk: str, lineWidthMM: np.float64, lineColor: int, fillColor: int, isFilled: bool)**

   Summary: Sets some major Properties on an external Polygon

   Input Parameters:

   Tk - The Key of the Polygon

   widthMM - Line Width in mm

   lineColor - Line Color

   fillColor - Filling Color (if set to be filled)

   isFilled - Fill the Polygon or not

   Output Parameters:

   None

8. **AddExternalText(x: np.float64, y: np.float64, textColor: int, text: str, angleDegree: np.float32, heightPt: np.float32, isBold: bool, isItalic: bool,isUnderline: bool, containerTK: str)**

   Summary: Insert a new External Text and returns its Key. External Objects are not persisted in the Database

   Input Parameters:

   X - X-Coordinate of the Text

Y - Y-Coordinate of the Text

textColor - Text Color

strText - The TEXT (Content of the Text)

angleDegree - Text Angle in Degree

heightPt - Text Height in Points

isBold - True if bold Text

isItalic - True if italic Text

isUnderline - True if underlined Text

containerTK - The Key of the Container (View) to store the new text into.

Output Parameters:

Tk(type: boolean) - Tk of the text added

9. **SetExternalTextText(Tk: str, text: str)**

   Summary: Sets the Content of an external Text

   Input Parameters:

   Tk - The Key of the external Text</param>

   strText - The Content

   Output Parameters:

   None

10. **SetExternalTextProperties(Tk: str, x: np.float64, y: np.float64, textColor: int, text: str, angleDegree: np.float32, heightPt: np.float32, isBold: bool, isItalic: bool,isUnderline: bool)**

    Summary: Sets some major Properties on an external Text

    Input Parameters:

    Tk - The Key of the external Text

    X - X-Coordinate f the Text

    Y - Y-Coordinate f the Text

    textColor - Text Color

    strText - Content of the Text

angleDegre - Text Angle in Degree

heightPt - Text Height in Points

isBold - True for bold Text

isItalic - True for italic Text

isUnderline - True for underlined Text

Output Parameters:

None

11. **AddExternalArrow(x: np.float64, y: np.float64, lineColor: int, fillColor: int, lineWidthMM: np.float64, isFilled: bool, symbolFactor: np.float64, containerTK: str)**

    Summary: Insert a new External Arrow and returns its Key. External Objects are not persisted in the Database

    Input Parameters:

    X - X-Coordinate of the Arrow

    Y - Y-Coordinate of the Text

    lineColor - arrow Color

    fillColor - The filling Color of the Arrow (if its set to be filled)

    lineWidthMM - The Line Width of the arrow in mm

    isFilled - True if the arrow shall be filled

    symbolFactor - Symbol Factor (for scaling the Symbol)

    containerTK - The Key of the Container (View) to store the new arrow into.

    Output Parameters:

    Tk(type: boolean) - Tk of the arrow added

12. **SetExternalArrowProperties(Tk: str, x: np.float64, y: np.float64, lineColor: int, fillColor: int, lineWidthMM: np.float64, isFilled: bool, symbolFactor: np.float64)**

    Summary: Sets some major Properties on an external Arrow

    Input Parameters:

Tk - The Key of the external Arrow

X - X-Coordinate of the Arrow

Y - Y-Coordinate of the Arrow

lineColor - arrow Color

fillColor - The filling Color of the Arrow (if its set to be filled)

lineWidthMM - The Line Width of the arrow in mm

isFilled - True if the arrow shall be filled

symbolFactor - Symbol Factor (for scaling the Symbol)

Output Parameters:

None

13. **AddExternalRectangle(left: np.float64, top: np.float64, right: np.float64, bottom: np.float64,lineColor: int, fillColor: int, lineWidthMM: np.float64, isFilled: bool, isRounded: bool, containerTK: str)**

    Summary: Insert a new External rectangle and returns its Key. External Objects are not persisted in the Database

    Input Parameters:

    left - X-Coordinate of the top left

    top - Y-Coordinate of the top left

    right – X-Coordinate of the right bottom

    bottom – Y-Coordinate of the right bottom

    lineColor - rectangle Color

    fillColor - The filling Color of the rectangle (if its set to be filled)

    lineWidthMM - The Line Width of the rectangle in mm

    isFilled - True if the rectangle shall be filled

    symbolFactor - Symbol Factor (for scaling the Symbol)

    containerTK - The Key of the Container (View) to store the new rectangle into.

    Output Parameters:

    Tk(type: boolean) - Tk of the rectangle added

14. **SetExternalRectangleProperties(Tk: str, left: np.float64, top: np.float64, right: np.float64, bottom: np.float64,lineColor: int, fillColor: int, lineWidthMM: np.float64, isFilled: bool, isRounded: bool)**

    Summary: Sets some major Properties on an external rectangle

    Input Parameters:

    left - X-Coordinate of the top left

    top - Y-Coordinate of the top left

    right – X-Coordinate of the right bottom

    bottom – Y-Coordinate of the right bottom

    lineColor - rectangle Color

    fillColor - The filling Color of the rectangle (if its set to be filled)

    lineWidthMM - The Line Width of the rectangle in mm

    isFilled - True if the rectangle shall be filled

    isRounded - True if the Rectangle should have rounded Corners

    Output Parameters:

    None

15. **AddExternalEllipse(self, left: np.float64, top: np.float64, right: np.float64, bottom: np.float64,lineColor: int, fillColor: int, lineWidthMM: np.float64, isFilled: bool, containerTK: str)**

    Summary: Insert a new External ellipse and returns its Key. External Objects are not persisted in the Database

    Input Parameters:

    left - X-Coordinate of the top left

    top - Y-Coordinate of the top left

    right – X-Coordinate of the right bottom

    bottom – Y-Coordinate of the right bottom

    lineColor - ellipse Color

    fillColor - The filling Color of the ellipse (if its set to be filled)

lineWidthMM - The Line Width of the ellipse in mm

isFilled - True if the ellipse shall be filled

containerTK - The Key of the Container (View) to store the new rectangle into.

Output Parameters:

Tk(type: boolean) - Tk of the ellipse added

16. **SetExternalEllipseProperties(self, Tk: str, left: np.float64, top: np.float64, right: np.float64, bottom: np.float64,lineColor: int, fillColor: int, lineWidthMM: np.float64, isFilled: bool)**

    Summary: Sets some major Properties on an external ellipse

    Input Parameters:

    left - X-Coordinate of the top left

    top - Y-Coordinate of the top left

    right – X-Coordinate of the right bottom

    bottom – Y-Coordinate of the right bottom

    lineColor - ellipse Color

    fillColor - The filling Color of the ellipse (if its set to be filled)

    lineWidthMM - The Line Width of the ellipse in mm

    isFilled - True if the ellipse shall be filled

    Output Parameters:

    None

17. **PrepareColoration()**

    Summary: Should always be called as first each Time a Network Coloration or external Visual Objects need to be created

    Input Parameters:

    None

    Output Parameters:

    None

18. **InitColorTable(iColors: list, maxColors: int)**

    Summary: Creates a linear Gradient Color Palete from the provided Colors and the maximal Number of Colors.

    Input Parameters:

    iColors - Color List, the Gradient shall be based on

    maxColors - The maximal Number of Colors. The following Numbers are recommanded: 16, 32, 64, 128 and 256.

    Output Parameters:

    retVal(bool) – init color table is success or not

19. **GetColor(valMin: np.float64, valMax: np.float64, val: np.float64)**

    Summary: Gets a Color at a given Index from the Color Palete

    Input Parameters:

    valMin - Scale Minimum Value

    valMax - Scale Maximum Value

    val - The current Value

    Output Parameters:

    retVal(bool) – return value of GetColor method

    idx – Index of the color

20. **GetColorTableEntries(result_i: np.float64, result_k: np.float64, scaleMin: np.float64, scaleMax: np.float64)**

    Summary: Gets all Colors that fit between Results on 2 consecutive Points, given the min and max Values of the Scale. This Call should always be done after InitColorTable()

    Input Parameters:

    result_i - Result Value at a Point

    result_k - Result Value at the next neighbouring Point

    scaleMin - The Minimum Value of the Scale

scaleMax - The Maximum Value of the Scale

Output Parameters:

retVal – return table entry

21. **SetWidthScaleProperties(valMin: np.float64, widthMin: np.float64, valMax: np.float64, widthMax: np.float64)**

    Summary: Sets Criteria for Calculating Pipe Width when Coloration

    Input Parameters:

    valMin - The Result-Value corresponding to the minimum Width

    widthMin - The minimum Width

    valMax - The Result-Value corresponding to the maximum Width

    widthMax - The maximum Width

    Output Parameters:

    retVal(bool) - True, always

22. **GetWidthFactor(actualValue: np.float64)**

    Summary: Calculates the Pipe Width Scaling Factor, dase on the actual Result on Pipe and the Vaues set in SetWidthScaleProperties(). If SetWidthScaleProperties() has not been called yet, a Value of 1 will be returned, indicating that the Pipe will be drawn with its normal Width.

    Input Parameters:

    actualValue - The actual Result on Pipe

    Output Parameters:

    retVal(float64) - The Pipe Width Scaling Factor

23. **ColoratePipe(Tk: str, lengths: list, Colors: list, widthFactors: list)**

    Summary: Method for colouring a Pipe with a single constant Color

    Input Parameters:

    Tk - The Tk of the Pipe

Color - The Constant Color for Coloration

widthFactors - Array of Factors for Width Scaling (1 Factor per Segment). Should have the Length of 'lengths' minus 1!

Output Parameters:

None

24. ResetColoration()

Summary: Resets the Coloration, so the UI shall draw Pipes as normally configured

Input Parameters:

None

Output Parameters:

None

25. DoColoration ()

Summary: Forces the UI to perform Pipe Coloration

Input Parameters:

None

Output Parameters:

None

26. StartTransaction(SessionName:str)

Summary: If Modifications on a Model are intended, it is recommended to make a Call of StartTransaction(), then do all the Modifications you need. And then call EndTransaction() as soon as you are finished with Modifications. This helps the Software to keep Track of Modifications, so the User can Undo/Redo them on the main UI (SirGraf)

Input Parameters:
SessionName(string) - give a meaningful name to start a transaction; Empty string or None will lead to error

Output Parameters:
None

27. **EndTransaction()**

Summary: If Modifications on a Model are intended, it is recommended to make a Call of StartTransaction(), then do all the Modifications you need. Ant then call EndTransaction() as soon as you are finished with Modifications. This helps the Software to keep Track of Modifications, so the User can Undo/Redo them on the main UI (SirGraf)

Input Parameters:

None

Output Parameters:

None

28. **StartEditSession(SessionName:str)**

Summary: Recommended for fast bulk Changes (e.g. Changing the Values of 40 thousand Nodes in a single Task). Similar to StartTransaction(), EndEditSession() should be called after the Caller is done with all his bulk Changes.

Input Parameters:

SessionName(string) - give a meaningful name to start a transaction; Empty string or None will lead to error

Output Parameters:

None

29. **EndEditSession()**

Summary: Closes an already started EditSession. Should always be called after a Call of StartEditSession() and all the Modifications applied.

Input Parameters:

None

Output Parameters:

None

30. **SaveChanges()**

Summary: Saves all Changes done on the Model back into Database (or XML File)

Input Parameters:

None

Output Parameters:

None