NEPLAN Calculation DLL

Table of Contents

How to write an application in VisualStudio with the NEPLAN Calculation DLL

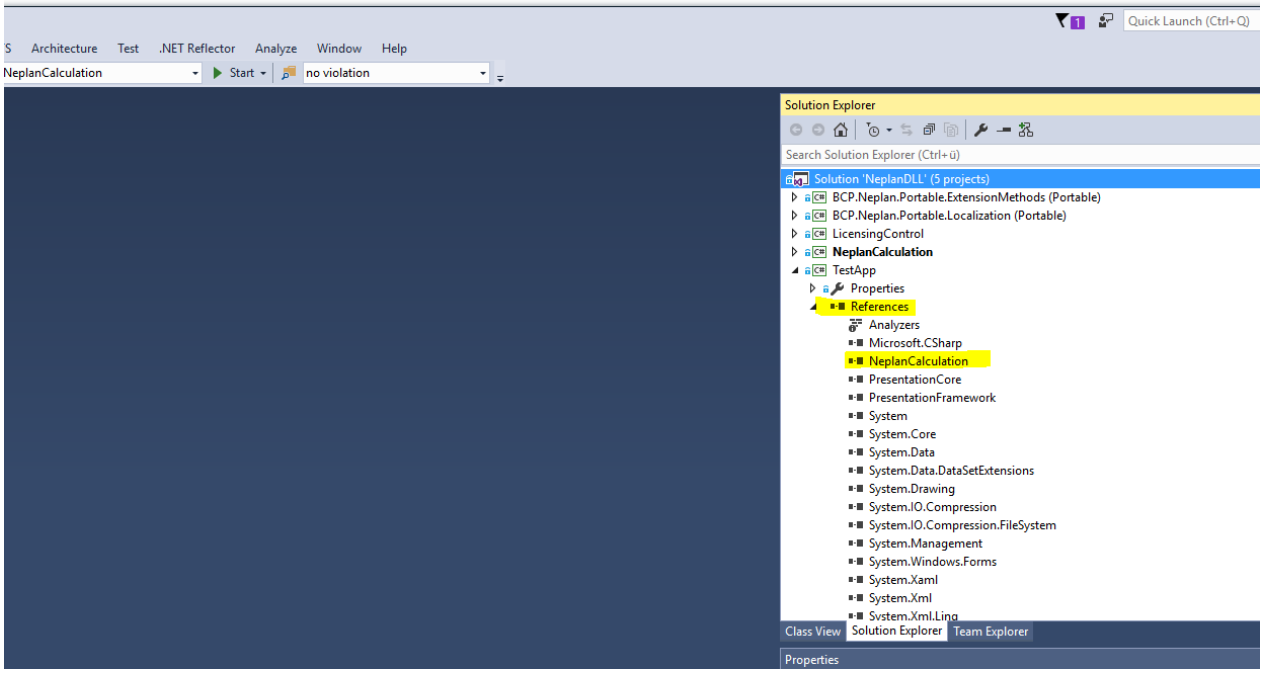
Please note:
There is no need to have a NEPLAN 360 version installed to use the NEPLAN calculation library (C# API library). Everything you need to integrate the NEPLAN algorithm in your software is already included in the API, e.g. import from various sources (xml, PSSE, CIM), calculations/simulations and setting of any input data and retrieving any results.

To be able to make your own application with access to the NEPLAN-DLL you need to have the following files:

- A file “NeplanCalculation.dll”.
- The file license.txt which was generated for you by NEPLAN based on the machine code of your computer.
- Following DLLs: ExtDynamicModels.dll, NeplanSolverKDLL.dll, NeplanSolverNDLL.dll and NeplanSolverUDLL.dll.
- The configuration file NLog.config.

Copy these files in a temporary folder.
Now create a Visual Studio project in the following way:

- Create a new project.
- In the solution tree right-click on “Reference”->”Add Reference”, then browse to “NeplanCalculation.dll”
- Right-Click your project and choose “Add…”->”Existing Item”. Choose the file “license.txt”
- Then right click “license.txt” and go to properties. Go to “Copy to Output Directory” and select “Copy if newer”. Go to “Build Action” and select “Content”
- Do the same as you have done for “license.txt” also for ExtDynamicModels.dll, NeplanSolverKDLL.dll, NeplanSolverNDLL.dll, NeplanSolverUDLL.dll, NLog.config
- Right click on the project and choose “Properties”. In the pane “Build” uncheck the option “prefer 32-bit” and select “Any CPU” as Configuration Platform



Available functions in the NEPLAN-DLL

Log functions:

- Add a listener for getting output information from the DLL
  - `public static void AddTraceListener(TextWriterTraceListener textWriteTraceList)`
- Add a listener for getting output information from the DLL
  - `public static void AddTraceListener(TraceListener traceList)`
- Clear all the added listener
  - `public static void ClearTraceListener()`

Initialization functions:

- Initialize the NeplanInterface object. If the object is null, then there were some problems with the license.
  The NeplanInterface object is used for calling all the functions of the DLL.
  - `public static NeplanInterface CreateInterface()`

Path functions:

- Defines the path to all Neplan libraries.
  - `public static void SetNeplanLibrariesPath(string path)`

License functions:

- Defines the Path to the license.txt
  - `public static string SetLicensePath(string path)`
- Get the code of the actual machine the program is running on. It is used to create a valid license.
  - `public static string getMachineCode()`
- Get the expiring date of the license.
  - `public DateTime GetLicenseExpiringDate()`
- Get the info of the actual machine the program is running on.
  - `public static string getMachineInfo()`
- Get list of enabled modules for the used license.
  - `public IEnumerable<string> getModulesEnabled()`
- Get the product version for the used license.
  - `public string getLicenseVersion()`

Project Import functions:

- Imports a list file. (".xml" or a ".neplst360") and returns the ImportProjectContainer Object
  - `public ImportProjectContainer GetProjectContainerFromFile(Stream stream)`
- Imports a list file. (".xml" or a ".neplst360")

- `public ImportReturnInfo ImportXMLFile(Stream stream)`

- Imports directly the ImportProjectContainer object.
  - `public ImportReturnInfo ImportImportProjectContainer (ImportProjectContainer importContainer)`
- Creates a project from the CDE files. All files should be added in a ZIP.
  - `public ImportReturnInfo ImportCDEAsZip(Stream stream, double nodeFrequency)`
- Import CIM Project. All the CIM files should be saved in a ZIP. A boundary file (ZIP) can be imported separately

  `public ImportReturnInfo ImportCim(Stream mainStream, Stream boundaryStream = null)`
- Imports a NEPLAN360 project to the DLL.
  - `public ImportReturnInfo ImportNep360Project(Stream stream)`
- Imports a NEPLAN360 project to the DLL.
  - `public ImportReturnInfo ImportNep360Project(string filePath)`
- Creates a project from PSSE files. Version number of the PSSE files must be added to the import function.
  - `public ImportReturnInfo ImportPSSEAsZIP(Stream stream, int version)`
  - `public ImportReturnInfo ImportPSSEAsZIP(string filepath, int version)`
- Creates a project from the UCTE.
  - `public ImportReturnInfo ImportUCTE(Stream stream)`

Update Import functions:

- Updates an existing project from an NDB file.
  - `public ImportReturnInfo ImportNDBFile(Stream stream)`
- Updates an existing project from the ZDB.
  - `public ImportReturnInfo ImportZDBFile(Stream stream)`
- Updates an existing project with CIM ids
  - `public bool ImportCIMIds(Stream stream)`

Export Project functions:

- Exports the project to a NEPLAN360 file.
  - `public bool ExportProject(string projectName, string description, Stream stream)`
- Exports the project to a UCT file.
  - `public bool ExportProjectAsUCTE(Stream outputStream)`
- Exports the project in CIM format. All files will be exported.
  - `public bool ExportProjectASCim(Stream outputStream, string boundaryPath, IEnumerable<Group_Area> areasToExport = null, DateTime? scenarioDateTime = null, string version = "1", string period = "1D")`

- Exports the project in CIM format.
  - `public bool ExportProjectASCim(Stream outputStream, bool exportEQ, bool exportSSH, bool exportTP, bool exportMerged, bool exportDY, bool exportDL, bool exportGL, bool exportSV, bool eNTSOEZIP, string boundaryPath, IEnumerable<Group_Area> areasToExport = null, DateTime? scenarioDateTime = null, string version = "1", string period = "1D")`

CIM ids functions:

- Updates an existing project with CIM ids
  - `public bool ImportCIMIds(Stream stream)`
- Exports only the CIM ids of the project
  - `public bool ExportCIMIds(Stream stream)`

Library import functions:

- Depending on the given file. The correct library import will be started
  - `public ImportReturnInfo ImportLibrary(Stream stream, string fileName)`
- Imports a NEPLAN360 library file
  - `public ImportReturnInfo ImportNeplanLib(Stream stream)`
- Imports a txt file based library
  - `public ImportReturnInfo ImportTxtLibrary(Stream reader)`
- Imports a xml file based library
  - `public ImportReturnInfo ImportXMLLibrary(Stream reader)`

Calculation functions:

- Runs a Contingency analysis calculation.
  - `public AnalysisReturnInfo RunContingencyAnalysis()`
- Runs a Heating analysis calculation.
  - `public AnalysisReturnInfo RunDistrictHeating()`
- Runs a Dynamic simulation. If the calcParamXML is empty the default parameter settings are used
  - `public AnalysisReturnInfo RunDynamicAnalysis()`
- Runs a Gas analysis calculation.
  - `public AnalysisReturnInfo RunGasAnalysis()`
- Runs a LoadFlow analysis calculation.
  - `public AnalysisReturnInfo RunLoadFlow()`
- Runs a PTDF Analysis calculation. The transactions and the branches are input to the method
  - `public AnalysisReturnInfo RunPTDF(IEnumerable< Transaction> transactions, IEnumerable<Branch> branches)`

- Runs a ShortCircuit calculation. Add the fault location elements with the faultedElements list. If the list is null or empty, it will consider the default fault locations.
  - `public AnalysisReturnInfo RunShortCircuit(IEnumerable<string> faultedElements = null)`
- Runs a DACH calculation on the given customer connection.
  - `public AnalysisReturnInfo RunShortCircuitDACH(CustomerConnection element)`

Result functions:

- Get the results of the last calculation as an XML string.
  - `public string GetAnalysisResult(string calcType, bool showOnlySummary = false)`
- Save the last results in a file.
  - `public void GetAnalysisResult(Stream fileStream, string calcType, bool showOnlySummary = false)`
- Get the results of all elements.
  - `public IEnumerable<ElementBaseResult> GetElementResults(string calcType)`
- Get all results of an element.
  - `public IEnumerable<ElementBaseResult> GetResultsOfElement(string calcType, Guid elementID)`
- Get the result associated to a port of an element.
  - `public ElementBaseResult GetResultOfElement(string calcType, Guid elementID, int portNr)`
  - `public ElementBaseResult GetResultOfElement(string calcType, string elementName, int portNr)`
- Get the value of a member of a result of an element.
  - `public bool GetResultValueOfElement<T>(string calcType, Guid elementID, string variablePath, int portNr, out T result)`
  - `public bool GetResultValueOfElement<T>(string calcType, string elementName, string variablePath, int portNr, out T result)`
- Get the value of a member of a result of a node.
  - `public bool GetResultValueOfNode<T>(string calcType, Guid elementID, string variablePath, out T result)`
- Get the result associated to a gas/water/district heating element.
  - `public ElementBaseResult GetResultOfGWHElement(string calcType, Guid elementID)`
- Get the value of a member of a result of a gas/water/district heating element.
  - `public bool GetResultValueOfGWHElement<T>(string calcType, Guid elementID, string variablePath, out T result)`
- Get Sensitivities result .
  - `public SensitivitiesResultSummary GetSensitivitiesResult()`

- Get summary results for the last calculation.
  - `public IEnumerable<SummaryBaseResult> GetSummaryResults(string calcType)`
- Defines if the last calculation has converged.
  - `public bool HasConverged(string calcType)`

Calculation parameter functions:

- Get parameters for calculation.
  - `public T GetCalcParameter<T>()`
  - `public CalcParameter GetCalcParameter(string calcType)`
- Set parameters for calculation.
  - `public void SetCalcParameter(CalcParameter calcParameter)`

Network handling functions:

- Create a new project.
  - `public bool CreateProject(string projectName, ProjectMediumType projectMediumType, string description = null)`
- Get all areas of the project.
  - `public IEnumerable<Group_Area> GetAreas()`
- Get area.
  - `public Group_Area GetAreaByID(Guid id)`
  - `public Group_Area GetAreaByName(string name)`
- Get area by sub area id.
  - `public Group_Area GetAreaBySubAreaID(Guid id)`
- Add new area.
  - `public Group_Area AddNewArea(string areaName)`
- Delete area.
  - `public bool DeleteArea(Guid areaID)`
- Get sub area.
  - `public Group_SubArea GetSubAreaByID(Guid subAreaID)`
  - `public Group_SubArea GetSubAreaByName(string subAreaName, string areaName = null)`
- Get sub areas from area.
  - `public IEnumerable<Group_SubArea> GetSubAreasOfArea(string areaName)`
  - `public IEnumerable<Group_SubArea> GetSubAreasOfArea(Guid areaID)`
- Add new sub area.
  - `public Group_SubArea AddNewSubArea(string subAreaName, Guid areaID)`
- Delete sub area.
  - `public bool DeleteSubArea(Guid subAreaID)`
- Get all zones of the project
  - `public IEnumerable<Group_Zone> GetZones()`

- Get zone.
  - `public Group_Zone GetZoneByID(Guid id)`
  - `public Group_Zone GetZoneByName(string name)`
- Add new zone.
  - `public Group_Zone AddNewZone(string zoneName)`
- Delete zone.
  - `public bool DeleteZone(Guid zoneID)`
- Get zone data.
  - `public ZoneData GetZoneData(Guid zoneID)`
- Get all feeders of the project.
  - `public IEnumerable<Group_Feeder> GetFeeders()`
- Get feeder.
  - `public Group_Feeder GetFeederByID(Guid id)`
  - `public Group_Feeder GetFeederByName(string name)`
- Get feeder of an element.
  - `public Group_Feeder GetFeederByElementID(Guid elementID)`
- Add new feeder.
  - `public Group_Feeder AddNewFeeder(string feederName, Guid nodeID, Guid elementID)`
- Get feeder data.
  - `public FeederData GetFeederData(Guid feederID)`
- Delete feeder.
  - `public bool DeleteFeeder(Guid feederID)`
  - `public bool DeleteFeeder(string feederName)`

Voltage level functions:

- Get all voltage levels.
  - `public IEnumerable<VoltageLevelData> GetVoltageLevels()`
- Get voltage level.
  - `public VoltageLevelData GetVoltageLevelByUn(double un)`
- Add and return a new voltage level.
  - `public VoltageLevelData AddNewVoltageLevel(double un)`
- Delete a voltage level.
  - `public bool DeleteVoltageLevel(double un)`

Pressure level functions:

- Get all pressure levels.
  - `public IEnumerable<PressureLevelData> GetPressureLevels()`
- Get pressure level.
  - `public PressureLevelData GetPressureLevelByPn(double pn)`
- Add and return a new pressure level.

- `public PressureLevelData AddNewPressureLevel(double pn)`

- Delete a pressure level.
  - `public bool DeletePressureLevel(double pn)`

Element functions:

- Get all elements of the project.
  - `public IEnumerable<TechElement> GetElements()`
- Get elements by type.
  - `public IEnumerable<T> GetElementsByType<T>()`
  - `public IEnumerable<TechElement> GetElementsByType(params string[] typeNames)`
- Get element.
  - `public TechElement GetElementByName(string name)`
  - `public TechElement GetElementByID(Guid id)`
- Get element id by name
  - `public Guid? GetElementIDByName(string name)`
- Get elements of zone.
  - `public IEnumerable<TechElement> GetElementsByZone(Guid zoneID)`
- Get elements of area
  - `public IEnumerable<TechElement> GetElementsByArea(Guid areaID)`
- Get elements of sub area.
  - `public IEnumerable<TechElement> GetElementsBySubArea(Guid subAreaID)`
- Get connected elements.
  - `public IEnumerable<TechElement> GetConnectedElements(Guid elementID)`
- Get the connected node to the element
  - `public TechElement GetNodeFromElement(Guid elementID, int portNr)`
- Add node.
  - `public bool AddNode(TechElement node, string name, Guid zoneID, Guid subAreaID)`
- Add element.
  - `public bool AddElement(TechElement element, string name, Guid zoneID, Guid subAreaID, short? phase = null, TechElement node1 = null, TechElement node2 = null, TechElement node3 = null, TechElement node4 = null)`
- Create element from library.
  - `public TechElement CreateElementFromLibrary(string libraryName, string librarySubName, string libraryItemName, TechElementTypes type)`
- Delete element.
  - `public bool DeleteElement(Guid elementID)`
- Change element zone.

- `public bool ChangeElementZone(Guid elementID, Group_Zone newZone)`

- Change element sub area.
  - `public bool ChangeElementSubArea(Guid elementID, Group_SubArea newSubArea)`
- Change a member of an element.
  - `public bool SetElementParameter<T>(Guid elementID, string memberName, T value)`
- Get a member value for an element.
  - `public bool GetElementParameter<T>(Guid elementID, string parameterName, out T result)`
- Get TechElementCalcSetting for an element.
  - `public TechElementCalcSetting GetCalcSettingOfElement(Guid elementID)`

Switch functions:

- Open or close a connection of the given element on the given port
  - `public bool OpenConnection(Guid elementID, short portNumber)`
  - `public bool CloseConnection(Guid elementID, short portNumber)`
- Open or close all connections of the given element
  - `public bool OpenAllConnectionsOfElement(string elementName)`
  - `public bool OpenAllConnectionsOfElement(Guid elementID)`
  - `public bool CloseAllConnectionsOfElement(Guid elementID)`
  - `public bool CloseAllConnectionsOfElement(string elementName)`
- Determines if the switch of the given element on the given port is connected
  - `public bool IsSwitchConnectedOfElement(string elementName, short portNumber)`
  - `public bool IsSwitchConnectedOfElement(Guid elementID, short portNumber)`

Connection functions:

- Determines if two elements are connected.
  - `public bool AreConnected(Guid element1ID, Guid element2ID)`
- Get the id of the connection between two elements.
  - `public Guid? GetConnectionID(Guid element1ID, Guid element2ID)`
- Add connection between an element and a node.
  - `public bool AddConnection(Guid elementID,int elementPortNumber, Guid nodeID)`
- Delete connection between two elements.
  - `public bool DeleteConnection(Guid element1ID, Guid element2ID)`

Container functions:

- Get all containers.

- `public IEnumerable<ElementContainer> GetContainers()`

- Get container by id.
  - `public ElementContainer GetContainer(Guid containerID)`
- Get container by name.
  - `public ElementContainer GetContainer(string containerName)`
- Get elements in the container, if `includeSubContainers is true, elements in sub containers are returned.`
  - `public IEnumerable<TechElement> GetElementsOfContainer(Guid containerID, bool includeSubContainers = false)`
- Get elements in the second container, if `includeSubContainers is true, elements in sub containers are returned.`
  - `public IEnumerable<TechElement> GetElementsOfSecondContainer(Guid containerID, bool includeSubContainers = false)`
- Get sub containers.
  - `public IEnumerable<ElementContainer> GetSubContainers(Guid containerID)`
- Add and return a new container.
  - `public ElementContainer AddNewContainer(string name, ElementContainerType type)`
- Add and return a new sub container.
  - `public ElementContainer AddSubContainer(string name, ElementContainerType type, Guid containerID)`
- Delete container.
  - `public bool DeleteContainer(Guid containerID)`
- Assign an element to a container.
  - `public bool AssignElementToContainer(Guid elementID, Guid containerID)`
- Assign an element to a second container.
  - `public bool AssignElementToSecondContainer(Guid elementID, Guid containerID)`
- Remove an element from its container.
  - `public bool RemoveElementFromContainer(Guid elementID)`
- Remove an element from its second container.
  - `public bool RemoveElementFromSecondContainer(Guid elementID)`

Diagrams functions:

- Get all diagrams of the project.
  - `public IEnumerable<Graphic_Diagrams> GetDiagrams()`
- Get all diagrams of the project.
- Get diagram.
  - `public Graphic_Diagrams GetDiagramByID(Guid diagramID)`

  - `public Graphic_Diagrams GetDiagramByName(string name)`
- Determines if the diagram contains graphics.
  - `public bool HasDiagramGraphics(Guid diagramID)`
- Determines if the diagram has layers.
  - `public bool HasDiagramLayers(Guid diagramID)`
- Add and return a new diagram.
  - `public Graphic_Diagrams AddNewDiagram(string name)`
- Delete diagram.
  - `public bool DeleteDiagram(string name)`

Layer functions:

- Get all layers. If diagramID is not null it returns only the layer of the specified diagram.
  - `public IEnumerable<Graphic_Layers> GetLayers(Guid? diagramID = null)`
- Get layer.
  - `public Graphic_Layers GetLayerByID(Guid layerID)`
  - `public Graphic_Layers GetLayerByName(string name, Guid diagramID)`
- Determines if the layer contains graphics.
  - `public bool HasLayerGraphics(Guid layerID)`
- Add new layer.
  - `public Graphic_Layers AddNewLayer(string name, Guid diagramID)`
- Delete layer.
  - `public bool DeleteLayer(Guid layerID)`

Graphic functions:

- Get all graphics. If `layerID` is not null it returns only the graphics of the specified layer.
  - `public IEnumerable<Graphic_Data> GetGraphics(Guid? layerID = null)`
- Get graphic.
  - `public Graphic_Data GetGraphicByID(Guid graphicID)`
- Get graphics of element.
  - `public IEnumerable<Graphic_Data> GetGraphicsOfElement(Guid elementID, Guid? layerID = null)`
- Get graphics of connection.
  - `public IEnumerable<Graphic_Data> GetGraphicsOfLink(Guid element1ID, Guid element2ID, Guid? layerID = null)`
- Determines if an element has graphics.
  - `public bool HasGraphics(Guid elementID)`
- Get the number of graphics of an element.
  - `public int GetNumberOfGraphics(Guid elementID)`
- Add and return a new graphic for an element.
  - `public Graphic_Data AddNewGraphicToElement(Guid elementID, Guid layerID)`

- Add and return a new graphic for a line.
  - `public Graphic_Data AddNewGraphicToLine(Guid lineID, Guid graphic1ID, Guid graphic2ID)`
- Add and return a new graphic for a connection.
  - `public Graphic_Data AddNewGraphicToLink(Guid fromGraphicID, Guid toGraphicID)`
- Add and return a new graphic for a busbar.
  - `public Graphic_Data AddNewGraphicToBusbar(Guid nodeID, Guid layerID)`
- Delete graphic.
  - `public bool DeleteGraphic(Guid graphicID)`

CIM data functions:

- Get all CIM data.
  - `public IEnumerable<CimData> GetCIMData()`
- Add a new CIM data to the project.
  - `public bool AddCIMData(CimData data)`
- Delete a CIM data.
  - `public bool DeleteCIMData(Guid cimDataID)`

Return objects

- ImportReturnInfo: The ImportReturnInfo object describes if the Import was successful (`ReturnValue is 0`). If not, then there is a list of error messages which can help to identify the problem.
- AnalysisReturnInfo: Returns the calculation information. If the ReturnInfo is 1 then the calculation was successful. The member LogList returns all the calculation messages (Warning, Errors, Info…).