There is a possibility to develop own external modules (called solvers) in the Dyssol (‘Solvers development.pdf’). A set of functions and variables depends on the type of the solver. But since all modules are inherited from the CExternalSolver class, it determines their basic functionality.

Interfaces of external solvers:

Internal variables
m_solverName m_authorName m_solverUniqueKey m_solverVersion
Constructors
CExternalSolver
Functions to work with basic info
GetName GetUniqueID GetAuthorName GetVersion
Virtual functions which can be overriden in inherited classes
Initialize Finalize

Internal variables

m_solverName
Name of the solver that will be displayed in user interface of Dyssol.

m_authorName
Solver’s author

m_solverUniqueKey
Unique identificator of the solver. Simulation environment distinguishes different solvers with the help of this identificator. You must ensure that ID of your solver is unique. This ID can be created manually or using GUID-generator of Visual Studio (Tools->GUID Genarator).

m_solverVersion
Current version of the solver.

Constructors

CExternalSolver *(void)*
Basic constructor of the solver. Creates an empty external solver. Called only once when solver is added to the unit.

Functions to work with basic info

std::string GetName *()*
Returns name of the solver.

std::string GetUniqueID *()*
Returns string key, unique among all solvers.

std::string GetAuthorName *()*
Returns name of the solver's author.

unsigned GetVersion *()*
Returns version of the solver.

Virtual functions which can be overriden in inherited classes

virtual void Initialize *()*
Solver‘s initialization. This function is called only once for each simulation during the initialization of unit. Implementation of this function is not obligatory and can be skipped.

virtual void Finalize *()*
Unit‘s finalization. This function is called only once for each simulation during the finalization of unti. Implementation of this function is not obligatory and can be skipped.