# class Grid

**Public:**

Functions:

* *Grid*()
* *Grid*(string catalogFilepath, string componentsListFilepath)
* ~*Grid*()
* gridBus\* *newBus*()
* loadNode\* *newLoad*()
* sourceNode\* *newSource*()
* filterNode\* *newFilter*()
* esmNode\* *newESM*()
* transformerNode\* *newTransformer()*
* converterNode *newConverter()*
* gridLine\* *newLine()*
* void *addNewCatalog*()
* void *addNewComponent*()
* int *numBuses()*;
* int *numLoads()*;
* int *numSources()*;
* int *numFilters()*;
* int *numESMs()*;
* int *numLines()*;
* int *numTransformers();*
* int *numConverters();*
* void *addCatalogEntry(*gridNode\**);*
* void *deleteCatalogEntry(*gridNode\**);*
* customNodeTree\* catalog;
* customNodeTree\* componentsList;

Variables:

* customNodeTree\* *catalog;*
* customNodeTree\* *componentsList;*

Structs:

* struct *busListElement*
  + gridBus\* *bus*
  + double *busVoltage*

**Private:**

Functions:

* int *findNumCatalogEntries* (string catalogFilepath)
* int *findNumComponents* (string componentsListFilepath)
* Qstring *newName(Qstring type)*

Variables:

* vector<busListElement\*>\* *busList*
* vector<loadNode\*>\* *loads*
* vector<sourceNode\*>\* *sources*
* vector<filterNode\*> *filters*
* vector<esmNode\*>\* *ESMs*
* vector<gridLine\*>\* *lines*;
* vector<transformerNode\*> *transformers*;
* vector<converterNodes\*> *converters*;
* vector<component\*>\* *activeComponents*
* commonModeGrid\* *cmEqModel*

# class gridNode

**Public:**

Functions:

* *gridNode*()
* ~*gridNode*()
* *gridNode(const gridNode &original);*
* bool *validityCheck*()
* string *getType()*
* void *setType(*std::string newType*)*
* string *getName ()*
* void *setName(*std::string newName*)*
* void *setCataog(*bool set);
* const bool& *checkCatalog()* const*;*

**Private:**

Functions:

* void *loadFromDataFile*(string filepath)

Variables:

* int *uniqueID*
* *Qstring dbName*
* Qstring *name*
* Qstring *type*
* Bool *catalog*

## class gridLine::gridNode

**Public:**

Functions:

* *gridLine*()
* ~*gridLine*()
* *gridLine(const gridLine &original);*
* bool *validityCheck*()

**Private:**

Functions:

* void *loadFromDataFile*(string filepath)

Variables:

* double *voltageA*
* double *breakerRating*
* int *numPhases*
* double *conduit\_Diameter*
* double *conduit\_Capacitance*
* double *conduit\_Resistance*
* double *conduit\_Inductance*

## class gridElement::gridNode

**Public:**

Functions:

* *gridElement*()
* ~*gridElement*()
* *gridElement(const gridElement &original);*
* bool *validityCheck*()

**Private:**

Functions:

Variables:

### class sourceNode::gridElement

**Public:**

Functions:

* *sourceNode()*
* *~sourceNode()*
* *sourceNode(const sourceNode &original);*
* bool *validityCheck()*

**Private:**

Functions:

Variables:

* double *ratedPower*
* double

### class loadNode::gridElement

**Public:**

Functions:

* *loadNode()*
* *~loadNode()*
* *loadNode(const loadNode &original);*
* bool *validityCheck()*

**Private:**

Functions:

* void loadFromDataFile(string filepath)

Variables:

* bool powerType
* double voltage
* double constPowerDemand
* int numPhases
* bool phaseA
* bool phaseB
* bool phaseC
* int profileType
* int transientMatrixSize;
* std::vector<std::vector<transientElement\*>>\*transientMatrix

Structs:

* transientElement
  + int type

### class esmNode::gridElement

**Public:**

Functions:

* *esmNode()*
* *~esmNode()*
* *esmNode(const esmNode &original);*
* bool *validityCheck()*

**Private:**

Functions:

* void loadFromDataFile(string filepath)

Variables:

### class filterNode::gridElement

**Public:**

Functions:

* *filterNode()*
* *~filterNode()*
* *filterNode(const filterNode &original);*
* bool *validityCheck()*

**Private:**

Functions:

Variables:

### class transformerNode::gridElement

**Public:**

Functions:

* *transformerNode()*
* *~ transformerNode ()*
* *transformerNode (const transformerNode &original);*
* bool *validityCheck()*

**Private:**

Functions:

* void *loadFromDataFile*(string filepath)

Variables:

* bool *transformer\_3pTypeA*
* bool *transformer\_3pTypeB*
* double *transformer\_Capacitance*
* double *transformer\_Resistance*
* double *transformer\_Inductance*
* double *transformer\_windingRatio*

### class converterNode::gridElement

**Public:**

Functions:

* *converterNode()*
* *~ converterNode ()*
* *converterNode (const converterNode &original);*
* bool *validityCheck()*

**Private:**

Functions:

* void *loadFromDataFile*(string filepath)

Variables:

* bool *pwrTypeA*
* bool *pwrTypeB*
* double *converter\_Capacitance*
* double *converter\_Resistance*
* double *converter\_Inductance*

## class gridBus::gridNode

**Public:**

Functions:

* *gridBus()*
* *~gridBus()*
* *gridBus(const gridBus &original);*
* bool *validityCheck()*
* double *getVoltage()*

**Private:**

Functions:

* void *loadFromDataFile*(string filepath)

Variables:

* int *numBreakers*
* double *bus\_Capacitance*
* double *bus\_Resistance*
* double *bus\_Inductance*
* double *­voltage*
* vector<loadNode\*> *loads*
* vector<sourceNode\*> *gensets*
* vector<gridBus\*> *childSWBDs*
* vector<gridBus\*> *equalSWBDs*
* vector<gridBus\*> *parentSWBDs*
* vector<filterNode\*> *filters*
* vector<esmNode\*> *ESMs*

# class component::Qlabel

This class is used for vsualizing the grid in the drag and drop viewport. Each instance contains a reference to a node within the grid

**Public:**

Functions:

* *Component()*
* *Component(gridNode\* inputNodeRef)*
* *~Component()*
* Std::string *checkNodeType*()
* gridNode *getNodeData*()
* void *updateThumbnail()*
* void *updateName()*

**Private:**

Functions:

Variables:

* string *name*
* gridNode\* *nodeRef*
* Qpixmap *thumbnail*

# class commonModeGrid

**Public:**

Functions:

* *commonModeGrid*()

**Private:**

Functions:

Variables:

# class customNodeTree : public QAbstractItemModel

**Public:**

Functions:

* *customNodeTree(const QStringList &headers, QObject \*parent = nullptr);*
* *~customNodeTree();*
* *QVariant data(const QModelIndex &index, int role) const override;*
* *gridNode\* getNodePtr(const QModelIndex &index);*
* *void setNodeData(const QModelIndex &index, gridNode\* node);*
* *QVariant headerData(int section, Qt::Orientation orientation, int role = Qt::DisplayRole) const override;*
* *QModelIndex index(int row, int column, const QModelIndex &parent = QModelIndex()) const override;*
* *QModelIndex parent(const QModelIndex &index) const override;*
* *int rowCount(const QModelIndex &parent = QModelIndex()) const override;*
* *int columnCount(const QModelIndex &parent = QModelIndex()) const override;*
* *Qt::ItemFlags flags(const QModelIndex &index) const override;*
* *bool setData(const QModelIndex &index, const QVariant &value, int role = Qt::EditRole) override;*
* *bool setHeaderData(int section, Qt::Orientation orientation, const QVariant &value, int role = Qt::EditRole) override;*
* *bool insertColumns(int position, int columns, const QModelIndex &parent = QModelIndex()) override;*
* *bool removeColumns(int position, int columns, const QModelIndex &parent = QModelIndex()) override;*
* *bool insertRows(int position, int rows, const QModelIndex &parent = QModelIndex()) override;*
* *bool removeRows(int position, int rows, const QModelIndex &parent = QModelIndex()) override;*
* *bool setCatalogLabel(const QModelIndex &index, QString labelName, int role);*
* *bool setFullData(const QModelIndex &index, const QList<QVariant> &data, int role = Qt::EditRole);*
* *bool checkLabel(const QModelIndex &index);*
* *QString getType(const QModelIndex &index); // Col 1*
* *QString getName(const QModelIndex &index); // Col 0*
* *customTreeItem\* getRoot();*
* *bool isRoot(const QModelIndex &index);*
* *void setExtDrag(bool draggable);*
* *bool checkExtDrag();*

**Private:**

Functions:

* *void setupModelData(const QStringList &lines, customTreeItem \*parent);*
* *customTreeItem \*getItem(const QModelIndex &index) const;*
* *QMimeData\* mimeData(const QModelIndex &index) const;*

Variables:

* customTreeItem \*rootItem;
* bool externalDragEnabled;

# class customTreeItem

**Public:**

Functions:

* *customTreeItem(const QList<QVariant> &data, customTreeItem\* parent = nullptr);*
* *~customTreeItem();*
* *customTreeItem \*child(int number);*
* *int childCount() const;*
* *int columnCount() const;*
* *QVariant data(int column) const;*
* *bool insertChildren(int position, int count, int columns);*
* *bool insertColumns(int position, int columns);*
* *customTreeItem \*parent();*
* *bool removeChildren(int position, int count);*
* *bool removeColumns(int position, int columns);*
* *int childNumber() const;*
* *bool setData(int column, const QVariant &value);*
* *gridNode\* returnRefNode();*
* *void setRefNode(gridNode\* newNode);*
* *void setLabel(bool labelVal);*
* *bool checkLabel();*
* *friend QDataStream &operator<<(QDataStream &out, customTreeItem ptr);*
* *friend QDataStream &operator>>(QDataStream &in, customTreeItem ptr);*

**Private:**

Functions:

Variables:

* QList<customTreeItem\*> childItems;
* QList<QVariant> itemData;
* customTreeItem\* parentItem;
* gridNode\* refNode;
* bool label = true;

# class MainWindow

**Public:**

Functions:

* *MainWindow(QWidget \*parent = nullptr);*
* *~MainWindow();*
* *void insertCatalogEntry\_shorePower();*

Variables:

* *Grid myGrid;*

**Public Slots:**

* *void updateActions\_catalogNetwork();*
* *void updateActions\_catalogConst();*
* *void updateActions\_compList();*

**Private Slots:**

* *void on\_pushButton\_clicked();*
* void insertCatalogEntry();
* QModelIndex insertCatalogLabel(QString name = "[Edit Database Name]", bool initialization = false);
* void on\_catalogView\_doubleClicked(const QModelIndex &index);
* void newCatalogLabel\_connector();
* void updatePropertiesEditorLabel\_input(QString name = "Item");
* void on\_catalogView\_1\_clicked(const QModelIndex &index);
* void launchGitHub();
* void databaseNetworkHelpButton();
* void databaseConstructorHelpButton();

**Private:**

Functions:

Variables: