BROdeLuxe2 SDK

Software Development Kit BROdeLuxe2 2.5

The BRO Engine

Contents

General

INTRODUCTION
BRO.dll is The BRO Engine. This dll contains the BRO constants, recipe objects, calculations, conversion and recipe data exchange methods. This SDK will enumerate them per class, with a brief explanation and code snippets. Code snippets will be in **C#**.

The namespace of BRO.dll is BRO (add:.`using BRO;`) and add BRO.dll as reference to your solution.
The BRO constants are aligned with the Beer XML standard.

The Code Snippets are kept simple. Keep in mind that BROdeLuxe2 works with a Recipe List and the recipe with the focus corresponds with the index of the focused recipe in that recipe list. Examples of that situation will be explained.

BROdeLuxe handles party-gyling. When party-gyling is applied the brew can be divided into a maximum of 3 splits (theoretically n splits but that's not very practically). A standard brew is not splitted and thus has a standard brew a split value of 1. The split value is expressed as Order (base 0, thus Order = Split value – 1).

Recipe setup in BROdeLuxe2 as Recipe List.
`GlobalVars.Recipes` is a Recipe List object with as index `GlobalVars.RecipeIndex`.

Recipe[0]	Recipe[1]	Recipe[2]	Recipe[3]	Recipe[4]	…

A recipe can be split up in 3 splits at maximum when party-gyling is applied. Theoretically it can be n splits, but that's very unrealistic.
A recipe is constructed with the following objects:

System Parameters
General, Brewers 1 - n
Objects, 1 – 3 splits
Fermentables, 1 - n
Mash Scheme, 1 – n mash steps
Sparge
Hops, 1- n per 1 – 3 splits
Additives, 0- n per 1 – 3 splits
Yeasts, 1- n per 1 – 3 splits
Water
Evaluations, 1 – 3 splits
Characteristics, 1 – 3 splits

BROdeLuxe recipes have status's. For every correct step in the recipe development the corresponding status bit is set:

```
// BROdeLuxe Status Bit settings (english):
// Bit  0 = 2^0  = 1        valid Objectives
// Bit  1 = 2^1  = 2        valid Mash Scheme
// Bit  2 = 2^2  = 4        valid Throw
// Bit  3 = 2^3  = 8        valid HopYield Thick Beer
// Bit  4 = 2^4  = 16       valid General
// Bit  5 = 2^5  = 32       valid HopYield Thin Beer
// Bit  6 = 2^6  = 64       valid Evaluation
// Bit  7 = 2^7  = 128      valid Yeast
// Bit  8 = 2^8  = 256      valid Additives
// Bit  9 = 2^9  = 512      valid Characteristics
// Bit 10 = 2^10 = 1024     valid Water
// Bit 11 = 2^11 = 2048     valid Sparge
// Bit 12 = 2^12 = 4096     valid BROSystem Parameters
```

The status bit order setting is historically developed and may look unlogic.

Address support issues (questions, examples, enhancement requests and so on) to BROdeLuxe2@gmail.com .

The BRO Constants

```
// constants
public const string HOPUSAGE_BITTER = "BITTER";
public const string HOPUSAGE_FIRSTWORT = "FIRST WORT";
public const string HOPUSAGE_AROMA = "AROMA";
public const string HOPUSAGE_DRYHOP = "DRY HOP";
public const string HOPUSAGE_HOPBED = "HOPBED";
public const string HOPUSAGE_MASH = "MASH";
public const string HOPUSAGE_BOIL = "BOIL";
public const string HOPFORM_LEAF = "LEAF";
public const string HOPFORM_PELLET = "PELLET";
public const string HOPFORM_PLUG = "PLUG";
public const string FERMENTABLETYPE_GRAIN = "GRAIN";
public const string FERMENTABLETYPE_SUGAR = "SUGAR";
public const string FERMENTABLETYPE_EXTRACT = "EXTRACT";
public const string FERMENTABLETYPE_DRYEXTRACT = "DRY EXTRACT";
public const string FERMENTABLETYPE_ADJUNCT = "ADJUNCT";
public const string YEASTTYPE_ALE = "ALE";
public const string YEASTTYPE_LAGER = "LAGER";
public const string YEASTTYPE_WHEAT = "WHEAT";
public const string YEASTTYPE_WINE = "WINE";
public const string YEASTTYPE_CHAMPAGNE = "CHAMPAGNE";
public const string YEASTFORM_LIQUID = "LIQUID";
public const string YEASTFORM_DRY = "DRY";
public const string YEASTFORM_SLANT = "SLANT";
public const string YEASTFORM_CULTURE = "CULTURE";
public const string ADDITIVETYPE_SPICE = "SPICE";
public const string ADDITIVETYPE_FINING = "FINING";
public const string ADDITIVETYPE_WATER_AGENT = "WATER AGENT";
public const string ADDITIVETYPE_HERB = "HERB";
public const string ADDITIVETYPE_FLAVOR = "FLAVOR";
public const string ADDITIVETYPE_OTHER = "OTHER";
public const string ADDITIVEUSE_BOIL = "BOIL";
public const string ADDITIVEUSE_MASH = "MASH";
public const string ADDITIVEUSE_PRIMARY = "PRIMARY";
public const string ADDITIVEUSE_SECONDARY = "SECONDARY";
public const string ADDITIVEUSE_BOTTLING = "BOTTLING";
public const string ADDITIVEUNIT_GRL = "GRAM PER LITER";
public const string ADDITIVEUNIT_MLL = "ML PER LITER";
public const string ADDITIVEUNIT_PL = "PIECES PER LITER";
public const string ADDITIVEUNIT_GR = "GRAM";
public const string ADDITIVEUNIT_ML = "ML";
public const string ADDITIVEUNIT_P = "PIECES";
public const int STATUS_OBJECTIVES = 1;
public const int STATUS_MASHSCHEME = 2;
public const int STATUS_THROW = 4;
public const int STATUS_HOP_THICK = 8;
public const int STATUS_GENERAL = 16;
public const int STATUS_HOP_THIN = 32;
public const int STATUS_EVALUATION = 64;
public const int STATUS_YEASTS = 128;
public const int STATUS_ADDITIVES = 256;
public const int STATUS_CHARACTERISTICS = 512;
public const int STATUS_WATER = 1024;
public const int STATUS_SPARGE = 2048;
public const int STATUS_SYSTEMPARMS = 4096;
public const int HARDNESS_MGPERLITERCALCIUMCARBONATE = 1;
public const int HARDNESS_GERMANDEGREES = 2;
public const int HARDNESS_FRENCHDEGREES = 4;
public const int HARDNESS_AMERICANDEGREES = 8;
public const int HARDNESS_ENGLISHDEGREES = 16;
public const double CARBONATION_MOTHERLY = 2.25;
```

The Recipe Objects

```
public class Additive

// Object
public Additive(
            string AdditiveName,
            double Amount,
            string Note = "",
            byte Order = 0,
            string Type = "",
            string Use = "",
            double Time = 0,
            string Unit = "")

// New Additive
public static BRO.Additive NewAdditive()

// New List Additive
public static List<BRO.Additive> NewListAdditives()
```

C#:
Create Additive and assign values.

```
// instantiate Additive
BRO.Additive BrewSaltAdditive = BRO.Additive.NewAdditive();

// set values
BrewSaltAdditive.AdditiveName = “SaltName”;
BrewSaltAdditive.Amount = 20;
BrewSaltAdditive.Note = “Used to modify brew water”;
BrewSaltAdditive.Order = 0;
```

```
public class BeerStyle

// Object
public BeerStyle(
            string Name,
            string Category = "",
            int Category_Number = 0,
            string Style_Letter = "",
            string Style_Guide = "",
            string Type = "",
            double OG_MIN = 0,
            double OG_MAX = 0,
            double FG_MIN = 0,
            double FG_MAX = 0,
            double IBU_MIN = 0,
            double IBU_MAX = 0,
            double COLOR_MIN = 0,
            double COLOR_MAX = 0,
            double CARB_MIN = 0,
            double CARB_MAX = 0,
            double ABV_MIN = 0,
            double ABV_MAX = 0)

// New BeerStyle
public static BRO.BeerStyle NewBeerStyle()

// New List BeerStyle
public static List<BRO.BeerStyle> NewListBeerStyles()
```

C#:
Add beerstyle to recipe:

```
// Create Recipe Object
BRO.Recipe MyRecipe = BRO.Recipe. NewRecipe();

// Add Beerstyle object
MyRecipe.BeerStyles.Add(BRO.BeerStyle.NewBeerStyle());
```

```
public class BROBrewer

// Object
public BROBrewer(
            string Name,
            string EmailAddress = "")

// New Brewer
public static BRO.BROBrewer NewBrewer()

// New List Brewer
public static List<BRO.BROBrewer> NewListBrewers()
```

C#:
Load brewers into datagridview. Brewers are retrieved from object `GlobalVars.BROGeneral.`

```
// Show data Brewers
private void ShowDataBrewers()
{
  // Brewers
  for (int C = 0; C < GlobalVars.BROGeneral.Brewers.Count(); C++)
  {
    // only add line when there is info to show
    if (GlobalVars.BROGeneral.Brewers[C].Name.Length > 0)
    {
      this.dataGridView1.Rows.Add(GlobalVars.BROGeneral.Brewers[C].Name.ToString(),
      GlobalVars.BROGeneral.Brewers[C].EmailAddress.ToString());
    }
  }
}
```

```
public class BROGeneral

// Object
public BROGeneral(
            string BeerName,
            List<BROBrewer> Brewers,
            DateTime BrewingDate,
            DateTime BottlingDate,
            string Comments = "",
            Boolean PartyGyling = false,
            string RecipeFileName = "",
            string BROdeLuxeVersion = "",
            int Splits = 1,
            int RecipeType = 0) // 0 = Beer; 1 = Mead; 2 = Stookwijn; 3 = Moonshine

// New BROGeneral
public static BRO.BROGeneral NewBROGeneral()
```

C#:
Show values from object `GlobalVars.BROGeneral` in controls.

```
// Show General Data
private void ShowDataRecipeGeneral()
{

  // show Data
  this.txtBeerName.Text = GlobalVars.BROGeneral.BeerName;
  this.monthcalBrewDate.SetDate (GlobalVars.BROGeneral.BrewingDate);
  this.monthcalBrewDate.AddBoldedDate(GlobalVars.BROGeneral.BrewingDate);
  this.monthcalBottleDate.SetDate(GlobalVars.BROGeneral.BottlingDate);
  this.monthcalBottleDate.AddBoldedDate(GlobalVars.BROGeneral.BottlingDate);
  this.chbGeneral_PartyGyling.Checked = GlobalVars.BROGeneral.PartyGyling;
  this.txtComments.Text = GlobalVars.BROGeneral.Comments;

  // for older recipes
  this.lblBrewDate.Text = "Brouwdatum: " + GlobalVars.BROGeneral.BrewingDate.ToString("dd-  MMM-
  yyyy");
  this.lblBottleDate.Text = "Botteldatum: " + GlobalVars.BROGeneral.BottlingDate.ToString("dd-
  MMM-yyyy");
}
```

```
public class BROObjectives

// Object
public BROObjectives(
            double Volume,
            double AlcoholPercentage,
            double FG,
            double CO2Percentage,
            double EBU,
            string BeerType)

// New BROObjectives
public static BRO.BROObjectives NewBROObjectives()

// New List BROObjectives
public static List<BRO.BROObjectives> NewListBROObjectives()
```

C#:
Sweep data from form controls into a BROObjectives object called `GlobalVars.BROObjectives` . The `Order` indicates the split (0 for standard brews and 0-2 when party-gyling is performed).

```
// sweep data into GlobalVars.BROObjectives[Order]

// add beer objectives
GlobalVars.BROObjectives[Order].Volume = Convert.ToDouble(this.txtVolume.Text);
GlobalVars.BROObjectives[Order].AlcoholPercentage = Convert.ToDouble(this.txtAlcohol.Text);
GlobalVars.BROObjectives[Order].FG = Convert.ToDouble(this.txtEindSG.Text);
GlobalVars.BROObjectives[Order].CO2Percentage = Convert.ToDouble(this.txtCarbonation.Text);
GlobalVars.BROObjectives[Order].EBU = Convert.ToDouble(this.txtEBU.Text);
```

```
public class BROSystemParameters

// Object
public BROSystemParameters(
            double VolumeMashKettle = 0,
            double VolumeLauterTun = 0,
            double VolumeBoilingKettle = 0,
            double VolumeUnderFilterSheet = 0,
            double WaterAddedUnderFilterSheet = 0,
            double Evaporation = 0,
            double BrewLoss = 0,
            string Name = "",
            string Note = "",
            int EquipmentType = 0)


// New System Parameters
public static BRO.BROSystemParameters NewSystemParameters()
```

C#:
Declare object `m_DefaultSystemParameters` and assign values from global variables (filled during start up from the config file).

```
public BRO.BROSystemParameters m_DefaultSystemParameters = new BRO.BROSystemParameters();

// sweep System parameters in m_DefaultSystemParameters
m_DefaultSystemParameters.VolumeMashKettle = GlobalVars.VolumeMashKettle;
m_DefaultSystemParameters.VolumeLauterTun = GlobalVars.VolumeLauterTun;
m_DefaultSystemParameters.VolumeBoilingKettle = GlobalVars.VolumeBoilingKettle;
m_DefaultSystemParameters.VolumeUnderFilterSheet = GlobalVars.VolumeUnderFilterSheet;
m_DefaultSystemParameters.WaterAddedUnderFilterSheet = GlobalVars.WaterAddedUnderFilterSheet;
m_DefaultSystemParameters.Evaporation = GlobalVars.Evaporation;
m_DefaultSystemParameters.BrewLoss = GlobalVars.BrewLoss;
m_DefaultSystemParameters.Name = GlobalVars.NameDefaultSystemParameters;
m_DefaultSystemParameters.Note = GlobalVars.NoteDefaultSystemParameters;
m_DefaultSystemParameters.EquipmentType = GlobalVars.EquipmentType;
```

```
public class CalculatedResults

// Object
public CalculatedResults(
            List<double> E_Gram,
            List<double> E_OG,
            List<double> E_SugarConcentration,
            List<double> E_EBC,
            List<double> R_AlcVolPerc,
            List<double> R_BZR,
            List<double> R_SchijnVG,
            List<double> R_EBU,
            List<double> R_EBC,
            List<double> E_PGTransitionInPlato,
            List<double> E_PGTransitionInLiter,
            double E_TotallSugar = 0,
            double E_TotallThrow = 0,
            double E_TotallMash = 0,
            double E_BZR = 0,
            double E_MashWaterVolume = 0,
            double E_SGHoofdWort = 0,
            double E_VolHoofdWort = 0,
            double E_VrijHoofdWort = 0,
            double E_WortNaSpoelen = 0,
            double E_SpargeWater = 0,
            double E_EBU_AH = 0,
            double E_DecoctionVolume = 0,
            double E_DecoctionAdjustVolume = 0)

// New CalculatedResults
public static BRO.CalculatedResults NewCalculatedResults()
```

`CalculatedResults` is an internal used object to store the estimated/calculated (E_) and measured/real (R_) values.

```
public class Characteristics

// Object
public Characteristics(
            string Colour = "",
            string Clearity = "",
            string Foam = "",
            string FoamStability = "",
            string Body = "",
            string MouthFeel = "",
            string Carbonation = "",
            DateTime ObservationDate = default(DateTime),
            byte Order = 0,
            string Note = "")

// New Characteristics
public static BRO.Characteristics NewCharacteristics()

// New List Characteristics
public static List<BRO.Characteristics> NewListCharacteristics()
```

C#:
Sweep data into object
`GlobalVars.Recipes[GlobalVars.RecipeIndex].RecipeCharacteristics[Order].` from form controls. The focused recipe is indicated with `GlobalVars.RecipeIndex` and `GlobalVars.Recipes` is a Recipe List object. The `Order` indicates the split (0 for standard brews and 0-2 when party-gyling is performed).

```
//update first item in array ListCharacteristics
GlobalVars.Recipes[GlobalVars.RecipeIndex].RecipeCharacteristics[Order].Colour = lblKleurR.Text;
GlobalVars.Recipes[GlobalVars.RecipeIndex].RecipeCharacteristics[Order].Clearity =
lblHelderheidR.Text;
GlobalVars.Recipes[GlobalVars.RecipeIndex].RecipeCharacteristics[Order].Foam = lblSchuimR.Text;
GlobalVars.Recipes[GlobalVars.RecipeIndex].RecipeCharacteristics[Order].FoamStability =
lblSchuimStabiliteitR.Text;
GlobalVars.Recipes[GlobalVars.RecipeIndex].RecipeCharacteristics[Order].Body = lblBodyR.Text;
GlobalVars.Recipes[GlobalVars.RecipeIndex].RecipeCharacteristics[Order].MouthFeel =
lblMondGevoelR.Text;
GlobalVars.Recipes[GlobalVars.RecipeIndex].RecipeCharacteristics[Order].Carbonation =
lblKoolzuurR.Text;
GlobalVars.Recipes[GlobalVars.RecipeIndex].RecipeCharacteristics[Order].ObservationDate =
Convert.ToDateTime(this.monthcalObservationDate.SelectionRange.Start).Date;
GlobalVars.Recipes[GlobalVars.RecipeIndex].RecipeCharacteristics[Order].Note = Note.Text;
```

```
public class Evaluation

// Object
public Evaluation(
            double RealVolume = 0,
            double RealOG = 0,
            double RealFG = 0,
            string Note = "",
            byte Order = 0,
            double RealStartPlato = -1,
            double RealEndPlato = -1,
            double pH_YoungBeer = -9999)

// New Evaluation
public static BRO.Evaluation NewEvaluation()

// New List Evaluation
public static List<BRO.Evaluation> NewListEvaluations()
```

C#:
Sweep data into object `GlobalVars.Evaluations.` The `Order` indicates the split (0 for standard brews and 0-2 when party-gyling is performed).

```
// Sweep values from Evaluations object into form controls
this.txtEvaluation_Volume.Text = GlobalVars.Evaluations[Order].RealVolume.ToString();
this.txtEvaluation_OG.Text = GlobalVars.Evaluations[Order].RealOG.ToString();
this.txtEvaluation_FG.Text = GlobalVars.Evaluations[Order].RealFG.ToString();
this.txtEvaluation_Note.Text = GlobalVars.Evaluations[Order].Note;
```

```
public class Fermentable

// Object
public Fermentable(
            string FermentableName,
            double Percentage,
            double ExtractionEfficiency,
            double EBC,
            bool KettleAddition,
            double Amount = 0,
            string Note = "")

// New Fermentable
public static BRO.Fermentable NewFermentable()

// New List Fermentables
public static List<BRO.Fermentable> NewListFermentables()
```

C#:
How to add the Fermentables to a datagridview?

```
// Show Fermentables
private void ShowDataFermentables()
{
// Clear grid
       this.dataGridView1.Rows.Clear();

       // Show data in grid
       for (int C = 0 ; C < GlobalVars.Fermentables.Count(); C++)
       {
       this.dataGridView1.Rows.Add(GlobalVars.Fermentables[C].FermentableName,
              GlobalVars.Fermentables[C].Percentage,
              GlobalVars.Fermentables[C].ExtractionEfficiency.ToString(),
              GlobalVars.Fermentables[C].EBC,
              GlobalVars.Fermentables[C].KettleAddition,
              GlobalVars.Fermentables[C].Note);
       }
}
```

```
public class Hop

// Object
public Hop(
            string HopName,
            string HopForm,
            double AlphaPercentage,
            double BetaPercentage,
            string Usage,
            double BoilingTime,
            double PercentageToTotallBitterness,
            double SG,
            double Amount,
            string Note = "",
            byte Order = 0,
            double HopStorageFactor = 1)

// New Hop
public static BRO.Hop NewHop()

// New List Hop
public static List<BRO.Hop> NewListHops()
```

C#:
How to find the max boiling time time of all hops used (LINQ used!)?

```
// Initialize
double DestinationBoilingTime = GlobalVars.Recipes[GlobalVars.RecipeIndex].Hops.Max((HPS) => {
return (HPS.BoilingTime); });
```

C#:
How to find the hops per split in case of party-gyling (LINQ used!)? `Order` contains Order number (0, 1 or 2).

```
// Hops list object
List<BRO.Hop> HopsPerSplit = new List<BRO.Hop>();

// Find the Hops per Split
HopsPerSplit = GlobalVars.Hops.FindAll((HPS) => { return (HPS.Order == Order); }).ToList();
```

C#:
How to find the bitter hops in the `HopsPerSplit` example?

```
// Bitterhop
if (HopsPerSplit[C].Usage.ToUpper() == BRO.BROConstants.HOPUSAGE_BITTER ||
HopsPerSplit[C].Usage.ToUpper() == BRO.BROConstants.HOPUSAGE_BOIL ||
HopsPerSplit[C].Usage.ToUpper() == BRO.BROConstants.HOPUSAGE_FIRSTWORT)
{
       // Do your thing…
}
```

```
public class Mash

// Object
public Mash(
            double WaterGrainRatio,
            double pHMash,
            List<MashStep> MashSteps,
            bool MashViaDecoction = false,
            double TemperatureDecoctionStep = 0,
            double TimeDecoctionStep = 0,
            double TemperatureBrewWater = 0,
            string Note = "")

// New Mash
public static BRO.Mash NewMash()
```

C#:
Sweep data from form control of mash scheme and sweep the data into mash object `GlobalVars.Mash`.

```
// sweep pH and WaterToGrainRatio into Global Recipe Mash
GlobalVars.Mash.WaterGrainRatio = Convert.ToDouble(this.txtMashRatio.Text);
GlobalVars.Mash.pHMash = Convert.ToDouble(this.txtpH.Text);
GlobalVars.Mash.Note = this.txtMashScheme_Note.Text;

// sweep data into Global Recipe Mash
if (this.chbMashViaDecoction.Checked)
{
  GlobalVars.Mash.MashViaDecoction = true;
  GlobalVars.Mash.TemperatureDecoctionStep =
Convert.ToDouble(this.txtTemperatureDecoction.Text);
  GlobalVars.Mash.TimeDecoctionStep = Convert.ToDouble(this.txtTimeDecoction.Text);
  GlobalVars.Mash.TemperatureBrewWater = Convert.ToDouble(this.txtTemperatureBrewwater.Text);
}
else
{
  GlobalVars.Mash.MashViaDecoction = false;
  GlobalVars.Mash.TemperatureDecoctionStep = 0;
  GlobalVars.Mash.TimeDecoctionStep = 0;
  GlobalVars.Mash.TemperatureBrewWater = 0;
}

// clear Mash Steps
GlobalVars.Mash.MashSteps.Clear();

// sweep data from grid into GlobalVars.Mash.MashSteps()
for (int C = 0; C < dataGridView1.Rows.Count; C++)
{
  // add row to GlobalVars.Mash.MashSteps
  try
  {
  GlobalVars.Mash.MashSteps.Add(new BRO.MashStep(
     Convert.ToDouble(dataGridView1.Rows[C].Cells[0].Value),
     Convert.ToDouble(dataGridView1.Rows[C].Cells[1].Value),
     dataGridView1.Rows[C].Cells[2].Value.ToString()));
  }
  catch
  {
  }
}

// sweep data into recipe and recalculate recipe
GlobalVars.Recipes[GlobalVars.RecipeIndex].RecipeMash = GlobalVars.Mash;
GlobalVars.Recipes[GlobalVars.RecipeIndex] =
BRO.Calculations.CalculateRecipe(GlobalVars.Recipes[GlobalVars.RecipeIndex]);
```

```
public class MashStep

// Object
public MashStep(
            double Temperature,
            double Time,
            string Note = "")

// New MashStep
public static BRO.MashStep NewMashStep()

// New List MashStep
public static List<BRO.MashStep> NewListMashSteps()
```

C#:
Snippet from Maisch-O-Matic software (to control temperature in Mash kettle during mash scheme). Recipe is opened via file and the mash steps are retrieved.

```
//Any mash steps present?
if (m_Recipe.RecipeMash.MashSteps.Count() > 0)
{
  //Clear grid
  this.dataGridView1.Rows.Clear();

  //Add Mash Step(s) to Datagrid
  for (int i = 0; i < m_Recipe.RecipeMash.MashSteps.Count(); i++)
  {
    //Add mash Step to Data Grid
    this.dataGridView1.Rows.Add(m_Recipe.RecipeMash.MashSteps[i].Temperature.ToString(),
    m_Recipe.RecipeMash.MashSteps[i].Time.ToString());

    //inform user
    MessageViewer("Mash Step added: " + m_Recipe.RecipeMash.MashSteps[i].Temperature.ToString()
    + " 'C " + m_Recipe.RecipeMash.MashSteps[i].Time.ToString() + " min");
  }
}
```

```
public class Sparge

// Object
public Sparge(
            double SpargeTillDegreePlato,
            double RealMainWortInPlato,
            double pH,
            double Temperature,
            List<double> SpargeConcentrations,
            string Note = "",
            bool NoSparging = false)

// New Sparge
public static BRO.Sparge NewSparge()
```

C#:
Sweep data from form control of sparge form and sweep the data into sparge object `GlobalVars.Sparge`
Hereafter, put sparge object `GlobalVars.Sparge` into recipe list
`GlobalVars.Recipes[GlobalVars.RecipeIndex].` and recalculate recipe.

```
// Grab data
GlobalVars.Sparge.SpargeTillDegreePlato = Convert.ToDouble(this.txtS_SpoelenTot.Text);
GlobalVars.Sparge.RealMainWortInPlato = Convert.ToDouble(this.txtS_WerkelijkSGHoofdWort.Text);
GlobalVars.Sparge.pH = Convert.ToDouble(this.txtS_pH.Text);
GlobalVars.Sparge.Temperature = Convert.ToDouble(this.txtS_Temperature.Text);
GlobalVars.Sparge.Note = this.txtS_Note.Text;

// sweep data into recipe and recalculate recipe
GlobalVars.Recipes[GlobalVars.RecipeIndex].RecipeSparge = GlobalVars.Sparge;
GlobalVars.Recipes[GlobalVars.RecipeIndex] =
BRO.Calculations.CalculateRecipe(GlobalVars.Recipes[GlobalVars.RecipeIndex]);
```

```
public class Water

// Object
public Water(
            string Name,
            double Calcium,
            double Magnesium,
            double Sodium,
            double Chloride,
            double Sulphate,
            double CO3,
            double HCO3,
            double pH,
            double Alkanality = 0,
            double TotallHardness = 0,
            double PercentageDemi = 0,
            string Comments = "",
            int IDWaterProfileSource = 0,
            int IDWaterProfileDestination = 0)

// New Water
public static BRO.Water NewWater()
```

C#:
Sweep data from `dataGridView1` form control of sparge form and sweep the data into sparge object `GlobalVars.Water .`

```
// Grab values Water Object
GlobalVars.Water.Name = this.txtW_Name.Text;
GlobalVars.Water.Calcium = Convert.ToDouble(dataGridView1.Rows[0].Cells[0].Value);
GlobalVars.Water.Magnesium = Convert.ToDouble(dataGridView1.Rows[0].Cells[1].Value);
GlobalVars.Water.Sulphate = Convert.ToDouble(dataGridView1.Rows[0].Cells[2].Value);
GlobalVars.Water.Sodium = Convert.ToDouble(dataGridView1.Rows[0].Cells[3].Value);
GlobalVars.Water.Chloride = Convert.ToDouble(dataGridView1.Rows[0].Cells[4].Value);
GlobalVars.Water.TotallHardness = Convert.ToDouble(dataGridView1.Rows[0].Cells[5].Value);
GlobalVars.Water.CO3 = Convert.ToDouble(dataGridView1.Rows[0].Cells[6].Value);
GlobalVars.Water.HCO3 = Convert.ToDouble(dataGridView1.Rows[0].Cells[7].Value);
GlobalVars.Water.Alkanality = Convert.ToDouble(dataGridView1.Rows[0].Cells[8].Value);
GlobalVars.Water.pH = Convert.ToDouble(this.txtW_pH.Text);
GlobalVars.Water.PercentageDemi = Convert.ToDouble(this.txtW_PercentageDemi.Text);
GlobalVars.Water.Comments = this.txtW_Note.Text;
GlobalVars.Water.IDWaterProfileSource = cmbWaterProfileSource.SelectedIndex + 1;
GlobalVars.Water.IDWaterProfileDestination = cmbWaterProfileDestination.SelectedIndex + 1;
```

C#:
Spin-off from Water treatment. When a brew salt is used, add the brew salt to the additives.

```
// Add Brew Salt To Additives
private void AddBrewSaltToAdditives(string Salt, double GramsOfSalt, string Description)
{
  // instantiate Additive
  BRO.Additive BrewSaltAdditive = BRO.Additive.NewAdditive();

  // set values
  BrewSaltAdditive.AdditiveName = Salt;
  BrewSaltAdditive.Amount = GramsOfSalt;
  BrewSaltAdditive.Note = Description;
  BrewSaltAdditive.Order = 0;

  // check if additive already exist within Recipe
  int IndexOfBrewSalt = GlobalVars.Recipes[GlobalVars.RecipeIndex].Additives.FindIndex(BrewSalt
  => BrewSalt.AdditiveName == Salt);

  if (IndexOfBrewSalt == -1) // not found
  {
```

```
    // Add additive
    GlobalVars.Recipes[GlobalVars.RecipeIndex].Additives.Add(BrewSaltAdditive);
  }
  else // found, thus replace the item
  {
    // replace additive
    GlobalVars.Recipes[GlobalVars.RecipeIndex].Additives[IndexOfBrewSalt] = BrewSaltAdditive;
  }
}
```

```
public class Yeast

// Object
public Yeast(
            string YeastName,
            string VendorName,
            string Type,
            double Amount,
            string Note = "",
            byte Order = 0,
            string YeastForm = "")

// New Yeast
public static BRO.Yeast NewYeast()

// New List Yeast
public static List<BRO.Yeast> NewListYeasts()
```

C#:
Find the selected yeasts per split. The `Order` indicates the split (0 for standard brews and 0-2 when party-gyling is performed).

```
// define Yeast list object
List<BRO.Yeast> YeastsPerSplit = new List<BRO.Yeast>();

// Grab Yeasts per Split
YeastsPerSplit = GlobalVars.Yeasts.FindAll((YPS) => { return (YPS.Order == Order); }).ToList();
```

```
public class Recipe

// Object
public Recipe(
            BROGeneral RecipeGeneral,
            List<BROObjectives> RecipeObjectives,
            Water RecipeWater,
            List<Fermentable> Fermentables,
            List<Hop> Hops,
            Mash RecipeMash,
            Sparge RecipeSparge,
            List<Yeast> Yeasts,
            List<Additive> Additives,
            List<Evaluation> RecipeEvaluations,
            List<Characteristics> RecipeCharacteristics,
            CalculatedResults RecipeResults,
            BROSystemParameters RecipeSystemParameters,
            List<BeerStyle> BeerStyles,
            int Status,
            string RecipeGuid = "",
            bool RecipeChanged = false)

// New Recipe
public static BRO.Recipe NewRecipe()

// New List Recipes
public static List<BRO.Recipe> NewListRecipes()
```

C#:
In this snippet, the recipe list is declared in class `GlobalVars`.

```
static class GlobalVars
{
  static List<BRO.Recipe> m_Recipes = new List<BRO.Recipe>();

  // Recipes
  public static List<BRO.Recipe> Recipes
  {
    get { return m_Recipes; }
    set { m_Recipes = value; }
  }
}
```

C#:
In this snippet, a new recipe is added to the list and the global vars `GlobalVars.NumberOfLoadedRecipes` and `GlobalVars.RecipeIndex` are updated.

```
// new empty recipe to list
GlobalVars.Recipes.Add(BRO.Recipe.NewRecipe());

// increment Recipe counter and define RecipeIndex
GlobalVars.NumberOfLoadedRecipes += 1;
if (GlobalVars.NumberOfLoadedRecipes >= 1)
{
  GlobalVars.RecipeIndex = GlobalVars.NumberOfLoadedRecipes - 1;
}
```

C#:
In this snippet, a recipe is removed from the recipe list at index `GlobalVars.RecipeIndex`. The global vars `GlobalVars.NumberOfLoadedRecipes` and `GlobalVars.RecipeIndex` are updated.

```
// RecipeList
GlobalVars.Recipes.RemoveAt(GlobalVars.RecipeIndex);

// maintain Recipe counter and define RecipeIndex
if (GlobalVars.NumberOfLoadedRecipes >= 0)
{
  GlobalVars.NumberOfLoadedRecipes -= 1;
  if (GlobalVars.NumberOfLoadedRecipes >= 1)
  {
     GlobalVars.RecipeIndex = GlobalVars.NumberOfLoadedRecipes - 1;
  }
}
```

C#:
In this snippet, all recipes are closed.

```
// RecipeList
GlobalVars.Recipes.Clear();

// maintain Recipe counter and define RecipeIndex
GlobalVars.NumberOfLoadedRecipes = 0;
GlobalVars.RecipeIndex = 0;
```

C#:
In this snippet, the status bit for Yeasts is set.

```
// Update Recipe's Status if needed
if ((GlobalVars.Recipes[GlobalVars.RecipeIndex].Status & BRO.BROConstants.STATUS_YEASTS) !=
BRO.BROConstants.STATUS_YEASTS)
{
  // Set Status bit 1; 2^7 = 128 (= BRO.BROConstants.STATUS_YEASTS)
  GlobalVars.Recipes[GlobalVars.RecipeIndex].Status =
  GlobalVars.Recipes[GlobalVars.RecipeIndex].Status + BRO.BROConstants.STATUS_YEASTS
}
```

The Calculations

```
public class Calculations


// Calculate Recipe - BROdeLuxe's New ENGINE!
public static BRO.Recipe CalculateRecipe(BRO.Recipe R)
```

C#:
Recalculate recipe with index `GlobalVars.RecipeIndex` and sweep the data in the same recipe.

```
// recalculate recipe
GlobalVars.Recipes[GlobalVars.RecipeIndex] =
BRO.Calculations.CalculateRecipe(GlobalVars.Recipes[GlobalVars.RecipeIndex]);
```

```
// Clear Recipe
public static BRO.Recipe ClearRecipe(BRO.Recipe R)

// Alkanality
public static double Alkanality(double Carbonate, double Bicarbonate)

// Amount Sugar For Bottling
public static double AmountSugarForBottling(double DesiredCO2, double BottleVolumeInLiters)

// Mash Volume
public static double MashVolume(BRO.Recipe R)
```

C#:
This snippets calculates the mash volume from recipe with index `GlobalVars.RecipeIndex` from recipe list `GlobalVars.Recipes`.

```
// Estimate Mash Volume
double MashVolume = BRO.Calculations.MashVolume(GlobalVars.Recipes[GlobalVars.RecipeIndex]);
```

```
// Boiling Time Based on Evaporation
public static double BoilingTimeBasedOnEvaporation(BRO.Recipe R)

// Calorie Calculation according to Ernst de Moor
public static double CalorieCalculation(BRO.Recipe R)

// Estimation of Begin SG based on End SG and SG Brix, according to Bill Pierce.
public static double EstimateStartSG_EndSG_EndBrix(double EndSG, double EndBrix)
```

The Conversions

```
public class Conversions

// Convert EBC to SRM
public static double Convert_EBC_SRM(double EBC)

// Convert SRM to EBC
public static double Convert_SRM_EBC(double SRM)

// Convert Concentration to Plato
public static double Convert_Conc_Plato(double Concentration)

// Convert Plato to Concentration
public static double Convert_Plato_Conc(double Plato)

// Convert SG (Specific Gravity) to Plato
public static double Convert_SG_Plato(double SG)

// Convert Plato to SG (Specific Gravity)
public static double Convert_Plato_SG(double Plato)

// Convert Concentration to SG (SG=Specific Gravity)
public static double Convert_Conc_SG(double Concentration)

// Convert SG to Concentration (SG=Specific Gravity)
public static double Convert_SG_Conc(double SG)
```

C#:
This snippets converts specific gravity of main wort into sugar concentration.

```
// get sugar concentration
double SugarConcentration = BRO.Conversions.Convert_SG_Conc(SGHoofdwort);
```

```
// Convert Start Brix to Start SG (kg/m3)
public static double Convert_StartBrix_StartSG(double Brix)

// Convert Start Brix, End Brix and Temperature to End SG (kg/m3)
public static double Convert_Brix_EndSG(double Start_Brix, double End_Brix, double
Temperature)

// Convert Degrees Celsius to Fahrenheit
public static double Convert_Celsius_Fahrenheit(double dCelsius)

// Convert Degrees Fahrenheit to Celsius
public static double Convert_Fahrenheit_Celsius(double dFahrenheit)

// Convert US Gallons to Liters
public static double Convert_USGallons_Liters(double dGallons)

// Convert Liters to US Gallons
public static double Convert_Liters_USGallons(double dLiters)

// Convert UK Gallons to Liters
public static double Convert_UKGallons_Liters(double dGallons)

// Convert Liters to UK Gallons
public static double Convert_Liters_UKGallons(double dLiters)

// Convert UK Pints to Liters
```

```
public static double Convert_UKPints_Liters(double dPints)

// Convert Liters to UK Pints
public static double Convert_Liters_UKPints(double dLiters)

// Convert Pint to Liters
public static double Convert_Pints_Liters(double dPints)

// Convert Liters to Pints
public static double Convert_Liters_Pints(double dLiters)

// Convert Ounce to Liters
public static double Convert_Ounces_Liters(double dOunces)

// Convert Liters to Ounces
public static double Convert_Liters_Ounces(double dLiters)

// Convert UK Ounce to Liters
public static double Convert_UKOunces_Liters(double dOunces)

// Convert Liters to UK Ounces
public static double Convert_Liters_UKOunces(double dLiters)

// Convert Ounces to Kilograms
public static double Convert_Ounces_Kilograms(double dOunces)

// Convert Kilograms to Ounces
public static double Convert_Kilograms_Ounces(double dKilograms)

// Convert Pounds to Kilograms
public static double Convert_Pounds_Kilograms(double dPounds)

// Convert Kilograms to Pounds
public static double Convert_Kilograms_Pounds(double dKilograms)

// Convert Grains to Kilograms
public static double Convert_Grains_Kilograms(double dGrains)

// Convert Kilograms to Grains
public static double Convert_Kilograms_Grains(double dKilograms)
```

The Recipe Data Exchange

```
public class RecipeDataExchange


// Save Recipe in BRO format
public static void SaveRecipeBRO(BRO.Recipe R, string BROFilePath)
```

C#:
This snippet shows how to save a recipe in *.bro format.

```
// BRO
BRO.RecipeDataExchange.SaveRecipeBRO(MyRecipe, @"C:\Temp\MyRecipe.bro");
```

C#:
This snippet shows how to save a recipe in *.bro format. It uses the `RecipeGeneral.RecipeFileName` as location.

```
// BRO
BRO.RecipeDataExchange.SaveRecipeBRO(GlobalVars.Recipes[GlobalVars.RecipeIndex],
GlobalVars.Recipes[GlobalVars.RecipeIndex].RecipeGeneral.RecipeFileName);
```

```
// Save Recipe in KBG format
public static void SaveRecipeKBG(BRO.Recipe R, string KBGFilePath)

// Save Recipe in XML format
public static void SaveRecipeXML(BRO.Recipe R, string XMLFilePath)

// Open Recipe in BRO Format
public static BRO.Recipe OpenRecipeBRO(string BROFilePath)
```

C#:
This snippet shows how to open a recipe in *.bro format, using a predefined full path or using the open file dialog path.

```
GlobalVars.Recipes[GlobalVars.RecipeIndex] =
BRO.RecipeDataExchange.OpenRecipeBRO(@"C:\Temp\MyRecipe.bro");

GlobalVars.Recipes[GlobalVars.RecipeIndex] =
BRO.RecipeDataExchange.OpenRecipeBRO(ofdOpenRecipe.FileName);
```

```
// Open Recipe in KBG Format
public static BRO.Recipe OpenRecipeKBG(string KBGFilePath)

// Open Recipe in XML
public static BRO.Recipe OpenRecipeXML(string XMLFilePath)

// Return Recipe as String
public static string ReturnRecipeAsString(string RecipeFilePath)

// Create Brewers List
public static string CreateBrewersList(BRO.Recipe R)

// Create Shopping List
public static string CreateShoppingList(BRO.Recipe R)
```

```
// Export Sparge Curve as CSV Trace
public static void ExportSpargeCurveAsCSVTrace(BRO.Recipe R, string
DestinationFullFilePath)
```