AS Data Manager

User Manual

Title	AS Data Manager
Document Type	User Manual
File Reference	AS_Datamanager.library
Version	2.1
Date	22.06.2026
Author	ASKS GmbH
Status	Released

ASKS GmbH · Tevesstraße 29a · 78176 Blumberg · www.as-ks.de · support@as-ks.de

Summary: This user guide describes the AS Data Manager – a CODESYS library for the persistent storage and restoration of PLC parameters (recipes, formats, machine parameters). It covers integration, initialization, data structuring, save and load operations, as well as the provided visualizations and extension options.

1 Overview

The AS Data Manager enables the persistent storage and restoration of parameters in a PLC – for example recipes, formats, or machine parameters. Data is stored in a CSV file by default. Additional storage formats can be implemented on request.

The key features of the library include saving and restoring data records in CSV files, automatic data backup when saving, and the ability to save and restore user-defined data types. Groups and values can be nested arbitrarily. Changes in the Data Manager, in groups, or in individual values are reported via notifications. Separate visualizations are available for the Data Manager, groups, group lists, value lists, and individual values. The library supports client-specific translation as well as extensible unit conversion.

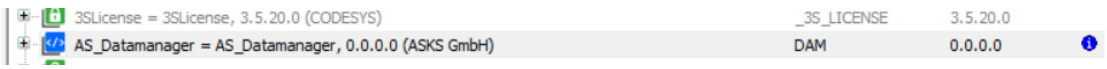
All error codes of the library are defined in the ERROR structure. The AS_Datamanager library provides the basic data types.

2 Commissioning

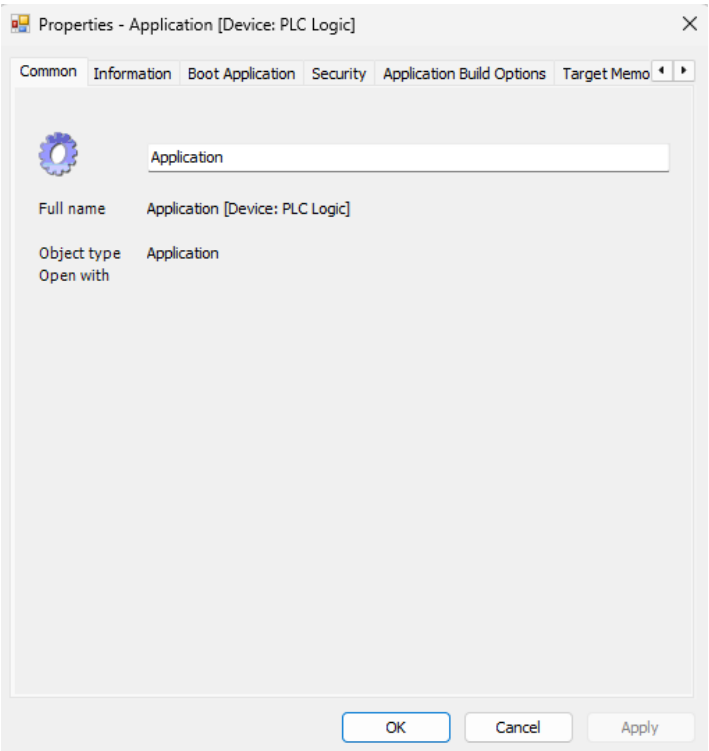
This chapter describes how the Data Manager is integrated and configured in a CODESYS project.

2.1 Application

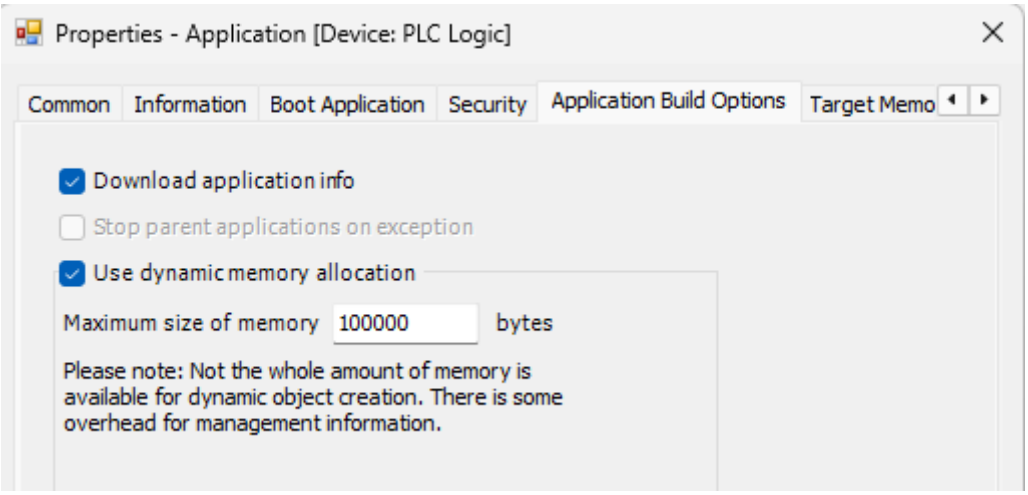
To use the Data Manager functions, the AS_Datamanager library must first be added in the Library Manager.



The dynamic memory of the application must then be increased. To do this, open the context menu under the “Application” node and select “Properties”.



Under “Application Build Options”, the “Use dynamic memory allocation” option must be enabled and the maximum size increased.



> **NOTE:** Without increasing the dynamic memory, the data manager cannot be initialized.

2.2 Initialize Data Manager

A Data Manager instance is declared in the CODESYS program and then initialized. The instance manages all groups and values as well as storage access.

```
Recipes.Init(                       // File - Recipe Management
        sName               := 'Recipe Management',     // Datamanager Name
        sTranslationFile    := 'txtTestTranslations',   // [] Translation file name
        sDir                := 'Recipes',               // This is the directory where to save/load the files
        sFileType           := 'csv',                   // [dmf] This is the name of the ending (txt, csv, ...)
        xDoBackup           := TRUE,                    // [FALSE] Datamanager is doing a backup if there are pending changes on save
        xLoadMinMax         := FALSE);                  // [FALSE] Load the Min - Max limits form the file
```

2.3 Groups

Groups structure data records hierarchically. They can be nested to any depth (see note), enabling a clear organization of complex parameter sets. A group is initialized via its constructor and added to the Data Manager or to a parent group.

```
Granulation.InitGroup(                  // Group - Granulation
        sName               := 'GRANULATION MIX',           // Group Name
        sTranslationFile    := 'txtTestTranslations',       // [] Translation file name
        parent              := Line1);                      // Parent Group

Steril.InitGroup(                       // Group - Sterilization
        sName               := 'STERILIZATION CYCLE',       // Group Name
        sTranslationFile    := 'txtTestTranslations',       // [] Translation file name
        parent              := Line1);                      // Parent Group
```

> **NOTE:** The maximum path length must not exceed 255 characters.

2.4 Values

The Data Manager supports the following data types: BOOL, DINT, ENUM, INT, REAL, SINT, STRING, UDINT, UINT, USINT and TIME. Some initialization parameters are optional; default values are specified in the reference documentation.
Here are some examples:

```
ExampleBool.Init(
    group       := ExampleGrp,                  // Parent Group
    sName       := 'EXAMPLE BOOL',              // Value Name
    xDefault    := FALSE,                       // Default Value
    sUnit       := '');                         // ['] Unit of the Value

ExampleInt.Init(
    group       := ExampleGrp,                  // Parent Group
    sName       := 'EXAMPLE INT',               // Value Name
    nMin        := 0,                           // [1] Min Value
    nDefault    := 10,                          // Default Value
    nMax        := 100,                         // [0] Max Value
    sUnit       := '');                         // ['] Unit of the Value

ExampleReal.Init(
    group       := ExampleGrp,                  // Parent Group
    sName       := 'EXAMPLE REAL',              // Value Name
    fMin        := 0.0,                         // [1] Min Value
    fDefault    := 10.0,                        // Default Value
    fMax        := 100.0,                       // [0] Max Value
    sUnit       := '',                          // ['] Unit of the Value
    nDecimals   := 2);                          // [3] Decimal places
```

Initialization of some Data Manager values (optional parameters in square brackets)
All values are listed and initialized in the example project.

The current value can be accessed at runtime via the Properties of the respective function block.

```
x := ExampleBool.xValue;
i := ExampleInt.nValue;
f := ExampleReal.fValue;
n := ExampleEnum.nValue;
str := ExampleString.sValue;
```

2.5 Save and Load

To save or load a data record, the Save() or Load() methods are called and the desired record name is passed. Missing values when loading are ignored or – depending on configuration – can be set to the default value.

```
IF xSave THEN
    xSave := FALSE;
    Recipes.Save('Recipes');
END_IF

IF xLoad THEN
    xLoad := FALSE;
    Recipes.Load('Recipes');
END_IF
```

2.6 Translation

Translation can be done in two ways: via the global language handler (application-wide) or via a group-specific text list. Both approaches can be combined within the same application.

Global Translation
A language file is registered once via `DAM.LAN.GV_LAN.g_language.AddGlobalLanguageFile()` and is then available to all groups. This typically happens during application initialization.

```
DAM.LAN.GV_LAN.g_language.AddGlobalLanguageFile('txtTestTranslations');     //Global Language File for Translations
```

Group-Specific Translation
A group can be assigned its own text list via the parameter `sTranslationFile` in `InitGroup()`. This overwrites the global file for strings within the scope of this group, enabling a modular, self-contained translation per group.

```
ExampleGrp.InitGroup(
    sName           := 'EXAMPLES',
    sTranslationFile:= 'txtExample',
    parent          := grp);
```

> **NOTE:** Namespaces: DAM = AS Data Manager, LAN = AS Language

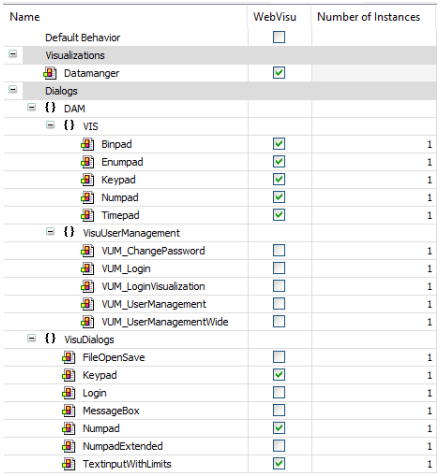
3 Visualization

The visualizations described in this chapter are open-source templates that can be used as-is or customized according to customer requirements.

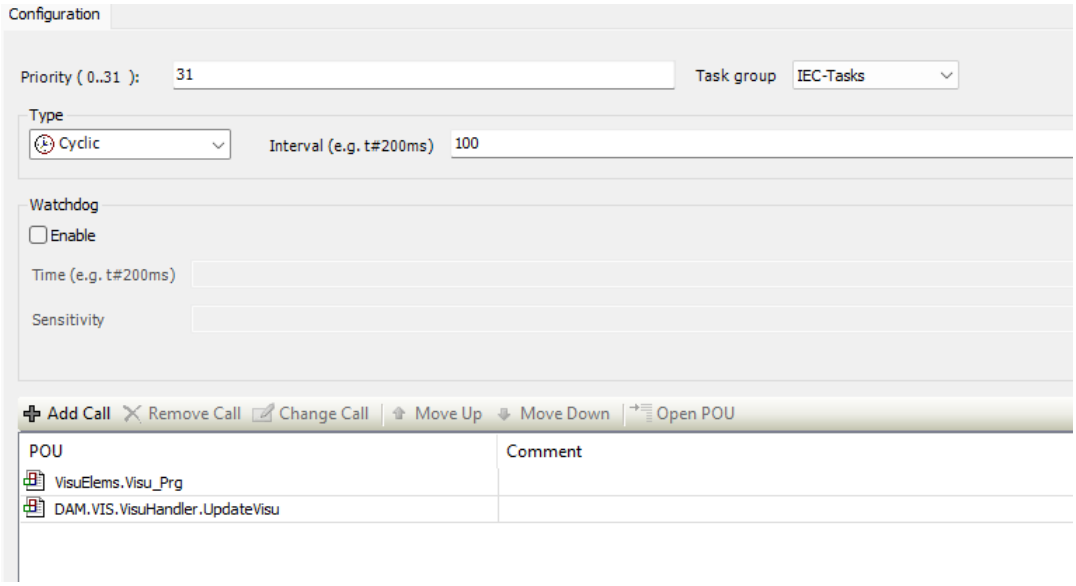
3.1 Preparations

Three configuration steps are required to use the DM visualization.

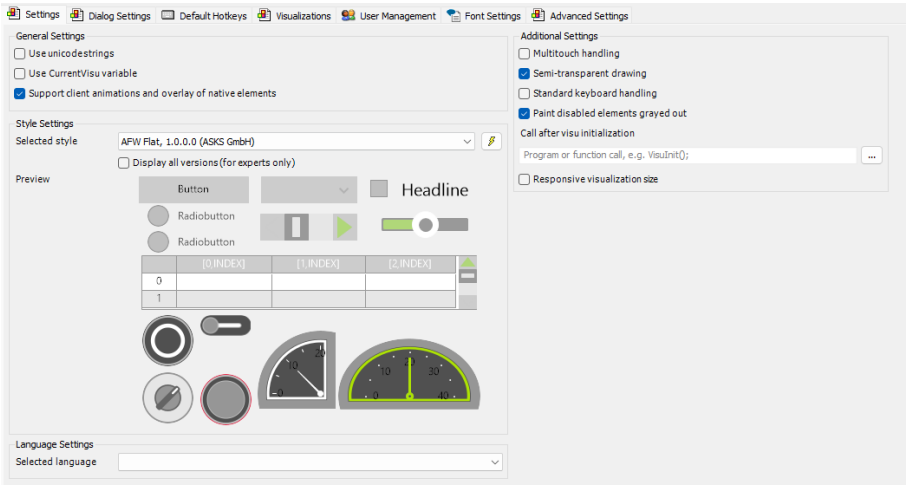
In the Visualization Manager, the option "Download all Dialogs" must be enabled, by checking all the needed Dialogs.



In the VISU_TASK, the updateVisu() method of the ASKS Visualization Handler must be called.

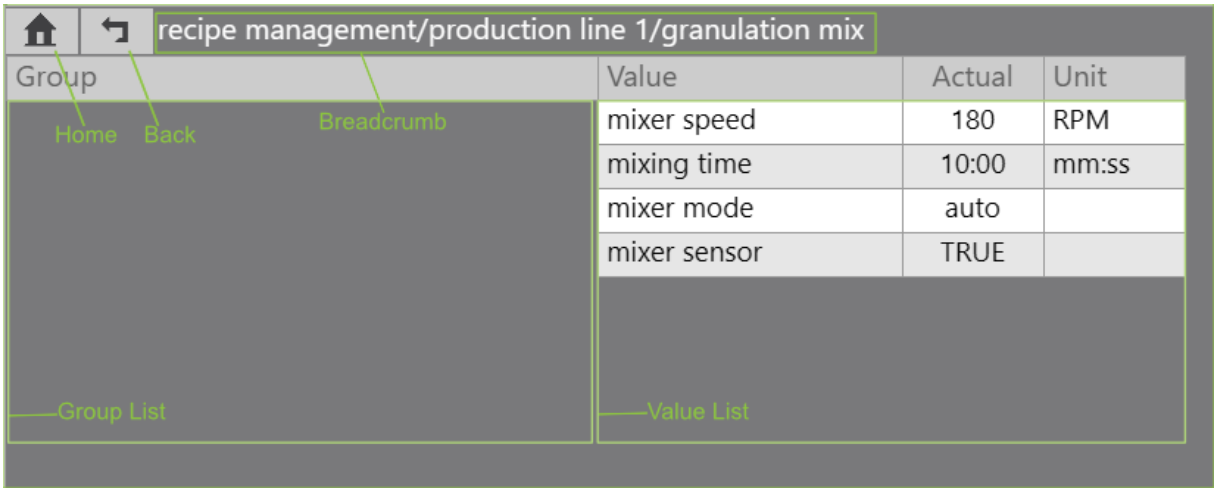


The visualization style "AFW Flat" must be selected in the Visualization Manager. Customizations to the visualizations are available from ASKS on request.

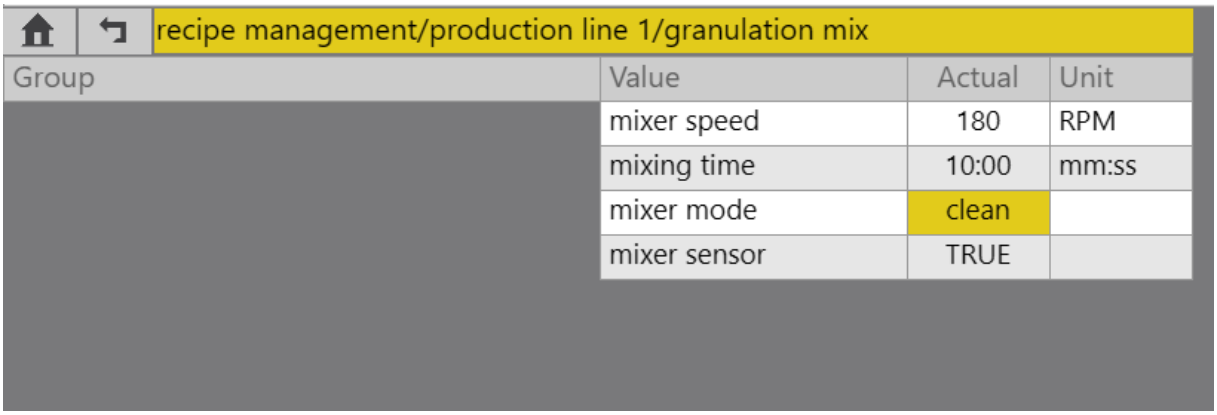


3.2 Data Record

To display a complete data record, the VisuDm visualization is instantiated in a frame and the desired group or Data Manager is passed. Groups and subgroups without values are hidden automatically.

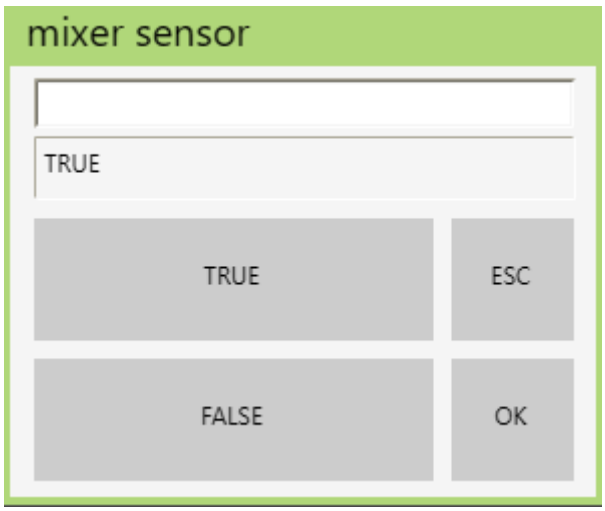


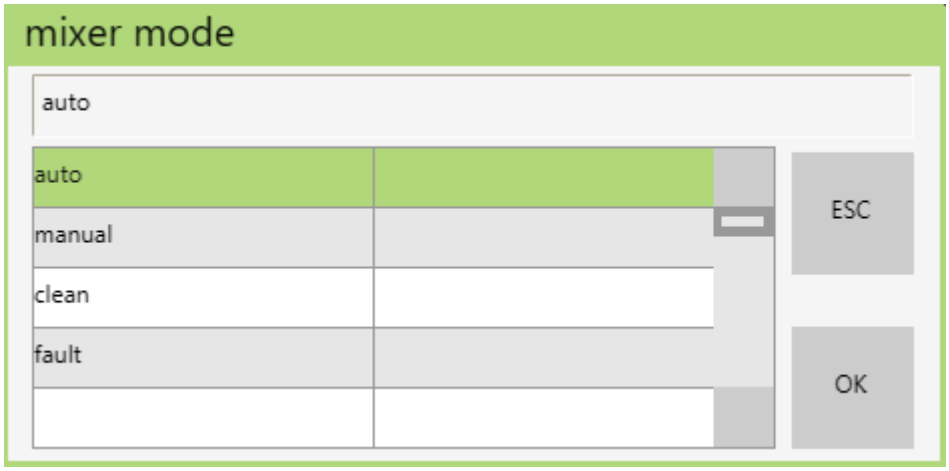
A yellow highlight indicates that there are unsaved changes in the data record.

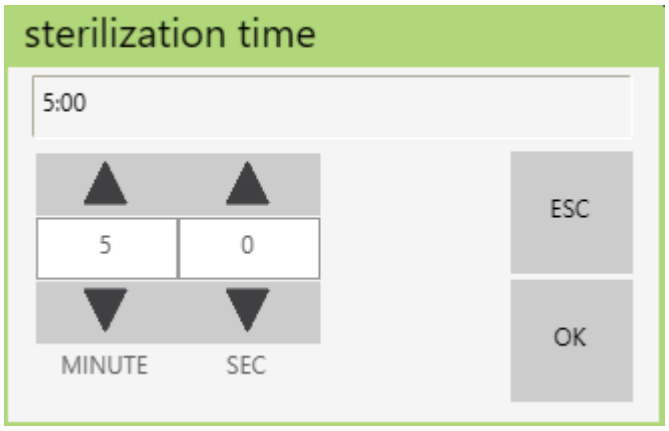


3.3 Values

In addition to the standard CODESYS input dialogs, the DM provides its own dialogs that simplify the input of specific values.







3.4 Translation

The Data Manager uses the CODESYS translation mechanism for group labels, value names, and units. Translation is configurable on a client-specific basis. In addition, a partial text resolution mechanism is integrated. To translate a substring, it is added to the respective group via grp.AddKey('').

3.4.1 Example: Partial Text Resolution

If the labels "STERILIZATION" and "CYCLE" are to be translated, a single entry of "STERILIZATION" and "CYCLE" in the translation file is sufficient to translate all strings containing that name.

ID	Default	de
STERILIZATION	sterilization	Sterilisierung
CYCLE	cycle	Zyklus

```
Steril.InitGroup(                       // Group - Sterilization
        sName               := 'STERILIZATION CYCLE',       // Group Name
        sTranslationFile    := 'txtTestTranslations',       // [] Translation file name
        parent              := Line1);                      // Parent Group
Steril.AddKey('STERILIZATION');
Steril.AddKey('CYCLE');

SterilTemp.Init(                        //Value - Sterilization Temperature
        group               := Steril,
        sName               := 'STERILIZATION TEMP',
        fMin                := 131,
        fDefault            := 185,
        fMax                := 275,
        sUnit               := 'FAH',
        nDecimals           := 2);

SterilTime.Init(                        //Value - Sterilization Time
        group               := Steril,
        sName               := 'STERILIZATION TIME',
        nDefault            := DAM.UTL.TimeFormat.Merge(nMin := 5),
        eFormat             := DAM.UTL.TIME_FORMAT.MINUTE_SEC);
```

> **NOTE:** ASKS GmbH uses English exclusively as the default ID. Uppercase IDs make untranslated entries immediately recognizable.

3.5 Unit Conversion

The REAL value of the Data Manager has built-in unit conversion functions. Various units are predefined and stored in the AS_System library. Additional units can be added on customer request.

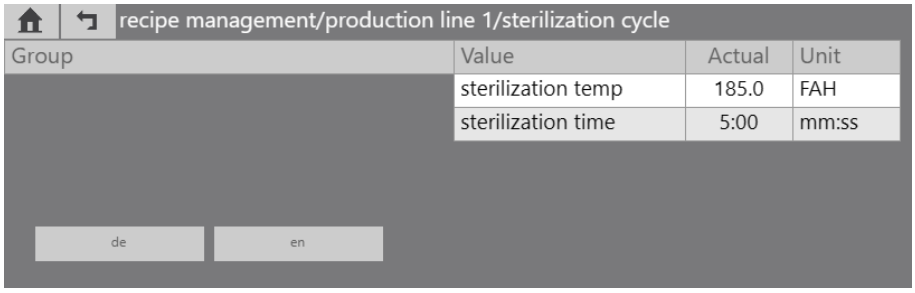
3.5.1 Example: Unit Conversion

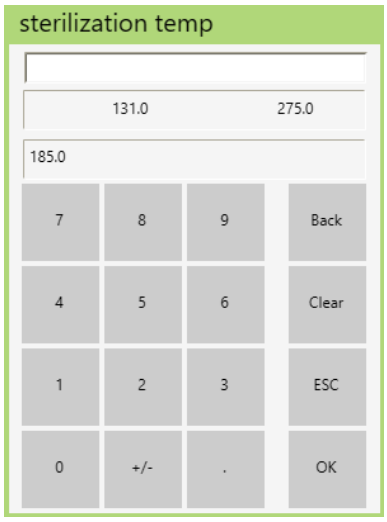
In an application, a temperature value in Fahrenheit is used; in the European market, Celsius is more common. The conversion is defined as a translation string in the format BASE<>TARGET, where BASE is the source unit and TARGET is the target unit.

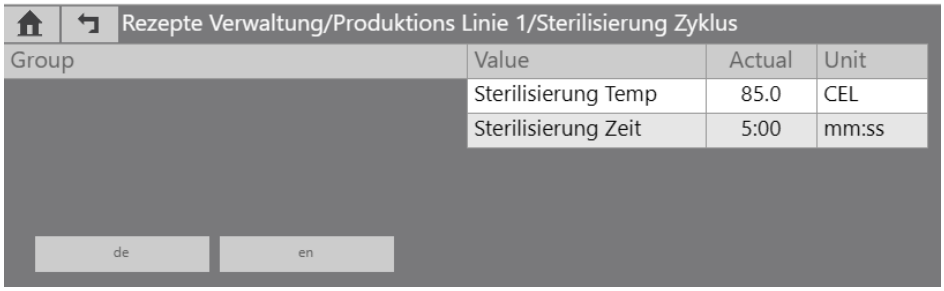
ID	Default	de
FAH	FAH	FAH<>CEL

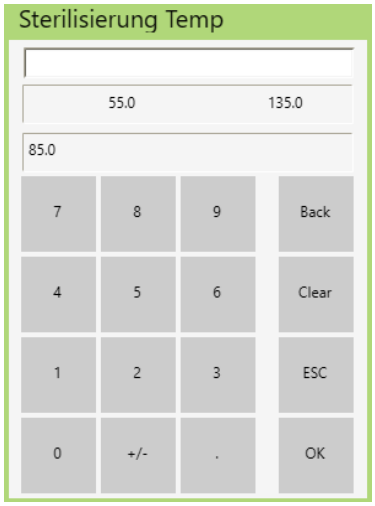
```
SterilTemp.Init(                        //Value - Sterilization Temperature
        group               := Steril,
        sName               := 'STERILIZATION TEMP',
        fMin                := 131,
        fDefault            := 185,
        fMax                := 275,
        sUnit               := 'FAH',
        nDecimals           := 2);
```

As soon as the corresponding language is switched to, the Data Manager converts the value automatically.









4 License

For information on licensing this library, please contact ASKS GmbH. Please reach us at support@as-ks.de or via our website www.as-ks.de.