

The logo for VLXE Thermodynamic Software. The letters 'V', 'L', 'X', and 'E' are rendered in a large, blue, 3D-style font with a gradient from light blue at the top to a darker blue at the bottom. Below the letters, the words 'THERMODYNAMIC SOFTWARE' are written in a smaller, blue, sans-serif font.

VLXE
THERMODYNAMIC SOFTWARE

VLXE SDK

© 2008 Dr. Torben Laursen

Preface

by Dr. Torben Laursen

This manual is for the VLXE SDK. It is intended for the users of VLXE who wants to use the VLXE dll in there own computer code.

Thanks for using VLXE!

VLXE SDK documentation

© 2008 Dr. Torben Laursen

All rights reserved. No parts of this work may be reproduced in any form or by any means - graphic, electronic, or mechanical, including photocopying, recording, taping, or information storage and retrieval systems - without the written permission of the publisher.

Products that are referred to in this document may be either trademarks and/or registered trademarks of the respective owners. The publisher and the author make no claim to these trademarks.

While every precaution has been taken in the preparation of this document, the publisher and the author assume no responsibility for errors or omissions, or for damages resulting from the use of information contained in this document or from the use of programs and source code that may accompany it. In no event shall the publisher and the author be liable for any loss of profit or any other commercial damage caused or alleged to have been caused directly or indirectly by this document.

Printed: October 2008 in Denmark

Publisher

VLXE ApS

Managing Editor

Dr. Torben Laursen

Technical Editors

*Dr. Torben Laursen
Lukas Andersen*

Contact Information:

VLXE ApS
Att. Torben Laursen
ScionDTU
Building 376, Diplomvej
2800 Lyngby
Denmark

e-mail: info@vlxe.com
Web: www.vlxe.com

Cell phone: + 45 27 20 02 16
Office phone: + 45 88 70 81 54
Fax.number: + 45 88 70 80 90

Table of Contents

Foreword	0
Part I Introduction	1
1 Contact VLXE	1
2 Functions	1
3 Units	2
4 How to perform a calculation	2
5 Compiler	2
Part II Error handling	3
1 ErrorMessage	3
2 ErrorGetIndex	3
Part III Model	4
1 Definition of a component	4
2 List of functions	4
3 Functions	6
Model group	6
ModelClear.....	6
ModelSetModelEOS.....	6
ModelSetModelCplgStandard.....	7
ModelSetModelCplgPolymer	7
ModelSetModelSLE	7
ModelSetModelAssociationMatrix.....	8
ModelSetModelAssociationMixingrule.....	8
Component group	8
ModelSetComponentType.....	9
ModelSetComponentIsAlkane.....	9
ModelSetComponentCriticalTemperature.....	9
ModelSetComponentCriticalPressure.....	10
ModelSetComponentCriticalVolume.....	10
ModelSetComponentAcentricFactor.....	11
ModelSetComponentHeatOfFormation.....	11
ModelSetComponentCplgPoly.....	11
ModelSetComponentCplgDIPPR.....	12
ModelSetComponentMonomerMolarMass.....	12
ModelSetComponentSLEc.....	12
ModelSetComponentSLEHu.....	13
ModelSetComponentSLEDc.....	13
ModelSetComponentSLEDa.....	14
ModelSetComponentSLETss.....	14
ModelSetComponentSLEHss.....	14

ModelSetComponentSLECPoly.....	15
ModelSetComponentPCSAFTdm.....	15
ModelSetComponentPCSAFTSigma.....	16
ModelSetComponentPCSAFTEpsilon.....	16
ModelSetComponentPCSAFTVolumeShift.....	16
ModelSetComponentAssociationUse.....	17
ModelSetComponentAssociationKappa.....	17
ModelSetComponentAssociationEpsilon.....	18
ModelSetComponentAssociationScheme.....	18
ModelSetComponentPolarUse.....	19
ModelSetComponentPolarx.....	19
ModelSetComponentPolarD.....	19
ModelSetComponentMSLdm.....	20
ModelSetComponentMSLv.....	20
ModelSetComponentMSLEpsilon.....	21
ModelSetComponentMSLVolumeShift.....	21
ModelSetComponentEOSQVolumeShift.....	21
ModelSetComponentBlockMassFraction.....	22
ModelSetComponentBlockName.....	22
ModelSetComponentSurfaceTensionIP.....	23
ModelSetComponentSurfaceTensionParachor.....	23
ModelSetComponentViscosityCritical.....	23
ModelSetComponentViscosityKz.....	24
Distribution group	24
ModelSetDistributionName.....	24
ModelSetDistributionPseudoName.....	25
ModelSetDistributionPseudoMeltingTemperature.....	25
ModelSetDistributionPseudoMolarMass.....	25
ModelSetDistributionPseudoMassFraction.....	26
Matrix group	26
ModelSetMatrixKijA.....	26
ModelSetMatrixKijB.....	27
ModelSetMatrixLijA.....	27
ModelSetMatrixLijB.....	27
ModelSetMatrixAssociationKappa.....	28
ModelSetMatrixAssociationEpsilon.....	28
ModelSetMatrixSurfaceTensionKij.....	29
4 Parameter overview	29
5 Association schemes	31
Part IV Calculations	31
1 List of functions	32
2 Functions	32
CalcSetStreamFeed	32
CalcGetTime	33
CalcBubbleP	33
CalcBubbleT	34
CalcDewP	34
CalcDewT	35
CalcFlashTP	36
CalcFlashPH	36

CalcFlashPS	37
CalcFlashTV	37
CalcFlashPV	38
CalcFlashTB	38
CalcFlashPB	39
CalcFlashTPFlow	39
Part V Results	40
1 List of functions	40
2 Overview of Properties	41
3 Functions	42
ResultGetNumberOfPoints	42
ResultGetArraySize	42
ResultGetPhaseType	43
ResultGetLabel	43
ResultGetPressure	44
ResultGetTemperature	44
ResultGetStablePhaseCount	45
ResultGetPhaseCount	45
ResultGetMoleFraction	46
ResultGetMassFraction	46
ResultGetLnK	46
ResultGetPhaseFractionMoleBased	47
ResultGetPhaseFractionMassBased	47
ResultGetVolume	48
ResultGetZ	48
ResultGetMolarMass	49
ResultGetDensity	49
ResultGetSolventMassFraction	50
ResultGetPolymerMassFraction	50
ResultGetEntropy	51
ResultGetCp	51
ResultGetCv	51
ResultGetEnthalpy	52
ResultGetJT	52
ResultGetSpeedOfSound	53
ResultGetThermalConductivity	53
ResultGetSurfaceTension	54
ResultGetViscosity	54
ResultGetMn	55
ResultGetMw	55
ResultGetMz	55
ResultGetBondingFraction	56
ResultGetSolubilityParameter	56
Part VI Properties	57
1 List of functions	57
2 Functions	58
PropUpdatePropertyTP	58
PropUpdateLnfTP	58

PropGetLnf	59
Part VII Examples	59
1 C++	60
2 C#	60
3 Fortran	60
Part VIII Background information	60
1 Ideal gas heat capacity	60
Part IX Appendix A: Version history	62
Index	63

1 Introduction

Welcome to the documentation for VLXE SDK!

This document describes how to use the VLXE dll's from other programming languages.

Example projects are provided for:

- C#
- C++
- Fortran

The SDK give access to all the models and calculations inside the dll.

The goal of this SDK has been to provide a simple yet powerful and fast interface. The argument types has been keep at a minimum to make it simpler to call the functions from a range of languages.

If you have any questions or suggestions on how to improve the SDK please [contact me](#).

1.1 Contact VLXE

If you need support or have any questions please contact VLXE. Please note that contact by e-mail is preferred

VLXE ApS
Dr. Torben Laursen
Diplomvej 376
Diplomvej
2800 Lyngby
Denmark

e-mail: info@vlxe.com
 Phone : +45 88 70 81 54
 Cell phone:+45 27 20 02 16
 Fax : +45 88 70 80 90
 Homepage: www.vlxe.com

1.2 Functions

The dll expose a number of functions that lets the call use all the functionality in the dll.

The functions are divided into 5 groups. Each function is prefixed with a group name to make it simpler for the user to see what group the function belongs to.

Nr	Group	Prefix	Description
1	Error handling	Error	These functions are to be used if any functions return a value other then -1. This indicate that a error occurred.
2	Model	Model	These functions lets the user define the model and all parameters used

3	Calculations	Calc	Lets the user call the calculations
4	Results	Result	Lets the user retrieve the results of a calculation
5	Properties	Prop	Lets the user call calculate a full range of properties

1.3 Units

The unit settings comes in 2 groups:

- Model parameters
- Calculation in- and out-put

Model parameters

The units for all the parameters are fixed and cannot be changed by the user.

Calculation in- and out-put

The units for all in and output are fixed and cannot be changed by the user.

Units	Default units
Composition	Massfraction
Temperature	[Kelvin]
Pressure	[Bar]
Volume	[cm ³]

1.4 How to perform a calculation

In order to perform a calculation 3 steps are needed:

- #1 [Write the model to the dll](#)
- #2 [Write the feed to the dll and call the calculation function](#)
- #3 [Retrieve the result from the dll](#)

Step 1 does not have to be repeated for each calculation.

So if several calculations like for example several flash calculations are to be done, step 1 only have to be done once.

1.5 Compiler

The SDK was written so the dll can be used from any language. It has been tested using C#, C++ and Fortran.

The C# and C++ compiler used is Visual Studio 2008

The Fortran compiler used is Intel 10.1 in Visual Studio 2008

The code can be used with other compilers without any problems.

Example projects are provided for:

- C#
- C++
- Fortran

2 Error handling

Error handling is a very important part of the interface.

The error handling is done using 2 steps

Step 1

Some functions return a int. If this int is equal to -1 no problems were detected.

If it was different from -1 a error was found and the caller needs to react.

Not all functions returns a errorindex. If a problem was found after calling these functions use `GetErrorIndex` to check if a problem was found.

Step 2

The function `GetErrorMessage` lets the caller retrieve a string with a error message to aid the user to pinpoint the problem.

The function `GetErrorIndex` lets the user retrieve a int that indicate the number of the error.

List if functions:

- [GetErrorMessage](#)
- [GetErrorIndex](#)

2.1 ErrorGetMessage

This function return a string with information on the latest found error.

The caller can use this information to locate and fix this problem.

This function is defined in C++ as:

```
extern "C" __declspec(dllexport) void __stdcall ErrorGetMessage(char *ErrorString)
```

Input

Argument	Type	Description
ErrorString	char pointer	Returns a error message if the function return a value different from -1. Use this information to see what is wrong.

2.2 ErrorGetIndex

This function return a int with information on the index of the latest found error.

The caller can use this information to locate and fix this problem.

This function is defined in C++ as:

```
extern "C" __declspec(dllexport) int __stdcall ErrorGetIndex(void)
```

Return

The function return the last found error index in the dll. Use `GetErrorMessage` to get more information

3 Model

This chapter describes how to set the thermodynamic model including all parameters. These functions all belong to the model group

3.1 Definition of a component

A component is a molecule in a mixture
VLXE can handle a range of different systems:

- Pure standard component
- Range of standard components
- Solvent(s)/Polymer
- Solvent(s)/copolymer

In order to provide a simple interface all components are seen as copolymer's. This is done since then all components can be defined as one type and the same interface can be used for all of them.

Examples:

Nr	Component	Number of blocks	Polydispersity
1	Methane	Mono	Mono
2	Water	Mono	Mono
3	HDPE with one pseudo component	Mono	Mono
4	HDPE with 100 pseudo components	Mono	100 pseudo components
5	EVA with on pseudo component	2 blocks	Mono
6	EVA with 50 pseudo components	2 blocks	50 pseudo components

HDPE: High density poly ethylene

EVA: Ethyl vinyl acetate

As it can be seen in the table, all components can be seen as a copolymer. For example methane is a copolymer with on block that is mono-disperse, while EVA is a copolymer with 2 blocks and a number of pseudo components.

3.2 List of functions

These are the functions used to write a model + parameters to the dll.

Each function is able of writing a single parameter to the dll.

The functions are divided into 4 groups:

- Model group: Defines the model settings
- Component group: Parameters for a component
- Distribution group: All distribution information
- Matrix group: All 2D matrix Example: kij's

Model group

- [ModelClear](#)
- [ModelSetModelEOS](#) □

- [ModelSetModelCplgStandard](#)
- [ModelSetModelCplgPolymer](#)
- [ModelSetModelSLE](#)
- [ModelSetModelAssociationMatrix](#)
- [ModelSetModelAssociationMixingrule](#)

Component group

- [ModelSetComponentType](#)
- [ModelSetComponentIsAlkane](#)
- [ModelSetComponentCriticalTemperature](#)
- [ModelSetComponentCriticalPressure](#)
- [ModelSetComponentCriticalVolume](#)
- [ModelSetComponentAcentricFactor](#)
- [ModelSetComponentHeatOfFormation](#)
- [ModelSetComponentCplgPoly](#)
- [ModelSetComponentCplgDIPPR](#)
- [ModelSetComponentMonomerMolarMass](#)
- [ModelSetComponentSLEc](#)
- [ModelSetComponentSLEHu](#)
- [ModelSetComponentSLEdc](#)
- [ModelSetComponentSLEda](#)
- [ModelSetComponentSLETss](#)
- [ModelSetComponentSLEHss](#)
- [ModelSetComponentSLEcPoly](#)
- [ModelSetComponentPCSAFTdm](#)
- [ModelSetComponentPCSAFTsigma](#)
- [ModelSetComponentPCSAFTEpsilon](#)
- [ModelSetComponentPCSAFTVolumeShift](#)
- [ModelSetComponentAssociationUse](#)
- [ModelSetComponentAssociationKappa](#)
- [ModelSetComponentAssociationEpsilon](#)
- [ModelSetComponentAssociationScheme](#)
- [ModelSetComponentPolarUse](#)
- [ModelSetComponentPolarx](#)
- [ModelSetComponentPolarD](#)
- [ModelSetComponentMSLdm](#)
- [ModelSetComponentMSLv](#)
- [ModelSetComponentMSLEpsilon](#)
- [ModelSetComponentMSLVolumeShift](#)
- [ModelSetComponentEOSQVolumeShift](#)
- [ModelSetComponentBlockMassFraction](#)
- [ModelSetComponentBlockName](#)
- [ModelSetComponentSurfaceTensionIP](#)
- [ModelSetComponentSurfaceTensionParachor](#)
- [ModelSetComponentViscosityCritical](#)
- [ModelSetComponentViscosityKz](#)

Distribution group

- [ModelSetDistributionName](#)
- [ModelSetDistributionPseudoName](#)
- [ModelSetDistributionPseudoMeltingTemperature](#)
- [ModelSetDistributionPseudoMolarMass](#)
- [ModelSetDistributionPseudoMassFraction](#)

Matrix group

- [ModelSetMatrixKijA](#)
- [ModelSetMatrixKijB](#)
- [ModelSetMatrixLijA](#)
- [ModelSetMatrixLijB](#)
- [ModelSetMatrixAssociationKappa](#)
- [ModelSetMatrixAssociationEpsilon](#)
- [ModelSetMatrixSurfaceTensionKij](#)

3.3 Functions

Enter topic text here.

3.3.1 Model group

Enter topic text here.

3.3.1.1 ModelClear

Call this function to clear the current model information in the dll

This function is defined in C++ as:

```
extern "C" __declspec(dllexport) int __stdcall ModelClear(void)
```

Return

The function return -1 if no problem was found. If a different number is returned use the function [ErrorGetMessage](#) for more information.

3.3.1.2 ModelSetModelEOS

This function is used to set the EOS to use.

This function is defined in C++ as:

```
extern "C" __declspec(dllexport) int __stdcall ModelSetModelEOS(char *Model)
```

Return

The function return -1 if no problem was found. If a different number is returned use the function [ErrorGetMessage](#) for more information.

Input

Argument	Type	Description
Model	char pointer	Definition for the selected model. Valid input are: <ul style="list-style-type: none"> • SRK • PR • SL • MSL • PC-SAFT • coPC-SAFT

3.3.1.3 ModelSetModelCpIgStandard

This function is used to write the model for the ideal gas heat capacity for standard(none polymer) components

This function is defined in C++ as:

```
extern "C" __declspec(dllexport) int __stdcall ModelSetModelCpIgStandard(
char *Model)
```

Return

The function return -1 if no problem was found. If a different number is returned use the function [ErrorMessage](#) for more information.

Input

Argument	Type	Description
Model	char pointer	Valid input: <ul style="list-style-type: none"> Polynomial DIPPR

3.3.1.4 ModelSetModelCpIgPolymer

This function is used to write the model for the ideal gas heat capacity for polymer components

This function is defined in C++ as:

```
extern "C" __declspec(dllexport) int __stdcall ModelSetModelCpIgPolymer(
char *Model)
```

Return

The function return -1 if no problem was found. If a different number is returned use the function [ErrorMessage](#) for more information.

Input

Argument	Type	Description
Model	char pointer	Valid input: <ul style="list-style-type: none"> Polynomial DIPPR

3.3.1.5 ModelSetModelSLE

This function is used to write the SLE model used for polymers to the dll

This function is defined in C++ as:

```
extern "C" __declspec(dllexport) int __stdcall ModelSetModelSLE(char
*Model)
```

Return

The function return -1 if no problem was found. If a different number is returned use the function [ErrorMessage](#) for more information.

Input

Argument	Type	Description
Model	char pointer	Valid input <ul style="list-style-type: none"> • Original Pan & Radosz • Full Pan & Radosz

3.3.1.6 ModelSetModelAssociationMatrix

Sets if the association matrix's are to be filled automatic or given by the user

This function is defined in C++ as:

```
extern "C" __declspec(dllexport) int __stdcall
ModelSetModelAssociationMatrix(char *Model)
```

Return

The function return -1 if no problem was found. If a different number is returned use the function [ErrorMessage](#) for more information.

Input

Argument	Type	Description
Model	char pointer	Valid input are: <ul style="list-style-type: none"> • Auto • User

3.3.1.7 ModelSetModelAssociationMixingrule

Sets mixing rule used in the association term by CPA.

This function is defined in C++ as:

```
extern "C" __declspec(dllexport) int __stdcall
ModelSetModelAssociationMixingrule(char *Model)
```

Return

The function return -1 if no problem was found. If a different number is returned use the function [ErrorMessage](#) for more information.

Input

Argument	Type	Description
Model	char pointer	Valid input are: <ul style="list-style-type: none"> • Auto (Same as CR1) • CR1 • ECR

3.3.2 Component group

Enter topic text here.

3.3.2.1 ModelSetComponentType

This defines the type of component.

Currently 2 types are supported:

- Standard
- Polymer

This function is defined in C++ as:

```
extern "C" __declspec(dllexport) int __stdcall ModelSetComponentType(int
ComponentIndex, int Value)
```

Return

The function return -1 if no problem was found. If a different number is returned use the function

[ErrorMessage](#) for more information.

Input

Argument	Type	Description
ComponetIndex	int	Index the component has in the total component list
Value	int	Type of component. Valid input: <ul style="list-style-type: none"> • 0 - Standard • 1 - Polymer

3.3.2.2 ModelSetComponentIsAlkane

This defines if the component is an alkane or not.

This function is defined in C++ as:

```
extern "C" __declspec(dllexport) int __stdcall ModelSetComponentIsAlkane(
int ComponentIndex, int Value)
```

Return

The function return -1 if no problem was found. If a different number is returned use the function

[ErrorMessage](#) for more information.

Input

Argument	Type	Description
ComponetIndex	int	Index the component has in the total component list
Value	int	Type of component. Valid input: <ul style="list-style-type: none"> • 0 - Alkane. For example Methane • 1 - None alkane. For example water

3.3.2.3 ModelSetComponentCriticalTemperature

Writes the critical temperature.

This function is defined in C++ as:

```
extern "C" __declspec(dllexport) int __stdcall
ModelSetComponentCriticalTemperature(int ComponentIndex, int BlockIndex,
double Value)
```

Return

The function return -1 if no problem was found. If a different number is returned use the function [ErrorMessage](#) for more information.

Input

Argument	Type	Description
ComponetIndex	int	Index the component has in the total component list
BlockIndex	int	The block the parameter is for
Value	double	Critical temperature. Units: Kelvin

3.3.2.4 ModelSetComponentCriticalPressure

Writes the critical pressure.

This function is defined in C++ as:

```
extern "C" __declspec(dllexport) int __stdcall
ModelSetComponentCriticalPressure(int ComponentIndex, int BlockIndex,
double Value)
```

Return

The function return -1 if no problem was found. If a different number is returned use the function [ErrorMessage](#) for more information.

Input

Argument	Type	Description
ComponetIndex	int	Index the component has in the total component list
BlockIndex	int	The block the parameter is for
Value	double	Critical pressure. Units: Bar

3.3.2.5 ModelSetComponentCriticalVolume

Writes the critical volume.

This function is defined in C++ as:

```
extern "C" __declspec(dllexport) int __stdcall
ModelSetComponentCriticalVolume(int ComponentIndex, int BlockIndex, double
Value)
```

Return

The function return -1 if no problem was found. If a different number is returned use the function [ErrorMessage](#) for more information.

Input

Argument	Type	Description
ComponetIndex	int	Index the component has in the total component list
BlockIndex	int	The block the parameter is for
Value	double	Critical volume. Units: cm3/mole

3.3.2.6 ModelSetComponentAcentricFactor

Writes the acentric factor.

This function is defined in C++ as:

```
extern "C" __declspec(dllexport) int __stdcall
ModelSetComponentAcentricFactor(int ComponentIndex, int BlockIndex, double
Value)
```

Return

The function return -1 if no problem was found. If a different number is returned use the function [ErrorMessage](#) for more information.

Input

Argument	Type	Description
ComponetIndex	int	Index the component has in the total component list
BlockIndex	int	The block the parameter is for
Value	double	Acentric factor. Units: -

3.3.2.7 ModelSetComponentHeatOfFormation

Writes heat of formation

This function is defined in C++ as:

```
extern "C" __declspec(dllexport) int __stdcall
ModelSetComponentHeatOfFormation(int ComponentIndex, int BlockIndex, double
Value)
```

Return

The function return -1 if no problem was found. If a different number is returned use the function [ErrorMessage](#) for more information.

Input

Argument	Type	Description
ComponetIndex	int	Index the component has in the total component list
BlockIndex	int	The block the parameter is for
Value	double	Heat of formation Units: [kJ/kg]

3.3.2.8 ModelSetComponentCplgPoly

Writes the parameters in the polynomial expression for the ideal gas heat capacity

This function is defined in C++ as:

```
extern "C" __declspec(dllexport) int __stdcall ModelSetComponentCpIgPoly(
int ComponentIndex, int BlockIndex, int SubIndex, double Value)
```

Return

The function return -1 if no problem was found. If a different number is returned use the function [ErrorMessage](#) for more information.

Input

Argument	Type	Description
----------	------	-------------

ComponetIndex	int	Index the component has in the total component list
BlockIndex	int	The block the parameter is for
SubIndex	int	Index in the Cp expression
Value	double	Parameter for the heat capacity Units: [kJ/kg]

3.3.2.9 ModelSetComponentCplgDIPPR

Writes the parameters in the DIPPR expression for the ideal gas heat capacity

This function is defined in C++ as:

```
extern "C" __declspec(dllexport) int __stdcall ModelSetComponentCpIgDIPPR(
int ComponentIndex, int BlockIndex, int SubIndex, double Value)
```

Return

The function return -1 if no problem was found. If a different number is returned use the function [ErrorGetMessage](#) for more information.

Input

Argument	Type	Description
ComponetIndex	int	Index the component has in the total component list
BlockIndex	int	The block the parameter is for
SubIndex	int	Index in the Cp expression
Value	double	Parameter for the heat capacity Units: [kJ/kg]

3.3.2.10 ModelSetComponentMonomerMolarMass

Writes the molar mass of the monomer.

This function is defined in C++ as:

```
extern "C" __declspec(dllexport) int __stdcall
ModelSetComponentMonomerMolarMass(int ComponentIndex, int BlockIndex,
double Value)
```

Return

The function return -1 if no problem was found. If a different number is returned use the function [ErrorGetMessage](#) for more information.

Input

Argument	Type	Description
ComponetIndex	int	Index the component has in the total component list
BlockIndex	int	The block the parameter is for
Value	double	Molar mass of the monomer Units: [g/mole]

3.3.2.11 ModelSetComponentSLEc

Writes the fraction of solid polymer that are in the crystal state.

This function is defined in C++ as:

```
extern "C" __declspec(dllexport) int __stdcall ModelSetComponentSLEc(int
ComponentIndex, int BlockIndex, double Value)
```

Return

The function return -1 if no problem was found. If a different number is returned use the function [ErrorMessage](#) for more information.

Input

Argument	Type	Description
ComponetIndex	int	Index the component has in the total component list
BlockIndex	int	The block the parameter is for
Value	double	Fraction of solid polymer in the crystal state. Units: [-]

3.3.2.12 ModelSetComponentSLEHu

Writes the enthalpy of melting for one mole of monomer

This function is defined in C++ as:

```
extern "C" __declspec(dllexport) int __stdcall ModelSetComponentSLEHu(int
ComponentIndex, int BlockIndex, double Value)
```

Return

The function return -1 if no problem was found. If a different number is returned use the function [ErrorMessage](#) for more information.

Input

Argument	Type	Description
ComponetIndex	int	Index the component has in the total component list
BlockIndex	int	The block the parameter is for
Value	double	Enthalpy of melting for one mole of monomer Units: [H/mole]

3.3.2.13 ModelSetComponentSLEdc

Writes the density of solid polymer in the crystal state

This function is defined in C++ as:

```
extern "C" __declspec(dllexport) int __stdcall ModelSetComponentSLEdc(int
ComponentIndex, int BlockIndex, double Value)
```

Return

The function return -1 if no problem was found. If a different number is returned use the function [ErrorMessage](#) for more information.

Input

Argument	Type	Description
ComponetIndex	int	Index the component has in the total component list
BlockIndex	int	The block the parameter is for
Value	double	Density of solid polymer in the crystal state Units: [g/cm3]

3.3.2.14 ModelSetComponentSLEDa

Writes the density of solid polymer in the amorphous state

This function is defined in C++ as:

```
extern "C" __declspec(dllexport) int __stdcall ModelSetComponentSLEDa(int
ComponentIndex, int BlockIndex, double Value)
```

Return

The function return -1 if no problem was found. If a different number is returned use the function [ErrorMessage](#) for more information.

Input

Argument	Type	Description
ComponetIndex	int	Index the component has in the total component list
BlockIndex	int	The block the parameter is for
Value	double	Density of solid polymer in the amorphous state Units: [g/cm3]

3.3.2.15 ModelSetComponentSLETss

Writes the temperature of the solid/solid transition for the solid polymer

This function is defined in C++ as:

```
extern "C" __declspec(dllexport) int __stdcall ModelSetComponentSLETss(int
ComponentIndex, int BlockIndex, double Value)
```

Return

The function return -1 if no problem was found. If a different number is returned use the function [ErrorMessage](#) for more information.

Input

Argument	Type	Description
ComponetIndex	int	Index the component has in the total component list
BlockIndex	int	The block the parameter is for
Value	double	Temperature of the solid/solid transition for the solid polymer Units: [Kelvin]

3.3.2.16 ModelSetComponentSLEHss

Writes the enthalpy of the solid/solid transition for the solid polymer

This function is defined in C++ as:

```
extern "C" __declspec(dllexport) int __stdcall ModelSetComponentSLEHss(int
ComponentIndex, int BlockIndex, double Value)
```

Return

The function return -1 if no problem was found. If a different number is returned use the function [ErrorMessage](#) for more information.

Input

Argument	Type	Description
ComponetIndex	int	Index the component has in the total component list
BlockIndex	int	The block the parameter is for
Value	double	Enthalpy of the solid/solid transition for the solid polymer Units: [J/mol]

3.3.2.17 ModelSetComponentSLECPoly

Writes the parameters in the polynomial expression for the heat capacity for the solid

This function is defined in C++ as:

```
extern "C" __declspec(dllexport) int __stdcall ModelSetComponentSLECPoly(
int ComponentIndex, int BlockIndex, int SubIndex, double Value)
```

Return

The function return -1 if no problem was found. If a different number is returned use the function [ErrorGetMessage](#) for more information.

Input

Argument	Type	Description
ComponetIndex	int	Index the component has in the total component list
BlockIndex	int	The block the parameter is for
SubIndex	int	Index in the Cp expression
Value	double	Parameter for the heat capacity Units: [kJ/kg]

3.3.2.18 ModelSetComponentPCSAFTdm

Writes the m parameter for PC-SAFT and copolymer PC-SAFT.

Note that the parameter is given as m/M.

This function is defined in C++ as:

```
extern "C" __declspec(dllexport) int __stdcall ModelSetComponentPCSAFTdm(
int ComponentIndex, int BlockIndex, double Value)
```

Return

The function return -1 if no problem was found. If a different number is returned use the function [ErrorGetMessage](#) for more information.

Input

Argument	Type	Description
ComponetIndex	int	Index the component has in the total component list
BlockIndex	int	The block the parameter is for
Value	double	m parameter for PC-SAFT and copolymer PC-SAFT Units: [mol/g]

3.3.2.19 ModelSetComponentPCSAFTSigma

Writes the Sigma parameter for PC-SAFT and copolymer PC-SAFT.

This function is defined in C++ as:

```
extern "C" __declspec(dllexport) int __stdcall
ModelSetComponentPCSAFTSigma(int ComponentIndex, int BlockIndex, double
Value)
```

Return

The function return -1 if no problem was found. If a different number is returned use the function [ErrorMessage](#) for more information.

Input

Argument	Type	Description
ComponetIndex	int	Index the component has in the total component list
BlockIndex	int	The block the parameter is for
Value	double	Sigma parameter for PC-SAFT and copolymer PC-SAFT Units: [A]

3.3.2.20 ModelSetComponentPCSAFTEpsilon

Writes the Epsilon parameter for PC-SAFT and copolymer PC-SAFT.

This function is defined in C++ as:

```
extern "C" __declspec(dllexport) int __stdcall
ModelSetComponentPCSAFTEpsilon(int ComponentIndex, int BlockIndex, double
Value)
```

Return

The function return -1 if no problem was found. If a different number is returned use the function [ErrorMessage](#) for more information.

Input

Argument	Type	Description
ComponetIndex	int	Index the component has in the total component list
BlockIndex	int	The block the parameter is for
Value	double	Epsilon parameter for PC-SAFT and copolymer PC-SAFT Units: [Kelvin]

3.3.2.21 ModelSetComponentPCSAFTVolumeShift

Writes the volume shift parameter for PC-SAFT and copolymer PC-SAFT.

This function is defined in C++ as:

```
extern "C" __declspec(dllexport) int __stdcall
ModelSetComponentPCSAFTVolumeShift(int ComponentIndex, int BlockIndex,
double Value)
```

Return

The function return -1 if no problem was found. If a different number is returned use the function [ErrorMessage](#) for more information.

Input

Argument	Type	Description
ComponetIndex	int	Index the component has in the total component list
BlockIndex	int	The block the parameter is for
Value	double	Volume shift parameter for PC-SAFT and copolymer PC-SAFT Units: [cm ³ /g]

3.3.2.22 ModelSetComponentAssociationUse

Turn the association term on and off.

This function is defined in C++ as:

```
extern "C" __declspec(dllexport) int __stdcall
ModelSetComponentAssociationUse(int ComponentIndex, int BlockIndex, int
Value)
```

Return

The function return -1 if no problem was found. If a different number is returned use the function [ErrorMessage](#) for more information.

Input

Argument	Type	Description
ComponetIndex	int	Index the component has in the total component list
BlockIndex	int	The block the parameter is for
Value	int	Turn the association term on and off. Valid input: <ul style="list-style-type: none"> • 0 : On • 1 : Off

3.3.2.23 ModelSetComponentAssociationKappa

Writes the kappa parameter used by the association term

This function is defined in C++ as:

```
extern "C" __declspec(dllexport) int __stdcall
ModelSetComponentAssociationKappa(int ComponentIndex, int BlockIndex,
double Value)
```

Return

The function return -1 if no problem was found. If a different number is returned use the function [ErrorMessage](#) for more information.

Input

Argument	Type	Description
ComponetIndex	int	Index the component has in the total component list
BlockIndex	int	The block the parameter is for
		Kappa parameter used by the association term

Value	double	Units: [-]
-------	--------	------------

3.3.2.24 ModelSetComponentAssociationEpsilon

Writes the epsilon parameter used by the association term

This function is defined in C++ as:

```
extern "C" __declspec(dllexport) int __stdcall
ModelSetComponentAssociationEpsilon(int ComponentIndex, int BlockIndex,
double Value)
```

Return

The function return -1 if no problem was found. If a different number is returned use the function [ErrorGetMessage](#) for more information.

Input

Argument	Type	Description
ComponetIndex	int	Index the component has in the total component list
BlockIndex	int	The block the parameter is for
Value	double	Epsilon parameter used by the association term Units: [-]

3.3.2.25 ModelSetComponentAssociationScheme

Writes association scheme used by the association term

This function is defined in C++ as:

```
extern "C" __declspec(dllexport) int __stdcall
ModelSetComponentAssociationScheme(int ComponentIndex, int BlockIndex, char
*ValueLabel, int Negative, int Neutral, int Positive)
```

Return

The function return -1 if no problem was found. If a different number is returned use the function [ErrorGetMessage](#) for more information.

Input

Argument	Type	Description
ComponetIndex	int	Index the component has in the total component list
BlockIndex	int	The block the parameter is for
Value	char pointer	Scheme used by the association term Valid input: See list here
Negative	int	Note: Is only used if Value=User Number of positive sites
Neutral	int	Note: Is only used if Value=User Number of neutral sites
Positive	int	Note: Is only used if Value=User Number of positive sites

3.3.2.26 ModelSetComponentPolarUse

Turn the polar term on and off.

This function is defined in C++ as:

```
extern "C" __declspec(dllexport) int __stdcall ModelSetComponentPolarUse(
int ComponentIndex, int BlockIndex, int Value)
```

Return

The function return -1 if no problem was found. If a different number is returned use the function [ErrorMessage](#) for more information.

Input

Argument	Type	Description
ComponetIndex	int	Index the component has in the total component list
BlockIndex	int	The block the parameter is for
Value	int	Turn the polar term on and off. Valid input: <ul style="list-style-type: none"> • 0 : On • 1 : Off

3.3.2.27 ModelSetComponentPolarx

Writes the x parameter for the polar term

This function is defined in C++ as:

```
extern "C" __declspec(dllexport) int __stdcall ModelSetComponentPolarx(int
ComponentIndex, int BlockIndex, double Value)
```

Return

The function return -1 if no problem was found. If a different number is returned use the function [ErrorMessage](#) for more information.

Input

Argument	Type	Description
ComponetIndex	int	Index the component has in the total component list
BlockIndex	int	The block the parameter is for
Value	double	x parameter for the polar term Units: [-]

3.3.2.28 ModelSetComponentPolarD

Writes the D parameter for the polar term

This function is defined in C++ as:

```
extern "C" __declspec(dllexport) int __stdcall ModelSetComponentPolarD(int
ComponentIndex, int BlockIndex, double Value)
```

Return

The function return -1 if no problem was found. If a different number is returned use the function [ErrorMessage](#) for more information.

Input

Argument	Type	Description
ComponetIndex	int	Index the component has in the total component list
BlockIndex	int	The block the parameter is for
Value	double	D parameter for the polar term Units: [D]

3.3.2.29 ModelSetComponentMSLdm

Writes the m parameter for SL or MSL EOS.

Note that the parameter is given as m/M.

This function is defined in C++ as:

```
extern "C" __declspec(dllexport) int __stdcall ModelSetComponentMSLdm(int
ComponentIndex, int BlockIndex, double Value)
```

Return

The function return -1 if no problem was found. If a different number is returned use the function [ErrorMessage](#) for more information.

Input

Argument	Type	Description
ComponetIndex	int	Index the component has in the total component list
BlockIndex	int	The block the parameter is for
Value	double	m parameter for SL or MSL Units: [mol/g]

3.3.2.30 ModelSetComponentMSLv

Writes the m parameter for SL or MSL EOS

This function is defined in C++ as:

```
extern "C" __declspec(dllexport) int __stdcall ModelSetComponentMSLv(int
ComponentIndex, int BlockIndex, double Value)
```

Return

The function return -1 if no problem was found. If a different number is returned use the function [ErrorMessage](#) for more information.

Input

Argument	Type	Description
ComponetIndex	int	Index the component has in the total component list
BlockIndex	int	The block the parameter is for
Value	double	v parameter for SL or MSL EOS Units: [cm3/mol]

3.3.2.31 ModelSetComponentMSLEpsilon

Writes the epsilon parameter for SL or MSL EOS

This function is defined in C++ as:

```
extern "C" __declspec(dllexport) int __stdcall ModelSetComponentMSLEpsilon(
int ComponentIndex, int BlockIndex, double Value)
```

Return

The function return -1 if no problem was found. If a different number is returned use the function [ErrorMessage](#) for more information.

Input

Argument	Type	Description
ComponetIndex	int	Index the component has in the total component list
BlockIndex	int	The block the parameter is for
Value	double	Epsilon parameter for SL or MSL EOS Units: [Kelvin]

3.3.2.32 ModelSetComponentMSLVolumeShift

Writes the volume shift parameter for SL or MSL EOS

This function is defined in C++ as:

```
extern "C" __declspec(dllexport) int __stdcall
ModelSetComponentMSLVolumeShift(int ComponentIndex, int BlockIndex, double
Value)
```

Return

The function return -1 if no problem was found. If a different number is returned use the function [ErrorMessage](#) for more information.

Input

Argument	Type	Description
ComponetIndex	int	Index the component has in the total component list
BlockIndex	int	The block the parameter is for
Value	double	Volume shift parameter for SL or MSL Units: [cm ³ /g]

3.3.2.33 ModelSetComponentEOSQVolumeShift

Writes the volume shift parameter for PR or SRK EOS

This function is defined in C++ as:

```
extern "C" __declspec(dllexport) int __stdcall
ModelSetComponentEOSQVolumeShift(int ComponentIndex, int BlockIndex, double
Value)
```

Return

The function return -1 if no problem was found. If a different number is returned use the function [ErrorMessage](#) for more information.

Input

Argument	Type	Description
ComponetIndex	int	Index the component has in the total component list
BlockIndex	int	The block the parameter is for
Value	double	Volume shift parameter for SRK or PR Units: [cm ³ /g]

3.3.2.34 ModelSetComponentBlockMassFraction

Writes the internal mass fraction of the block.

Note: For components that are not copolymer's this value will always be 1.

This function is defined in C++ as:

```
extern "C" __declspec(dllexport) int __stdcall
ModelSetComponentBlockMassFraction(int ComponentIndex, int BlockIndex,
double Value)
```

Return

The function return -1 if no problem was found. If a different number is returned use the function [ErrorMessage](#) for more information.

Input

Argument	Type	Description
ComponetIndex	int	Index the component has in the total component list
BlockIndex	int	The block the parameter is for
Value	double	Mass fraction of the block in the copolymer/ component Units: [-]

3.3.2.35 ModelSetComponentBlockName

Write the name of the block

Note this information is not needed by the calculations

This function is defined in C++ as:

```
extern "C" __declspec(dllexport) int __stdcall ModelSetComponentBlockName(
int ComponentIndex, int BlockIndex, char *ValueLabel)
```

Return

The function return -1 if no problem was found. If a different number is returned use the function [ErrorMessage](#) for more information.

Input

Argument	Type	Description
ComponetIndex	int	Index the component has in the total component list
BlockIndex	int	The block the parameter is for
ValueLabel	char pointer	Name of the block

3.3.2.36 ModelSetComponentSurfaceTensionIP

Writes the parameters in the polynomial expression for the surface tension influence parameter

This function is defined in C++ as:

```
extern "C" __declspec(dllexport) int __stdcall ModelSetComponentCpIgDIPPR(
int ComponentIndex, int BlockIndex, int SubIndex, double Value)
```

Return

The function return -1 if no problem was found. If a different number is returned use the function [ErrorMessage](#) for more information.

Input

Argument	Type	Description
ComponetIndex	int	Index the component has in the total component list
BlockIndex	int	The block the parameter is for
SubIndex	int	Index in the polynomial expression. Valid range: 0-2
Value	double	Parameters in the polynomial expression for the surface tension influence parameter Units: [J m ⁵ /mol ²]

3.3.2.37 ModelSetComponentSurfaceTensionParachor

Writes the parachor parameter used to calculate surface tension

This function is defined in C++ as:

```
extern "C" __declspec(dllexport) int __stdcall
ModelSetComponentSurfaceTensionParachor(int ComponentIndex, int
BlockIndex, double Value)
```

Return

The function return -1 if no problem was found. If a different number is returned use the function [ErrorMessage](#) for more information.

Input

Argument	Type	Description
ComponetIndex	int	Index the component has in the total component list
BlockIndex	int	The block the parameter is for
Value	double	Parachor parameter used to calculate surface tension Units: []

3.3.2.38 ModelSetComponentViscosityCritical

Writes the critical viscosity used by the f-theory to calculate viscosity.

This function is defined in C++ as:

```
extern "C" __declspec(dllexport) int __stdcall
ModelSetComponentViscosityCritical(int ComponentIndex, int BlockIndex,
double Value)
```

Return

The function return -1 if no problem was found. If a different number is returned use the function [ErrorMessage](#) for more information.

Input

Argument	Type	Description
ComponetIndex	int	Index the component has in the total component list
BlockIndex	int	The block the parameter is for
Value	double	Critical viscosity used by the f-theory to calculate viscosity. Units: [uP]

3.3.2.39 ModelSetComponentViscosityKz

Writes the Kz parameter used by the f-theory to calculate viscosity.

This function is defined in C++ as:

```
extern "C" __declspec(dllexport) int __stdcall
ModelSetComponentViscosityKz(int ComponentIndex, int BlockIndex, double
Value)
```

Return

The function return -1 if no problem was found. If a different number is returned use the function [ErrorMessage](#) for more information.

Input

Argument	Type	Description
ComponetIndex	int	Index the component has in the total component list
BlockIndex	int	The block the parameter is for
Value	double	Kz parameter used by the f-theory to calculate viscosity. Units: [-]

3.3.3 Distribution group

Enter topic text here.

3.3.3.1 ModelSetDistributionName

Writes the name of the distribution.

Note: This information is not used by the code.

This function is defined in C++ as:

```
extern "C" __declspec(dllexport) int __stdcall ModelSetDistributionName(int
ComponentIndex, char *ValueLabel)
```

Return

The function return -1 if no problem was found. If a different number is returned use the function [ErrorMessage](#) for more information.

Input

Argument	Type	Description
----------	------	-------------

ComponetIndex	int	Index the component has in the total component list
ValueLabel	char pointer	Name of the distribution

3.3.3.2 ModelSetDistributionPseudoName

Writes the name of the pseudo component.

Note: This information is not used by the code.

This function is defined in C++ as:

```
extern "C" __declspec(dllexport) int __stdcall
ModelSetDistributionPseudoName(int ComponentIndex, int PseudoIndex, char
*ValueLabel)
```

Return

The function return -1 if no problem was found. If a different number is returned use the function [ErrorGetMessage](#) for more information.

Input

Argument	Type	Description
ComponetIndex	int	Index the component has in the total component list
PseudoIndex	int	Index of the pseudo component.
ValueLabel	char pointer	Name of the pseudo component

3.3.3.3 ModelSetDistributionPseudoMeltingTemperature

Writes the melting temperature of the pseudo component.

Note: This information is only used in the SLE calculations.

This function is defined in C++ as:

```
extern "C" __declspec(dllexport) int __stdcall
ModelSetDistributionPseudoMeltingTemperature(int ComponentIndex, int
PseudoIndex, double Value)
```

Return

The function return -1 if no problem was found. If a different number is returned use the function [ErrorGetMessage](#) for more information.

Input

Argument	Type	Description
ComponetIndex	int	Index the component has in the total component list
PseudoIndex	int	Index of the pseudo component.
Value	double	Melting temperature of the pseudo component. Units: [Kelvin]

3.3.3.4 ModelSetDistributionPseudoMolarMass

Writes the molar mass of the pseudo component.

This function is defined in C++ as:

```
extern "C" __declspec(dllexport) int __stdcall
ModelSetDistributionPseudoMolarMass(int ComponentIndex, int PseudoIndex,
double Value)
```

Return

The function return -1 if no problem was found. If a different number is returned use the function [ErrorMessage](#) for more information.

Input

Argument	Type	Description
ComponetIndex	int	Index the component has in the total component list
PseudoIndex	int	Index of the pseudo component.
Value	double	Molar mass of the pseudo component. Units: [g/mole]

3.3.3.5 ModelSetDistributionPseudoMassFraction

Writes the mass fraction of the pseudo component.

Note: This information is only used in the SLE calculations.

This function is defined in C++ as:

```
extern "C" __declspec(dllexport) int __stdcall
ModelSetDistributionPseudoMassFraction(int ComponentIndex, int
PseudoIndex, double Value)
```

Return

The function return -1 if no problem was found. If a different number is returned use the function [ErrorMessage](#) for more information.

Input

Argument	Type	Description
ComponetIndex	int	Index the component has in the total component list
PseudoIndex	int	Index of the pseudo component.
Value	double	Mass fraction of the pseudo component. Units: [-]

3.3.4 Matrix group**3.3.4.1 ModelSetMatrixKijA**

Writes the a parameter for the kij matrix.

This function is defined in C++ as:

```
extern "C" __declspec(dllexport) int __stdcall ModelSetMatrixKijA(int i,
int j, double Value)
```

Return

The function return -1 if no problem was found. If a different number is returned use the function [ErrorMessage](#) for more information.

Input

Argument	Type	Description
i	int	First index in the 2D matrix
j	int	Second index in the 2D matrix
Value	double	a parameter for the kij matrix. Units: [-]

3.3.4.2 ModelSetMatrixKijB

Writes the b parameter for the kij matrix.

This function is defined in C++ as:

```
extern "C" __declspec(dllexport) int __stdcall ModelSetMatrixKijB(int i,
int j, double Value)
```

Return

The function return -1 if no problem was found. If a different number is returned use the function [ErrorMessage](#) for more information.

Input

Argument	Type	Description
i	int	First index in the 2D matrix
j	int	Second index in the 2D matrix
Value	double	a parameter for the kij matrix. Units: [Kelvin ⁻¹]

3.3.4.3 ModelSetMatrixLijA

Writes the a parameter for the lij matrix.

This function is defined in C++ as:

```
extern "C" __declspec(dllexport) int __stdcall ModelSetMatrixLijA(int i,
int j, double Value)
```

Return

The function return -1 if no problem was found. If a different number is returned use the function [ErrorMessage](#) for more information.

Input

Argument	Type	Description
i	int	First index in the 2D matrix
j	int	Second index in the 2D matrix
Value	double	a parameter for the lij matrix. Units: [-]

3.3.4.4 ModelSetMatrixLijB

Writes the b parameter for the lij matrix.

This function is defined in C++ as:

```
extern "C" __declspec(dllexport) int __stdcall ModelSetMatrixLijB(int i,
int j, double Value)
```

Return

The function return -1 if no problem was found. If a different number is returned use the function [ErrorMessage](#) for more information.

Input

Argument	Type	Description
i	int	First index in the 2D matrix
j	int	Second index in the 2D matrix
Value	double	b parameter for the lij matrix. Units: [Kelvin ⁻¹]

3.3.4.5 ModelSetMatrixAssociationKappa

Kappa matrix used by the association term.

This function is defined in C++ as:

```
extern "C" __declspec(dllexport) int __stdcall ModelSetMatrixLijB(int i,
int j, double Value)
```

Return

The function return -1 if no problem was found. If a different number is returned use the function [ErrorMessage](#) for more information.

Input

Argument	Type	Description
i	int	First index in the 2D matrix
j	int	Second index in the 2D matrix
Value	double	Kappa matrix used by the association term. Units: [-]

3.3.4.6 ModelSetMatrixAssociationEpsilon

Writes the Epsilon matrix used by the association term.

Note: This matrix is normally calculated automatic by the association term

This function is defined in C++ as:

```
extern "C" __declspec(dllexport) int __stdcall
ModelSetMatrixAssociationEpsilon(int i, int j, double Value)
```

Return

The function return -1 if no problem was found. If a different number is returned use the function [ErrorMessage](#) for more information.

Input

Argument	Type	Description
i	int	First index in the 2D matrix
j	int	Second index in the 2D matrix

Value	double	Epsilon matrix used by the association term. Units: [Kelvin]
-------	--------	---

3.3.4.7 ModelSetMatrixSurfaceTensionKij

Writes the kij parameter matrix used by the linear gradient theory.

Note: This matrix is normally calculated automatic by the association term

This function is defined in C++ as:

```
extern "C" __declspec(dllexport) int __stdcall
ModelSetMatrixSurfaceTensionKij(int i, int j, double Value)
```

Return

The function return -1 if no problem was found. If a different number is returned use the function [ErrorGetMessage](#) for more information.

Input

Argument	Type	Description
i	int	First index in the 2D matrix
j	int	Second index in the 2D matrix
Value	double	kij parameter matrix used by the linear gradient theory. Units: [-]

3.4 Parameter overview

The table below gives a overview over all the model and parameter settings which is available. The table below lists all options + the units used for each

Nr	ValueType	Units	Type	Description
	Component: Name	[-]	String	Name of the component
1	Component: Type	[-]	numerical	Sets the type of component. Options for Value: 0: Standard component 1: Polymer
2	Component: IsAlkane	[-]	Numeric	Alkane or non-Alkane. Options for Value: 0: Alkane 1: non-Alkane
3	Component: DBIndex	[-]	Numeric	Internal use. Indicate the VLXE database index
5	Standard: Tc	[Kelvin]	Numeric	Critical temperature
6	Standard: Pc	[Bar]	Numeric	Critical pressure
7	Standard: Vc	[cm ³ /mol]	Numeric	Critical volume
8	Standard: AcentricFactor	[-]	Numeric	Acentric factor
9	Standard: HeatOfFormation	[kJ/kg]	Numeric	Heat of formation for the component
10	CpIGPoly	[kJ/kg]	Numeric	Ideal gas Cp using a polynomial expression
11	CpIGDIPPR	[kJ/kg]	Numeric	Ideal gas Cp using a DIPPR expression
12	Monomer: Name	[-]	String	Name of the monomer that makes

				the block
13	Monomer: M	[g/mole]	Numeric	Molar mass of the monomer that makes the block
14	SLE: c	[-]	Numeric	Crystallinity fraction. Fraction of solid polymer that is in crystal form.
15	SLE: Hu	[J/mol]	Numeric	Enthalpy of melting of one mole of monomer
16	SLE: Dc	[g/cm ³]	Numeric	Density of crystalline polymer
17	SLE: Da	[g/cm ³]	Numeric	Density of amorphous polymer
18	SLE: Tss	[Kelvin]	Numeric	Solid/Solid transition temperature
19	SLE: Hss	[J/mol]	Numeric	Enthalpy of solid/solid transition
20	SLE: CpPoly	[kJ/(kg Kelvin)]	Numeric	Heat capacity of solid polymer
21	PCSAFT: dm	[mol/g]	Numeric	PC-SAFT parameter
22	PCSAFT: Sigma	[Å]	Numeric	PC-SAFT parameter
23	PCSAFT: Epsilon	[Kelvin]	Numeric	PC-SAFT parameter
24	PCSAFT: VolumeShift	[cm ³ /g]	Numeric	PC-SAFT parameter.
25	Association: Use	[-]	Numeric	Turn association term on and off. Valid input: 0: On 1: Off
26	Association: Kappa	[-]	Numeric	Association term parameter
27	Association: Epsilon	[Kelvin]	Numeric	Association term parameter
28	Association: Scheme	[-]	String	Association term parameter
29	Polar: Use	[-]	Numeric	Turn polar term on and off. Valid input: 0: On 1: Off
30	Polar: x	[-]	Numeric	Polar term parameter
31	Polar: D	[D]	Numeric	Polar term parameter
32	MSL: dm	[mole/g]	Numeric	SL/MSL parameter
33	MSL: v	[cm ³ /mole]	Numeric	SL/MSL parameter
34	MSL: Epsilon	[Kelvin]	Numeric	SL/MSL parameter
35	MSL: VolumeShift	[cm ³ /g]	Numeric	SL/MSL parameter
36	EOSQ: VolumeShift	[cm ³ /g]	Numeric	PR/SRK parameter
37	Block: DBIndex	[-]	Numeric	Internal use. Indicate the VLXE database index for the block.
38	Block: MassFraction	[-]	Numeric	Massfraction of the block within the copolymer/component
39	Block: Name	[-]	String	Name of the block within the copolymer/component
40	SurfaceTension: IP	[J m ⁵ /mol ²]	Numeric	Influence parameter used in surface tension
41	SurfaceTension: Parachor	[]	Numeric	W/K surface tension model parameter
42	Viscosity: Critical	[uP]	Numeric	f-theory viscosity parameter
43	Viscosity: Kz	[-]	Numeric	f-theory viscosity parameter

3.5 Association schemes

The association term is used in PC-SAFT and copolymer PC-SAFT. It allows the model to handle more complex mixtures since it will take cross association into account.

A large numbers of schemes are implemented into VLXE

Nr	Scheme	Description
1	1A	1 neutral site
2	1C	1 negative site (active only when mixed with a molecule with at least one positive site)
3	1D	1 positive site (active only when mixed with a molecule with at least one negative site)
4	2A	2 neutral site
5	2B	1 negative site and 1 positive site
6	2C	2 negative sites (active only when mixed with a molecule with at least one positive site)
7	2D	2 positive sites (active only when mixed with a molecule with at least one negative site)
8	3A	3 neutral sites
9	3B	2 positive sites and 1 negative site
10	4A	4 neutral sites
11	4B	3 positive sites and one negative site
12	4C	2 positive sites and 2 negative sites
13	User	User can define a scheme. The maximum number of sites allowed is 4

The user can combine the schemes in any way which is possible.

The schemes 1C, 1D, 2C and 2D are for components that does not self-associate but is able to associate with other components like water or methanol.

The most common case is polar components like acetone.

However this pose the problem of obtaining the parameters needed for e.g. acetone since it cannot be determined by fitting to pure component data.

To solve this problem the "Consortium of complex fluid" has suggested that the kappa parameter is taken from the other component while the epsilon is set to zero.

The standard mixing rules can then be used.

This method may not always give the best result and VLXE therefore support user given parameters where no mixing rules are used.

The user therefore now has to select either automatic or user.

If "Automatic" is selected the standard mixing rules are used and the user has to give AssociationkappaAB and epsilonAB for each component.

If "User" is selected no mixing rules are used and the user has to give both the kappaAB and epsilonAB matrix.

For more information please see the reference: 1

4 Calculations

This chapter describes how to call the calculations in the dll

All functions belong to the calculation group

There are 3 steps needed.
 #1 Change the units in the dll if needed
 #2 Write the feed to the dll
 #3 Call the calculation function.

4.1 List of functions

These are the functions used to call the calculations

Support functions

- [CalcSetStreamFeed](#)
- [CalcGetTime](#)

Calculation functions

- [CalcBubbleP](#)
- [CalcBubbleT](#)
- [CalcDewP](#)
- [CalcDewT](#)
- [CalcFlashTP](#)
- [CalcFlashPH](#)
- [CalcFlashPS](#)
- [CalcFlashTV](#)
- [CalcFlashPV](#)
- [CalcFlashTB](#)
- [CalcFlashPB](#)
- [CalcFlashTPFlow](#)

Click on each for more information

4.2 Functions

Enter topic text here.

4.2.1 CalcSetStreamFeed

This function is used to define the composition used in a calculation.

Note that the units expected by the dll can be altered by using SetUnits

This function is used to write all the distribution of a component to the dll

This function is defined in C++ as:

```
extern "C" __declspec(dllexport) int __stdcall CalcSetStreamFeed(int
Stream, int ComponentIndex, double Value)
```

Return

The function return -1 if no problem was found. If a different number is returned check the argument ErrorString for more information.

Input

Argument	Type	Description
Stream	int	Index of the stream. Valid range are 0-9

ComponentIndex	int	Component index where the feed value is to be written
Value	double	Feed value.

Code examples

Example

4.2.2 CalcGetTime

Function returns the time a calculation took i seconds

This function is defined in C++ as:

```
extern "C" __declspec(dllexport) double __stdcall CalcGetTime(void)
```

Return

The function return the time a calculation took in seconds.

Code examples

Example

4.2.3 CalcBubbleP

This function perform a bubble pressure calculation.

The function works for both polymer and non-polymer systems and can handle both VLE and LLE points.

This function is defined in C++ as:

```
extern "C" __declspec(dllexport) int __stdcall CalcBubbleP(double Temperature, int PointType, double MinimumPressure, double MaximumPressure)
```

Return

The function return -1 if no problem was found. If a different number is returned use the function [ErrorMessage](#) for more information.

Input

Argument	Type	Units	Description
Temperature	double	User defined Default: Kelvin	Temperature to use in calculation
PointType	int	[-]	Type of point to look for. Valid input are: <ul style="list-style-type: none"> • 0 (Auto) • 1 (LLE) • 2 (VLE)
MinimumPressure	double	User defined Default: Bar	Note: Polymer systems only. Smallest possible bubble pressure to look for.
MaximumPressure	double	User defined Default: Bar	Note: Polymer systems only. Biggest possible bubble pressure to look for. Note that the value can go outside the physical range.

Code examples

Example

4.2.4 CalcBubbleT

This function perform a bubble temperature calculation.

The function works for both polymer and non-polymer systems and can handle both VLE and LLE points.

This function is defined in C++ as:

```
extern "C" __declspec(dllexport) int __stdcall CalcBubbleT(double
Pressure, int PointType, double MinimumTemperature, double
MaximumTemperature)
```

Return

The function return -1 if no problem was found. If a different number is returned use the function [ErrorMessage](#) for more information.

Input

Argument	Type	Units	Description
Pressure	double	User defined Default: Bar	Pressure to use in calculation
PointType	int	[-]	Type of point to look for. Valid input are: <ul style="list-style-type: none"> • 0 (Auto) • 1 (LLE) • 2 (VLE)
Minimum Temperature	double	User defined Default: Kelvin	Note: Polymer systems only. Smallest possible bubble temperature to look for.
Maximum Temperature	double	User defined Default: Kelvin	Note: Polymer systems only. Biggest possible bubble temperature to look for. Note that the value can go outside the physical range.

Code examples

Example

4.2.5 CalcDewP

This function perform a dew pressure calculation.

The function works only for non-polymer systems and can handle only VLE points.

This function is defined in C++ as:

```
extern "C" __declspec(dllexport) int __stdcall CalcDewP(double
Temperature, int PointType, double MinimumPressure, double
MaximumPressure)
```

Return

The function return -1 if no problem was found. If a different number is returned use the function [ErrorMessage](#) for more information.

Input

Argument	Type	Units	Description
Temperature	double	User defined Default: Kelvin	Temperature to use in calculation
PointType	int	[-]	Type of point to look for. Valid input are: <ul style="list-style-type: none"> • 0 (Auto) • 1 (LLE) • 2 (VLE)
MinimumPressure	double	User defined Default: Bar	Note: Polymer systems only. Smallest possible bubble pressure to look for.
MaximumPressure	double	User defined Default: Bar	Note: Polymer systems only. Biggest possible bubble pressure to look for. Note that the value can go outside the physical range.

Code examples

Example

4.2.6 CalcDewT

This function perform a dew temperature calculation.

The function works only for non-polymer systems and can handle only VLE points.

This function is defined in C++ as:

```
extern "C" __declspec(dllexport) int __stdcall CalcDewT(double Pressure,
int PointType, double MinimumTemperature, double MaximumTemperature)
```

Return

The function return -1 if no problem was found. If a different number is returned use the function [ErrorMessage](#) for more information.

Input

Argument	Type	Units	Description
Pressure	double	User defined Default: Bar	Temperature to use in calculation
PointType	int	[-]	Type of point to look for. Valid input are: <ul style="list-style-type: none"> • 0 (Auto) • 1 (LLE) • 2 (VLE)
Minimum Temperature	double	User defined Default: Kelvin	Note: Polymer systems only. Smallest possible bubble temperature to look for.
Maximum Temperature	double	User defined Default: Kelvin	Note: Polymer systems only. Biggest possible bubble temperature to look for. Note that the value can go outside the physical range.

Code examples

Example

4.2.7 CalcFlashTP

Function that perform a multiphase flash at fixed temperature and pressure

This function is defined in C++ as:

```
extern "C" __declspec(dllexport) int __stdcall CalcFlashTP(double Temperature, double Pressure, int PhaseCount)
```

Return

The function return -1 if no problem was found. If a different number is returned use the function [ErrorMessage](#) for more information.

Input

Argument	Type	Units	Description
Temperature	double	User defined Default: [Kelvin]	Temperature to use in calculation
Pressure	double	User defined Default: [Bar]	Pressure to use in the calculation
PhaseCount	int	[-]	Number of phases the flash will look for. Valid input are: 2: The flash will maximum look for 2 phases 3: The flash will maximum look for 3 phases 0: No limit on the number of phases to look for

Code examples

Example

4.2.8 CalcFlashPH

Function that perform a multiphase flash at fixed pressure and enthalpy

This function is defined in C++ as:

```
extern "C" __declspec(dllexport) int __stdcall CalcFlashPH(double Pressure, double Enthalpy, int PhaseCount)
```

Return

The function return -1 if no problem was found. If a different number is returned use the function [ErrorMessage](#) for more information.

Input

Argument	Type	Units	Description
Pressure	double	User defined Default: [Bar]	Pressure to use in calculation
Enthalpy	double	User defined Default: [kJ/kg]	Enthalpy to use in the calculation
PhaseCount	int	[-]	Number of phases the flash will look for. Valid input are: 2: The flash will maximum look for 2 phases 3: The flash will maximum look for 3 phases 0: No limit on the number of phases to look for

Code examples

Example

4.2.9 CalcFlashPS

Function that perform a multiphase flash at fixed pressure and entropy

This function is defined in C++ as:

```
extern "C" __declspec(dllexport) int __stdcall CalcFlashPS(double
Pressure, double Entropy, int PhaseCount)
```

Return

The function return -1 if no problem was found. If a different number is returned use the function [ErrorMessage](#) for more information.

Input

Argument	Type	Units	Description
Pressure	double	User defined Default: [Bar]	Pressure to use in calculation
Entropy	double	User defined Default: [kJ/(kg kelvin)]	Entropy to use in the calculation
PhaseCount	int	[-]	Number of phases the flash will look for. Valid input are: 2: The flash will maximum look for 2 phases 3: The flash will maximum look for 3 phases 0: No limit on the number of phases to look for

Code examples

Example

4.2.10 CalcFlashTV

Function that perform a multiphase flash at fixed temperature and volume

This function is defined in C++ as:

```
extern "C" __declspec(dllexport) int __stdcall CalcFlashTV(double
Temperature, double Volume, int PhaseCount)
```

Return

The function return -1 if no problem was found. If a different number is returned use the function [ErrorMessage](#) for more information.

Input

Argument	Type	Units	Description
Temperature	double	User defined Default: [Kelvin]	Temperature to use in calculation
Volume	double	User defined Default: [cm3]	Volume to use in the calculation
PhaseCount	int	[-]	Currently this flash type can only handle 2 phases. Argument is unused.

Code examples

Example

4.2.11 CalcFlashPV

Function that perform a multiphase flash at fixed pressure and volume

This function is defined in C++ as:

```
extern "C" __declspec(dllexport) int __stdcall CalcFlashPV(double
Pressure, double Volume, int PhaseCount)
```

Return

The function return -1 if no problem was found. If a different number is returned use the function [ErrorMessage](#) for more information.

Input

Argument	Type	Units	Description
Pressure	double	User defined Default: [Kelvin]	Pressure to use in calculation
Volume	double	User defined Default: [cm3]	Volume to use in the calculation
PhaseCount	int	[-]	Currently this flash type can only handle 2 phases. Argument is unused.

Code examples

Example

4.2.12 CalcFlashTB

Function that perform a multiphase flash at fixed temperature and phase fraction (Mole based)

This function is defined in C++ as:

```
extern "C" __declspec(dllexport) int __stdcall CalcFlashTB(double
Temperature, double PhaseFraction, int PhaseCount)
```

Return

The function return -1 if no problem was found. If a different number is returned use the function [ErrorMessage](#) for more information.

Input

Argument	Type	Units	Description
Temperature	double	User defined Default: [Kelvin]	Temperature to use in calculation
PhaseFraction	double	User defined Default: [-] Note: Phase fraction is mole based.	Phase fraction to use in the calculation
PhaseCount	int	[-]	Currently this flash type can only handle 2 phases. Argument is unused.

Code examples

Example

4.2.13 CalcFlashPB

Function that perform a multiphase flash at fixed pressure and phase fraction (Mole based)

This function is defined in C++ as:

```
extern "C" __declspec(dllexport) int __stdcall CalcFlashPB(double
Pressure, double PhaseFraction, int PhaseCount)
```

Return

The function return -1 if no problem was found. If a different number is returned use the function [ErrorMessage](#) for more information.

Input

Argument	Type	Units	Description
Pressure	double	User defined Default: [Bar]	Pressure to use in calculation
PhaseFraction	double	User defined Default: [-] Note: Phase fraction is mole based.	Phase fraction to use in the calculation
PhaseCount	int	[-]	Currently this flash type can only handle 2 phases. Argument is unused.

Code examples

Example

4.2.14 CalcFlashTPFlow

Function that perform a multiphase flash at fixed temperature and pressure and has the added feature compared to CalcFlashTP that is will reuse the last flash result in order to make the flash faster.

This function is provided to user that has a need to perform a large number of flash's where the step between each flash is small, either in temperature, pressure or feed. A example can be a flow simulator or other kind of simulators.

In order to use the function simply call it. The function will then check if a result is available. If a result is present it will skip the stability analysis and converge the given system. If it fail's to converge or no result is present it will perform a full flash.

Since no extrapolation is currently used it is important that step size is small. The users will have to try them self in order to find the optimal step size.

This function is defined in C++ as:

```
extern "C" __declspec(dllexport) int __stdcall CalcFlashTPFlow(double
Temperature, double Pressure, int PhaseCount)
```

Return

The function return -1 if no problem was found. If a different number is returned use the function [ErrorMessage](#) for more information.

Input

Argument	Type	Units	Description
Temperature	double	User defined	Temperature to use in calculation

		Default: [Kelvin]	
Pressure	double	User defined Default: [Bar]	Pressure to use in the calculation
PhaseCount	int	[-]	Number of phases the flash will look for. Valid input are: 2: The flash will maximum look for 2 phases 3: The flash will maximum look for 3 phases 0: No limit on the number of phases to look for

Code examples

Example

5 Results

The results from VLXE are all arranged in the same way. This is done so the interface to the results can be handle by the same functions no matter the calculation the user want to perform.

All results are seen as a series of point where each point can have a number of phases. The phases are arranged as:

- System
- Feed
- Phase 1
- Phase 2
- etc.

For example the result of a cloud point will have one point and 4 phases. A phase envelope will have a vary number of points and each point will have 4 phases. A flash will have one point and a number of phases where 3 is minimum.

5.1 List of functions

These are the functions used to get the result of a calculations. All functions belong to the result group.

Result functions

- [ResultGetNumberOfPoints](#)
- [ResultGetArraySize](#)
- [ResultGetPhaseType](#)
- [ResultGetLabel](#)
- [ResultGetPressure](#)
- [ResultGetTemperature](#)
- [ResultGetStablePhaseCount](#)
- [ResultGetPhaseCount](#)
- [ResultGetMoleFraction](#)
- [ResultGetMassFraction](#)
- [ResultGetLnK](#)
- [ResultGetPhaseFractionMoleBased](#)
- [ResultGetPhaseFractionMassBased](#)
- [ResultGetVolume](#)

- [ResultGetZ](#)
- [ResultGetMolarMass](#)
- [ResultGetDensity](#)
- [ResultGetPolymerMassFraction](#)
- [ResultGetSolventMassFraction](#)
- [ResultGetEntropy](#)
- [ResultGetCp](#)
- [ResultGetCv](#)
- [ResultGetEnthalpy](#)
- [ResultGetJT](#)
- [ResultGetSpeedOfSound](#)
- [ResultGetThermalConductivity](#)
- [ResultGetSurfaceTension](#)
- [ResultGetViscosity](#)
- [ResultGetMn](#)
- [ResultGetMw](#)
- [ResultGetMz](#)
- [ResultGetBondingFraction](#)
- [ResultGetSolubilityParameter](#)

Click on each for more information

5.2 Overview of Properties

A extensive range of properties can be retrieved from the dll

Point properties

Name	Default Units	Description
NumberOfPoints	[-]	Number of points in the result
ArraySize	[-]	Size of all the component arrays (equal to the number of components)
Temperature	[Kelvin]	Temperature for the point
Pressure	[Bar]	Pressure for the point

Phase properties

Name	Default Units	Description
NumberOfPhases	[-]	Total number of phases: System + Feed + Number of stable phases
NumberOfStablePhases	[-]	Number of stable phases found
Molefraction	[-]	Molefraction for a component
Molefraction	[-]	Massfraction for a component
LnK	[-]	LnK for a component relative to ?
PhaseFraction(MoleBased)	[-]	Phase fraction on a mole base
PhaseFraction(MassBased)	[-]	Phase fraction on a mass base
Volume	[cm3/mole]	Volume of the phase
Z	[-]	Compressibility factor
MolarMass	[g/mole]	Molar mass of the phase

Density	[g/cm ³]	Density of the phase
PolymerMassFraction	[-]	Massfraction of the polymer in the phase
SolventMassFraction	[-]	Sum of the massfraction of all the solvents
Entropy	[kJ/(kg Kelvin)]	Entropy
Cp	[kJ/(kg Kelvin)]	Heat capacity
Cv	[kJ/(kg Kelvin)]	Heat capacity
Enthalpy	[kJ/kg]	Enthalpy
JT	[Kelvin/Bar]	Johle/Thomson coefficient
SpeedOfSound	[m/s]	Speed of sound
ThermalConductivity	[W/(m Kelvin)]	Thermal Conductivity
SurfaceTension	[mN/m]	Surface tension. Note only valid for a 2 phase system
Viscosity	[cP]	Viscosity
Mn	[g/mole]	Note: Only valid for polymer systems
Mw	[g/mole]	Note: Only valid for polymer systems
Mz	[g/mole]	Note: Only valid for polymer systems
BondingFraction	[-]	Association term: Fraction of sites not bonded.

5.3 Functions

Enter topic text here.

5.3.1 ResultGetNumberOfPoints

Get the number of points in the result.

For example a flash or cloud point will have one point where a phase envelope will have more than one.

This function is defined in C++ as:

```
extern "C" __declspec(dllexport) int __stdcall ResultGetNumberOfPoints(void)
```

Return

The number of points in the result.

Note

If a error/problem is found use the function [ErrorGetIndex](#) and [ErrorGetMessage](#) to get more information

Code examples

5.3.2 ResultGetArraySize

Return the size of the component arrays like Mole fraction or LnK

This function is defined in C++ as:

```
extern "C" __declspec(dllexport) int __stdcall ResultGetArraySize(int
```

PointIndex, `int` PhaseIndex)

Return

The size of the component arrays like Mole fraction or LnK

Input

Argument	Type	Description
PointIndex	int	Point to retrieve the value from
PhaseIndex	int	Phase to retrieve the value from

Note

If a error/problem is found use the function [ErrorGetIndex](#) and [ErrorGetMessage](#) to get more information

Code examples

5.3.3 ResultGetPhaseType

Return the type of phase.

2 types of phases are supported Solid and vapor/liquid.

This function is defined in C++ as:

```
extern "C" __declspec(dllexport) int __stdcall ResultGetPhaseType(int
PointIndex, int PhaseIndex)
```

Return

The type of phase:

- 0 - Solid
- 1 - Vapor/liquid

Input

Argument	Type	Description
PointIndex	int	Point to retrieve the value from
PhaseIndex	int	Phase to retrieve the value from

Note

If a error/problem is found use the function [ErrorGetIndex](#) and [ErrorGetMessage](#) to get more information

Code examples

5.3.4 ResultGetLabel

Return the label of the point. This can for example label the critical point on a phase envelope

This function is defined in C++ as:

```
extern "C" __declspec(dllexport) int __stdcall ResultGetLabel(int
PointIndex, char *LabelIndex)
```

Return

-1

Input

Argument	Type	Description
PointIndex	int	Point to retrieve the value from
LabelIndex	char pointer	Label of the point

Note

If a error/problem is found use the function [ErrorGetIndex](#) and [ErrorGetMessage](#) to get more information

Code examples**5.3.5 ResultGetPressure**

Returns the pressure of the given point.

This function is defined in C++ as:

```
extern "C" __declspec(dllexport) double __stdcall ResultGetPressure(int PointIndex)
```

Return

The value of pressure in the units set by the caller.

Input

Argument	Type	Description
PointIndex	int	Point to retrieve the value from

Note

If a error/problem is found use the function [ErrorGetIndex](#) and [ErrorGetMessage](#) to get more information

Code examples**5.3.6 ResultGetTemperature**

Returns the temperature of the given point.

This function is defined in C++ as:

```
extern "C" __declspec(dllexport) double __stdcall ResultGetTemperature(int PointIndex)
```

Return

The value of temperature in the units set by the caller.

Input

Argument	Type	Description
PointIndex	int	Point to retrieve the value from

Note

If a error/problem is found use the function [ErrorGetIndex](#) and [ErrorGetMessage](#) to get more information

Code examples

5.3.7 ResultGetStablePhaseCount

Returns the number of stable phases found.

This function is defined in C++ as:

```
extern "C" __declspec(dllexport) int __stdcall ResultGetStablePhaseCount(
int PointIndex)
```

Return

Number of stable phases found.

Input

Argument	Type	Description
PointIndex	int	Point to retrieve the value from

Note

If a error/problem is found use the function [ErrorGetIndex](#) and [ErrorGetMessage](#) to get more information

Code examples

5.3.8 ResultGetPhaseCount

Returns the total number of phases found.

Is found as: System + Feed + number of stable phases

This function is defined in C++ as:

```
extern "C" __declspec(dllexport) int __stdcall ResultGetPhaseCount(int
PointIndex)
```

Return

The total number of phases found.

Input

Argument	Type	Description
PointIndex	int	Point to retrieve the value from

Note

If a error/problem is found use the function [ErrorGetIndex](#) and [ErrorGetMessage](#) to get more information

Code examples

5.3.9 ResultGetMoleFraction

Return the mole fraction of a component in a phase.

This function is defined in C++ as:

```
extern "C" __declspec(dllexport) double __stdcall ResultGetMoleFraction(int
PointIndex, int PhaseIndex, int ComponentIndex)
```

Return

The mole fraction of a component in a phase.

Input

Argument	Type	Description
PointIndex	int	Point to retrieve the value from
PhaseIndex	int	Phase to retrieve the value from
ComponentIndex	int	Component to retrieve the value from

Note

If a error/problem is found use the function [ErrorGetIndex](#) and [ErrorGetMessage](#) to get more information

Code examples

5.3.10 ResultGetMassFraction

Return the mass fraction of a component in a phase.

This function is defined in C++ as:

```
extern "C" __declspec(dllexport) double __stdcall ResultGetMassFraction(int
PointIndex, int PhaseIndex, int ComponentIndex)
```

Return

The mass fraction of a component in a phase.

Input

Argument	Type	Description
PointIndex	int	Point to retrieve the value from
PhaseIndex	int	Phase to retrieve the value from
ComponentIndex	int	Component to retrieve the value from

Note

If a error/problem is found use the function [ErrorGetIndex](#) and [ErrorGetMessage](#) to get more information

Code examples

5.3.11 ResultGetLnK

Return the LnK of a component in a phase.

This function is defined in C++ as:

```
extern "C" __declspec(dllexport) double __stdcall ResultGetLnK(int
PointIndex, int PhaseIndex, int ComponentIndex)
```

Return

The LnK of a component in a phase.

Input

Argument	Type	Description
PointIndex	int	Point to retrieve the value from
PhaseIndex	int	Phase to retrieve the value from
ComponentIndex	int	Component to retrieve the value from

Note

If a error/problem is found use the function [ErrorGetIndex](#) and [ErrorGetMessage](#) to get more information

Code examples

5.3.12 ResultGetPhaseFractionMoleBased

Return the phase fraction (Beta) calculated on a mole bases.

This function is defined in C++ as:

```
extern "C" __declspec(dllexport) double __stdcall
ResultGetPhaseFractionMoleBased(int PointIndex, int PhaseIndex)
```

Return

The phase fraction (Beta) calculated on a mole bases.

Input

Argument	Type	Description
PointIndex	int	Point to retrieve the value from
PhaseIndex	int	Phase to retrieve the value from

Note

If a error/problem is found use the function [ErrorGetIndex](#) and [ErrorGetMessage](#) to get more information

Code examples

5.3.13 ResultGetPhaseFractionMassBased

Return the phase fraction (Beta) calculated on a mass bases.

This function is defined in C++ as:

```
extern "C" __declspec(dllexport) double __stdcall
ResultGetPhaseFractionMassBased(int PointIndex, int PhaseIndex)
```

Return

The phase fraction (Beta) calculated on a mass bases.

Input

Argument	Type	Description
PointIndex	int	Point to retrieve the value from
PhaseIndex	int	Phase to retrieve the value from

Note

If a error/problem is found use the function [ErrorGetIndex](#) and [ErrorGetMessage](#) to get more information

Code examples**5.3.14 ResultGetVolume**

Return the volume of the phase. Units: [cm3/mole]

This function is defined in C++ as:

```
extern "C" __declspec(dllexport) double __stdcall ResultGetVolume(int PointIndex, int PhaseIndex)
```

Return

The volume of the phase. Default Units: [cm3/mole]

Input

Argument	Type	Description
PointIndex	int	Point to retrieve the value from
PhaseIndex	int	Phase to retrieve the value from

Note

If a error/problem is found use the function [ErrorGetIndex](#) and [ErrorGetMessage](#) to get more information

Code examples**5.3.15 ResultGetZ**

Return the compress ability factor (Z) of the phase. Units: [-]

This function is defined in C++ as:

```
extern "C" __declspec(dllexport) double __stdcall ResultGetZ(int PointIndex, int PhaseIndex)
```

Return

The compress ability factor (Z) of the phase. Units: [-]

Input

Argument	Type	Description
PointIndex	int	Point to retrieve the value from

PhaseIndex	int	Phase to retrieve the value from
------------	-----	----------------------------------

Note

If a error/problem is found use the function [ErrorGetIndex](#) and [ErrorGetMessage](#) to get more information

Code examples**5.3.16 ResultGetMolarMass**

Return the molar mass of the phase. Units: [g/mole]

This function is defined in C++ as:

```
extern "C" __declspec(dllexport) double __stdcall ResultGetMolarMass(int PointIndex, int PhaseIndex)
```

Return

The molar mass of the phase. Units: [g/mole]

Input

Argument	Type	Description
PointIndex	int	Point to retrieve the value from
PhaseIndex	int	Phase to retrieve the value from

Note

If a error/problem is found use the function [ErrorGetIndex](#) and [ErrorGetMessage](#) to get more information

Code examples**5.3.17 ResultGetDensity**

Return the density of the phase. Units: [g/cm3]

This function is defined in C++ as:

```
extern "C" __declspec(dllexport) double __stdcall ResultGetDensity(int PointIndex, int PhaseIndex)
```

Return

The density of the phase. Units: [g/cm3]

Input

Argument	Type	Description
PointIndex	int	Point to retrieve the value from
PhaseIndex	int	Phase to retrieve the value from

Note

If a error/problem is found use the function [ErrorGetIndex](#) and [ErrorGetMessage](#) to get more information

Code examples

5.3.18 ResultGetSolventMassFraction

Return the solvent mass fraction of the phase. Units: [-]

This function is defined in C++ as:

```
extern "C" __declspec(dllexport) double __stdcall  
ResultGetSolventMassFraction(int PointIndex, int PhaseIndex)
```

Return

The solvent mass fraction of the phase. Units: [-]

Input

Argument	Type	Description
PointIndex	int	Point to retrieve the value from
PhaseIndex	int	Phase to retrieve the value from

Note

If a error/problem is found use the function [ErrorGetIndex](#) and [ErrorGetMessage](#) to get more information

Code examples

5.3.19 ResultGetPolymerMassFraction

Return the polymer mass fraction of the phase. Units: [-]

This function is defined in C++ as:

```
extern "C" __declspec(dllexport) double __stdcall  
ResultGetPolymerMassFraction(int PointIndex, int PhaseIndex)
```

Return

The polymer mass fraction of the phase. Units: [-]

Input

Argument	Type	Description
PointIndex	int	Point to retrieve the value from
PhaseIndex	int	Phase to retrieve the value from

Note

If a error/problem is found use the function [ErrorGetIndex](#) and [ErrorGetMessage](#) to get more information

Code examples

5.3.20 ResultGetEntropy

Return the entropy of the phase. Default Units: [kJ/(kg Kelvin)]

This function is defined in C++ as:

```
extern "C" __declspec(dllexport) double __stdcall ResultGetEntropy(int
PointIndex, int PhaseIndex)
```

Return

The entropy of the phase. Default Units: [kJ/(kg Kelvin)]

Input

Argument	Type	Description
PointIndex	int	Point to retrieve the value from
PhaseIndex	int	Phase to retrieve the value from

Note

If a error/problem is found use the function [ErrorGetIndex](#) and [ErrorGetMessage](#) to get more information

Code examples

5.3.21 ResultGetCp

Return the heat capacity (Cp) of the phase. Default Units: [kJ/(kg Kelvin)]

This function is defined in C++ as:

```
extern "C" __declspec(dllexport) double __stdcall ResultGetCp(int
PointIndex, int PhaseIndex)
```

Return

The heat capacity (Cp) of the phase. Default Units: [kJ/(kg Kelvin)]

Input

Argument	Type	Description
PointIndex	int	Point to retrieve the value from
PhaseIndex	int	Phase to retrieve the value from

Note

If a error/problem is found use the function [ErrorGetIndex](#) and [ErrorGetMessage](#) to get more information

Code examples

5.3.22 ResultGetCv

Return the heat capacity (Cv) of the phase. Default Units: [kJ/(kg Kelvin)]

This function is defined in C++ as:

```
extern "C" __declspec(dllexport) double __stdcall ResultGetCv(int
```

PointIndex, `int` PhaseIndex)

Return

The heat capacity (Cv) of the phase. Default Units: [kJ/(kg Kelvin)]

Input

Argument	Type	Description
PointIndex	int	Point to retrieve the value from
PhaseIndex	int	Phase to retrieve the value from

Note

If a error/problem is found use the function [ErrorGetIndex](#) and [ErrorGetMessage](#) to get more information

Code examples

5.3.23 ResultGetEnthalpy

Return the enthalpy of the phase. Default Units: [kJ/(kg)]

This function is defined in C++ as:

```
extern "C" __declspec(dllexport) double __stdcall ResultGetEnthalpy(int
PointIndex, int PhaseIndex)
```

Return

The enthalpy of the phase. Default Units: [kJ/(kg)]

Input

Argument	Type	Description
PointIndex	int	Point to retrieve the value from
PhaseIndex	int	Phase to retrieve the value from

Note

If a error/problem is found use the function [ErrorGetIndex](#) and [ErrorGetMessage](#) to get more information

Code examples

5.3.24 ResultGetJT

Return the Joule/Thomsen coefficient (JT) of the phase. Default Units: [Kelvin/Bar]

This function is defined in C++ as:

```
extern "C" __declspec(dllexport) double __stdcall ResultGetJT(int
PointIndex, int PhaseIndex)
```

Return

The Joule/Thomsen coefficient (JT) of the phase. Default Units: [Kelvin/Bar]

Input

Argument	Type	Description
PointIndex	int	Point to retrieve the value from
PhaseIndex	int	Phase to retrieve the value from

Note

If a error/problem is found use the function [ErrorGetIndex](#) and [ErrorGetMessage](#) to get more information

Code examples**5.3.25 ResultGetSpeedOfSound**

Return the speed of sound of the phase. Default Units: [m/s]

This function is defined in C++ as:

```
extern "C" __declspec(dllexport) double __stdcall ResultGetSpeedOfSound(int PointIndex, int PhaseIndex)
```

Return

The speed of sound of the phase. Default Units: [m/s]

Input

Argument	Type	Description
PointIndex	int	Point to retrieve the value from
PhaseIndex	int	Phase to retrieve the value from

Note

If a error/problem is found use the function [ErrorGetIndex](#) and [ErrorGetMessage](#) to get more information

Code examples**5.3.26 ResultGetThermalConductivity**

Return the thermal conductivity of the phase. Default Units: [W/(m Kelvin)]

This function is defined in C++ as:

```
extern "C" __declspec(dllexport) double __stdcall ResultGetThermalConductivity(int PointIndex, int PhaseIndex)
```

Return

The thermal conductivity of the phase. Default Units: [W/(m Kelvin)]

Input

Argument	Type	Description
PointIndex	int	Point to retrieve the value from
PhaseIndex	int	Phase to retrieve the value from

Note

If a error/problem is found use the function [ErrorGetIndex](#) and [ErrorGetMessage](#) to get more

information

Code examples

5.3.27 ResultGetSurfaceTension

Return the surface tension for a 2 phase system. Default Units: [mN/m]

This function is defined in C++ as:

```
extern "C" __declspec(dllexport) double __stdcall ResultGetSurfaceTension(
int PointIndex, int PhaseIndex)
```

Return

The surface tension for a 2 phase system. Default Units: [mN/m]

Input

Argument	Type	Description
PointIndex	int	Point to retrieve the value from
PhaseIndex	int	Phase to retrieve the value from

Note

If a error/problem is found use the function [ErrorGetIndex](#) and [ErrorGetMessage](#) to get more information

Code examples

5.3.28 ResultGetViscosity

Return the viscosity of a phase. Default Units: [cP]

This function is defined in C++ as:

```
extern "C" __declspec(dllexport) double __stdcall ResultGetViscosity(int
PointIndex, int PhaseIndex)
```

Return

The viscosity of a phase. Default Units: [cP]

Input

Argument	Type	Description
PointIndex	int	Point to retrieve the value from
PhaseIndex	int	Phase to retrieve the value from

Note

If a error/problem is found use the function [ErrorGetIndex](#) and [ErrorGetMessage](#) to get more information

Code examples

5.3.29 ResultGetMn

Return Mn for a phase. Default Units: [g/mole]

Note: This property is only available for a polymer system.

This function is defined in C++ as:

```
extern "C" __declspec(dllexport) double __stdcall ResultGetMn(int
PointIndex, int PhaseIndex)
```

Return

The Mn for a phase. Default Units: [g/mole]

Input

Argument	Type	Description
PointIndex	int	Point to retrieve the value from
PhaseIndex	int	Phase to retrieve the value from

Note

If a error/problem is found use the function [ErrorGetIndex](#) and [ErrorGetMessage](#) to get more information

Code examples

5.3.30 ResultGetMw

Return Mw for a phase. Default Units: [g/mole]

Note: This property is only available for a polymer system.

This function is defined in C++ as:

```
extern "C" __declspec(dllexport) double __stdcall ResultGetMw(int
PointIndex, int PhaseIndex)
```

Return

The Mw for a phase. Default Units: [g/mole]

Input

Argument	Type	Description
PointIndex	int	Point to retrieve the value from
PhaseIndex	int	Phase to retrieve the value from

Note

If a error/problem is found use the function [ErrorGetIndex](#) and [ErrorGetMessage](#) to get more information

Code examples

5.3.31 ResultGetMz

Return Mz for a phase. Default Units: [g/mole]

Note: This property is only available for a polymer system.

This function is defined in C++ as:

```
extern "C" __declspec(dllexport) double __stdcall ResultGetMz(int
PointIndex, int PhaseIndex)
```

Return

The Mz for a phase. Default Units: [g/mole]

Input

Argument	Type	Description
PointIndex	int	Point to retrieve the value from
PhaseIndex	int	Phase to retrieve the value from

Note

If a error/problem is found use the function [ErrorGetIndex](#) and [ErrorGetMessage](#) to get more information

Code examples

5.3.32 ResultGetBondingFraction

Return the bonding fraction for a site calculated by the association term. Units: [-]

Note: Bonding fraction is defined as the fraction of sites not bonded.

This function is defined in C++ as:

```
extern "C" __declspec(dllexport) double __stdcall ResultGetBondingFraction(
int PointIndex, int PhaseIndex, int ComponentIndex, int BlockIndex, int
SiteIndex)
```

Return

The bonding fraction for a site calculated by the association term. Units: [-]

Input

Argument	Type	Description
PointIndex	int	Point to retrieve the value from
PhaseIndex	int	Phase to retrieve the value from
ComponentIndex	int	Component to retrieve the value from
BlockIndex	int	Block to retrieve the value from
SiteIndex	int	Site to retrieve the value from

Note

If a error/problem is found use the function [ErrorGetIndex](#) and [ErrorGetMessage](#) to get more information

Code examples

5.3.33 ResultGetSolubilityParameter

This function returns the solubility parameter for the given phase.
It is calculated as:

$$\sqrt{\frac{\frac{J}{\text{mol}}}{\text{cm}^3}} = \sqrt{1 \cdot 10^6 \frac{J}{\text{m}^3}} = \sqrt{\text{MPa}}$$

This function is defined in C++ as:

```
extern "C" __declspec(dllexport) double __stdcall
ResultGetSolubilityParameter(int PointIndex, int PhaseIndex)
```

Return

The bonding fraction for a site calculated by the association term. Units: [-]

Input

Argument	Type	Description
PointIndex	int	Point to retrieve the value from
PhaseIndex	int	Phase to retrieve the value from

Note

If a error/problem is found use the function [ErrorGetIndex](#) and [ErrorGetMessage](#) to get more information

Code examples

6 Properties

The functions in properties lets the user calculate a range of values from the chosen thermodynamic model.

For example fugacity's and related derivatives or properties like enthalpy or volume.

When calculating properties from a EOS 2 out of 3 variables has to be fixed. The 3 possible set are:

- Temperature/Pressure
- Temperature/Volume
- Pressure/Volume

VLXE comes with all 3 options.

6.1 List of functions

The functions are divided into 2 groups.

The first group perform the calculations and the second one lets the caller retrieve the results.

Calculations

- PropUpdate_TP
- PropUpdateLnTP

Results

6.2 Functions

Enter topic text here.

6.2.1 PropUpdatePropertyTP

This function update a range of properties at fixed temperature and pressure

This function is defined in C++ as:

```
extern "C" __declspec(dllexport) int __stdcall PropUpdate_TP(double
Temperature, double Pressure, int VolumeType)
```

Return

The function return -1 if no problem was found. If a different number is returned check the argument ErrorMessage for more information.

Input

Argument	Type	Units	Description
Temperature	double	User defined	Temperature to use in calculation
Pressure	double	User defined	Pressure to use in the calculation
VolumeType	int	[-]	Type of volume root to use from the EOS <ul style="list-style-type: none"> • 0: Liquid • 1: Vapor • 2: Auto (Stable root)

Note

If a error/problem is found use the function [ErrorIndex](#) and [ErrorMessage](#) to get more information

6.2.2 PropUpdateLnfTP

This function calculates the natural log to fugacity's and several analytical derivatives:

- Temperature
- Pressure
- Mole numbers (Depends on the argument Updatedni)

This function is defined in C++ as:

```
extern "C" __declspec(dllexport) int __stdcall PropUpdateLnfTP(double
Temperature, double Pressure, int VolumeType, int Updatedni, int
UseCompress)
```

Return

The function return -1 if no problem was found. If a different number is returned use the [error functions](#) to get more information.

Input

Argument	Type	Units	Description
Temperature	double	User defined	Temperature to use in calculation

Pressure	double	User defined	Pressure to use in the calculation
VolumeType	int	[-]	Type of volume root to use from the EOS <ul style="list-style-type: none"> • 0: Liquid • 1: Vapor • 2: Auto (Stable root)
Updatedni	int	[-]	Update the derivatives with respect to mole numbers. Valid input are: <ul style="list-style-type: none"> • 0: No • 1: Yes
UseCompress	int	[-]	Advanced use. Determine if the arrays for a poly-disperse polymer are compressed or not. Valid input are: <ul style="list-style-type: none"> • 0: No • 1: Yes

6.2.3 PropGetLnf

This function lets the user retrieve the natural log to the fugacity.

This function is defined in C++ as:

```
extern "C" __declspec(dllexport) double __stdcall PropGetLnf (int
ComponentIndex)
```

Return

The function return the natural log to the fugacity for the given component number.

Input

Argument	Type	Units	Description
ComponentIndex	int	[-]	Index of the component for which the property is wanted

7 Examples

Examples on how to use the SDK is given for 3 languages:

- [C++](#)
- [C#](#)
- [Fortran](#)

For each language a set of classes are provided that lets the user communicate with the dll. Fortran does not have classes so example code is provided that is less generic.

4 examples are given for each language. for each the model is written, a cloud point is calculated and some result properties are retrieved from the dll

Nr.	Name	Description
1	n-Hexane	Example for a single small molecule
2	n-Hexane + Methanol	Example for a binary system of small molecules
3	n-Hexane + HDPE with 7 pseudo components	Example with a poly-disperse polymer and one solvent
4	n-Hexane + EVA with 7 pseudo components	Example with a poly-disperse co-polymer and one solvent

7.1 C++

See the example project on the VLXE homepage

7.2 C#

See the example project on the VLXE homepage

7.3 Fortran

See the example project on the VLXE homepage

8 Background information

Enter topic text here.

8.1 Ideal gas heat capacity

A number of physical properties can be calculated from an equation of state combined with a model for an ideal gas heat capacity.

VLXE output the most common:

- Enthalpy
- Entropy

The two most common expressions for the ideal gas heat capacity in VLXE are:

- Polynomial
- DIPPR

The user sets the model independent for solvents and polymers.

Reference temperature and pressure

When calculating e.g. enthalpy and entropy a reference temperature and pressure is needed in the integration of the ideal gas expression,

The values used in VLXE are:

Reference temperature: 298.15 Kelvin

Reference pressure: 1.0 bar

Polynomial

The polynomial is given as:



Note the units of C_p .

$C(7)$ and $C(8)$ defines the temperature range where the expression is valid

For co-polymer the formula used is:

$$C_p \left[\frac{kJ}{kg \cdot kelvin} \right] = \sum_{b=1}^{Blockcount} bmf \cdot (C^b(1) + C^b(2) \cdot T + C^b(3) \cdot T^2 + C^b(4) \cdot T^3 + C^b(5) \cdot T^4 + C^b(6) \cdot T^5)$$

$, C^b(7) \leq T \leq C^b(8)$. $bmf = \text{block massfraction}$

DIPPR

The DIPPR expression is given by:



Note the units of C_p .

$C(7)$ and $C(8)$ defines the temperature range where the expression is valid

For co-polymers the formula used is:

$$C_p \left[\frac{kJ}{kg \cdot kelvin} \right] = \sum_{b=1}^{Blockcount} bmf \cdot \left(C^b(1) + C^b(2) \cdot \left(\frac{\frac{C^b(3)}{T}}{\sinh\left(\frac{C^b(3)}{T}\right)} \right)^2 + C^b(4) \cdot \left(\frac{\frac{C^b(5)}{T}}{\cosh\left(\frac{C^b(5)}{T}\right)} \right)^2 \right)$$

$$C^b(6) \leq T \leq C^b(7), bmf = \text{block massfraction}$$

Often the parameters used are reported as dimensionless values, however it is converted into units as:



Note that in the DIPPR expression only $C1$, $C2$ and $C4$ has to be converted.

Poly disperse polymers

Polymers that are poly disperse poses a special problem since the parameters for the ideal heat capacity expression cannot be given for each pseudo component and often is not known.

The method that is used in VLXE is based on the fact that for heavy alkane's the value for C_p on a mass basis becomes independent on the alkane as the molar mass becomes big.

This information is used, so only one set of parameters has to be given for a polymer. The parameters are then scaled using the molar mass of each pseudo component.

Parameters for a given polymer mono-/poly-disperse

If no values are available for a given polymer it is recommended to use the values for a $C_{20}H_{42}$:

Eicosane.

The values are:

	C(1)	C(2)	C(3)	C(4)	C(5)	C(6)	C(7)	C(8)
Polynomial C _p (kJ/kg K)	8.1694E-1	-3.0569E-4	1.5706E-5	-2.1058E-8	8.5058E-12	0	200	1000
DIPPR C _p (kJ/(kg K))	1.1496	3.9249	1636.0	2.6367	-726.27	200	1500	-

Not much research has been done in this field. As more research is done VLXE will become updated.

Please contact VLXE if you prefer another method to included in.

9 Appendix A: Version history

Below is a brief version history of the SDK

22-Oct-2008 - Version (6.3.0)

- Provided new example projects for C#, C++ and Fortran.

25-Apr-2008 - Version (5.1.0)

- Fixed a few misprints in this documentation.
- Added [solubility parameter](#) to the result list.
- Fortran example project is now available.
- Added a new function: [CalcFlashTPFlow](#). It is designed to increase the speed by reusing the last flash result. To be used in flow simulators.

05-Mar-2008 - Version (5.0.2)

- Fixed a bug in the documentation for CalcFlashPH and CalcFlashPS. The table said temperature where it should be pressure.

21-Feb-2008 - Version (5.0.1)

- Removed the "const" from all arguments, since it has no direct use
- Added the 4 bubble/dew + all 7 flash types current available.
- Moved the C++ example project to a console project. This is done to have a "pure" C++ solution
- Made a "pure" C# example.
- Improved the output from both the C++ and C# example projects

13-Feb-2008 - Version (5.0.0)

- First public beta released

26-Jan-2008

- Work started on this SDK

Index

- A -

Appendix A: Version history 62
Association scheme 31
Association term 31

- C -

C# 60
C++ 60
CalcDewP 34
CalcDewT 35
CalcFlashPH 36
CalcFlashTPFlow 39
Calculation group 31
Calculations 31, 32, 36
Contact VLXE 1

- D -

Default units 2
Definition of a component 4

- E -

Error handling 3
ExampleFortran 60
Examples 59, 60

- F -

FlashTP 36
Fortran 60
Function groups 1
Functions 1
FunctionsProperties 57

- G -

GetErrorIndex 3
GetErrorMessage 3

- H -

How to perform a calculation 2

- I -

Ideal gas heat capacity 60

- L -

List of functions 4, 32, 40

- M -

Model 4, 29
Model group of functions 4
Modelclear 6
ModelSetComponentAssociationUse 17
ModelSetComponentCplgDIPPR 12
ModelSetComponentCplgPoly 11
ModelSetModelAssociationMatrix 8
ModelSetModelCplgPolymer 7
ModelSetModelCplgStandard 7
ModelSetModelEOS 6
ModelSetModelSLE 7

- P -

Parameter overview 29
Properties 41, 57
PropGetLnf 59
PropUpdateLnfTP 58
PropUpdatePropertyTP 58

- R -

ResultGetPressure 44
Results 40, 41, 44

- S -

SetStreamFeed 32

- U -

Units 2

- V -

Version history 62