

**CAPE-OPEN**

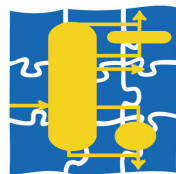
Expanding Process Modelling Capability  
through  
Software Interoperability Standards

---

## Errata and clarifications

**Thermodynamic and Physical Properties v1.1**

---



**CO ▾ LaN**

[www.colan.org](http://www.colan.org)

---

## ARCHIVAL INFORMATION

---

Filename	Outstanding_issues_1.1.doc
Authors	CO-LaN consortium
Status	Public
Date	28 March 2008
Version	Version 1.10
Number of pages	25
Versioning	version 1.00 edited by Michel Pons (CO-LaN) and Werner Drewitz (BASF AG)
	Version 1.01 edited by Michel Pons (CO-LaN) incorporating text from Jasper van Baten (AmsterCHEM), Ross Taylor, Harry Kooijman (ChemSep) and Richard Baur (Shell Global Solutions).
	Version 1.02 edited by Michel Pons (CO-LaN).
	Version 1.03 edited by Jasper van Baten (AmsterCHEM) and Michel Pons (CO-LaN).
	Version 1.04 incorporating definition of mole fraction derivatives by Jasper van Baten
	Version 1.05 edited by Werner Drewitz
	Version 1.06 edited by Werner Drewitz
	Version 1.07 edited by Jasper van Baten
	Version 1.08 edited by Michel Pons: incorporating modified versions of SetSinglePhaseProp and SetTwoPhaseProp methods.
	Version 1.09 edited by Werner Drewitz: paragraph 3.8 added
	Version 1.10 edited by Michel Pons (proof-read)
Additional material	
Web location	<a href="http://www.colan.org/Spec%2011/Outstanding_issues_1.1.pdf">http://www.colan.org/Spec%2011/Outstanding_issues_1.1.pdf</a>
Implementation specifications version	1.1
Comments	

---

## IMPORTANT NOTICES

---

### **Disclaimer of Warranty**

CO-LaN documents and publications include software in the form of *sample code*. Any such software described or provided by CO-LaN --- in whatever form --- is provided "as-is" without warranty of any kind. CO-LaN and its partners and suppliers disclaim any warranties including without limitation an implied warrant or fitness for a particular purpose. The entire risk arising out of the use or performance of any sample code --- or any other software described by the CAPE-OPEN Laboratories Network --- remains with you.

Copyright © 2001-2008 CO-LaN. All rights are reserved unless specifically stated otherwise.

CO-LaN is a not for profit organization established under French law of 1901.

### **Trademark Usage**

Many of the designations used by manufacturers and sellers to distinguish their products are claimed as trademarks. Where those designations appear in CO-LaN publications, and the authors are aware of a trademark claim, the designations have been printed in caps or initial caps.

Microsoft, Microsoft Word, Visual Basic, Visual Basic for Applications, Internet Explorer, Windows and Windows NT are registered trademarks and ActiveX is a trademark of Microsoft Corporation.

Netscape Navigator is a registered trademark of Netscape Corporation.

Adobe Acrobat is a registered trademark of Adobe Corporation.

---

## SUMMARY

---

This document describes clarifications and corrects errors in the CAPE-OPEN Thermodynamic and Physical Properties interface specification version 1.1.

---

## **ACKNOWLEDGEMENTS**

---

Many individuals and their organizations have contributed to this document. Richard Szczepanski from Infochem Computer Services Ltd., Werner Drewitz from BASF AG, Jasper van Baten from AmsterCHEM and Michel Pons from CO-LaN were among the main contributors.

---

# CONTENTS

---

<b>1.</b>	<b>INTRODUCTION.....</b>	<b>7</b>
<b>2.</b>	<b>AUDIENCE .....</b>	<b>7</b>
<b>3.</b>	<b>CLARIFICATIONS.....</b>	<b>8</b>
3.1	ChemicalFormula (section 7.5.2).....	8
3.2	ICapeThermoMaterialContext::SetMaterial() (section 6.2).....	9
3.3	(unconstrained) Mole fraction derivatives (section 7.6).....	12
3.4	Two-phase property derivatives w. r. t. temperature/pressure (section 7.6).....	13
3.5	Ideal gas state reference pressure .....	16
3.6	Is heat of formation included in entropy and enthalpy? .....	17
3.7	Remarks about getting, setting and calculating properties and the use of phase identifiers .....	18
3.8	Can (Pseudo-)Component Properties be set by a Property Calculator? .....	23
<b>4.</b>	<b>ERRATA .....</b>	<b>24</b>
4.1	Section 5.5 page 25.....	24
4.2	Section 5.5 page 115 (units vanderwaals entities).....	24
4.3	Section 6.3 ECapeBadInvOrder instead of EcapeBadInvokationOrder .....	24
<b>5.</b>	<b>BIBLIOGRAPHY.....</b>	<b>25</b>

## **1. Introduction**

This document lists errata and clarifications that have been raised and added after the formal approval process for the CAPE-OPEN Thermodynamic and Physical Properties interface specification version 1.1.

The intent is to clarify the interface specification where needed and to pinpoint what is missing in the specification document.

## **2. Audience**

This document is intended primarily for software developers who want to build CAPE-OPEN Property Package, Equilibrium Calculator and Physical Property Calculator components. It is also intended for the developers of other software components, such as Unit Operations and Reaction Packages, which make use of these Thermodynamic and Physical Properties software components. Finally, it is intended for developers of CAPE-OPEN-compliant Process Modelling Environments because it relates to how the interfaces are to be used to implement communication between the environment and the external software components.

This document is not intended for end-users of CAPE-OPEN software components or process simulation software.

### 3. Clarifications

#### 3.1 ChemicalFormula (section 7.5.2)

Issue: Are unitary atomicities explicitly specified?

Is this really required to be specified by the standard or can it be left to the Property Package owner?

Advice: the chemical formula is not well suited to be used for compound identification so no automatic comparison should be made.

Issue: Is the value of chemicalFormula case-independent, as returned by method GetCompoundList of ICapeThermoCompounds interface?

Advice: It is required for this value to be case sensitive; otherwise it is not possible to tell the difference between for example CS and Cs (if there was such a thing as Carbon-mono-sulfide).

**Adopted:** page 113 section 7.5.2.

<http://webbook.nist.gov/chemistry>

Compounds can be accessed directly with

<http://webbook.nist.gov/cgi/cbook.cgi?Formula=ch4&NoIon=on&Units=SI>

or

<http://webbook.nist.gov/cgi/cbook.cgi?Name=water&Units=SI>

CAS numbers can be undefined, for example for petroleum fractions, in which case comparison has to be done by looking at constant properties.

ChemicalFormula

The formula is delivered in Hill nomenclature [11]: organic compounds: first C, then H, other atoms alphabetical; inorganic compounds: all atoms alphabetical. The value for ChemicalFormula is case sensitive.

The following properties are returned by the GetCompoundConstant method of the ICapeThermoCompounds interface as numerical (Double) values.

### 3.2 ICapeThermoMaterialContext::SetMaterial() (section 6.2)

Issue: There is no clear statement that if the PME plays with the compound list in a Material Object then it needs to call SetMaterial () again on the PP. It may result that every call to the PP is preceded by a call to SetMaterial ().

Proposal: add to section 6.2 notes the following statement:

"After a call to SetMaterial() has been received, the object implementing ICapeThermoMaterialContext can assume that the number, name and order of compounds for that Material Object will remain fixed until the next call to SetMaterial() or UnSetMaterial()."

Advice: the necessity of this became apparent during testing between COCO, Multiflash and gO:CAPE-OPEN. COCO reuses the same Material Object again and again to retrieve compound constants for a single compound. Before each call it changes the Material Object list of compounds to contain just the compound it is interested in. As a result of this, Multiflash needed to query the Material Object on *every* call to find out what compound it contained because this may have changed since the previous call. This was not very efficient.

Adopted: description of SetMaterial method in section 6.2 pages 59-60 is now

# SetMaterial

**Interface Name**      **ICapeThermoMaterialContext**

**Method Name**        SetMaterial

**Returns**              CapeError

## Description

Allows the client of a component that implements this interface to pass an ICapeThermoMaterial interface to the component, so that it can access the properties of a Material.

## Arguments

Name	Type	Description
[in] <i>material</i>	CapeInterface	The Material interface.

## Notes

The SetMaterial method allows a Thermodynamic and Physical Properties component, such as a Property Package, to be given the ICapeThermoMaterial interface of a Material Object. This interface gives the component access to the description of the Material for which Property Calculations or Equilibrium Calculations are required. The component can access property values directly using this interface. A client can also use the ICapeThermoMaterial interface to query a Material Object for its ICapeThermoCompounds and ICapeThermoPhases interfaces, which provide access to Compound and Phase information, respectively.

It is envisaged that the SetMaterial method will be used to check that the Material Interface supplied is valid and useable. For example, a Property Package may check that there are some Compounds in a Material Object and that those Compounds can be identified by the Property Package. In addition a Property Package may perform any initialisation that depends on the configuration of a Material Object. A Property Calculator component might typically use this method to query the Material Object for any required information concerning the Compounds.

Calling the UnsetMaterial method of the ICapeThermoMaterialContext interface has the effect of removing the interface set by the SetMaterial method.

After a call to SetMaterial() has been received, the object implementing ICapeThermoMaterialContext can assume that the number, name and order of compounds for that Material Object will remain fixed until the next call to SetMaterial() or UnSetMaterial().

## Exceptions

ECapeNoImpl – The operation is “not” implemented even if this method can be called for reasons of compatibility with the CAPE-OPEN standards. That is to say that the operation exists, but it is not supported by the current implementation.

ECapeInvalidArgument – The input argument is not a valid CapeInterface.

ECapeFailedInitialisation – The pre-requisites for the property calculation are not valid. For example:

- There are no Compounds in the object that implements the ICapeThermoMaterial interface.

- The Compounds cannot be identified by the client (*e.g.* a Property Package). This case is a possibility if the way a Material Object has been configured by a PME is not consistent with the Property Package being used.

ECapeUnknown – The error to be raised when other error(s), specified for the operation, are not suitable.

### 3.3 (unconstrained) Mole fraction derivatives (section 7.6)

Issue: Some users need mole fraction derivatives in addition to mole number derivatives. They were defined in version 1.0 of the interface specification but were eliminated in version 1.1 because there is no unique definition of mole fraction derivatives and it was thought that the mole number derivatives could also serve as mole fraction derivatives. See Annex 1 for a detailed discussion of what is needed.

Proposal: adopt a definition of `.DmolFraction` as the unconstrained mole fraction derivative of properties using the following definition:

$$\left(\frac{\partial M}{\partial x_k}\right)_{x_{j \neq k}, T, P} = \frac{\partial M(x_1, \dots, x_N, T, P)}{\partial x_k}$$

**Adopted:** section 7.6 page 125 is now: see next item.

### 3.4 Two-phase property derivatives w. r. t. temperature/pressure (section 7.6)

Issue: If for two-phase properties temperature/pressure of the two phases are different (at nonequilibrium points) it is not clear what the meaning of property.Dtemperature or property.Dpressure is.

Proposal: The meaning of property.Dtemperature or property.Dpressure for two-phase properties is the sum of the derivatives for each phase.

Adopted: section 7.6 page 125 is now:

Derivatives are built from the property identifier: a point with a D meaning "Derivative" and the name for the independent variable. The only independent variables that may be specified are temperature, pressure and mole numbers, as shown in the table below.

Derivative identifier	meaning	units
property.Dtemperature	derivative of property with respect to temperature with pressure and composition fixed.	[property]/K
property.Dpressure	derivative of property with respect to pressure with temperature and composition fixed.	[property]/Pa
property.Dmoles	<p>derivatives of property with respect to mole number keeping pressure and temperature and other mole numbers fixed for a mixture containing a total of one mole of material. For some property H the ith element of derivative is</p> $H.Dmoles_i = \bar{h}_i = \left( \frac{\partial H}{\partial n_i} \right)_{p,T,n_{j \neq i}}$ <p>For a two-phase property the mole number derivatives are evaluated independently for each phase by keeping the temperature and pressure of both phases and the mole numbers in the other phase fixed.</p>	[property]/mol

Derivative identifier	meaning	units
Property.DmolFraction	<p>derivatives of property with respect to mole fraction, keeping pressure and temperature and other mole fractions fixed. The mole fractions are therefore treated as independent variables. These derivatives are a mathematical construction and do not necessarily have a physical meaning. The derivatives depend on the specific implementation of the property in the property package and may therefore not be unique. So mole fraction derivatives from different Property Packages can't be expected to coincide in general. However they should coincide as directional derivatives with directions <math>d</math> that lie in the plane</p> $\sum_{i=1}^N x_i = 1, \text{ i.e. } \sum_{i=1}^N d_i = 0.$ <p>The directional derivative is the scalar product of the derivative ("gradient") and the direction <math>d</math>:</p> $\nabla_x H(\vec{x}, T, P) \cdot \vec{d} = \sum_{i=1}^N \frac{\partial H(x_1, \dots, x_N, T, P)}{\partial x_i} \cdot d_i$ <p>For some property <math>H</math>, the <math>i^{\text{th}}</math> element of the derivative is</p> $H.DmolFraction = \left( \frac{\partial H}{\partial x_i} \right)_{x_{j \neq i}, T, P} = \frac{\partial H(x_1, \dots, x_N)}{\partial x_i}$ <p>For a two-phase property the mole number derivatives are evaluated independently for each phase by keeping the temperature and pressure of both phases and the mole fractions in the other phase fixed.</p>	

Derivatives of two-phase properties are not equilibrium derivatives. That means that composition derivatives are evaluated independently for each phase keeping the temperature and pressure of both phases and the composition of the other phase fixed. Similarly a temperature (or pressure) derivative does not imply any change in the phase compositions or the pressure (or temperature).

For two-phase property derivatives with respect to temperature property.Dtemperature is the sum of the derivatives for each phase even if the temperatures in both phases are not equal. The same is valid for property.Dpressure.

### 3.5 Ideal gas state reference pressure

Issue: The ideal gas state reference pressure is important for entropy calculation. It is often equal to 1 atm or 1 bar meaning a little difference for the absolute values. The precise reference pressure is at the moment not available.

Proposal: make ideal gas state reference pressure available as a “universal constant” with identifier IdealGasStateReferencePressure.

Adopted: section 7.5.1 pages 111-112 is now

The following constants are returned by the GetUniversalConstant method of the ICapeThermoUniversalConstants interface. The possible return types are Double or String. Note: only the units of measurement are specified in the CAPE-OPEN standards, not the values.

Identifier	typical value	units	return type
avogadroConstant	6.022 141 99(47)×10 <sup>23</sup>	1/mol	Double
boltzmannConstant	1.380 6503(24)×10 <sup>-23</sup>	J /K	Double
molarGasConstant	8.314 472(15)	J /mol/ K	Double
speedOfLightInVacuum	299792458(1)	m/s	Double
standardAccelerationOfGravity	9.806 65	m /s <sup>2</sup>	Double
IdealGasStateReferencePressure	101325	Pa	Double

### 3.6 Is heat of formation included in entropy and enthalpy?

Issue: It is not stated in the spec whether entropy and enthalpy includes the heat of formation, so it is up to decision of the implementer of the Property Package. It may or may not be included!

Answer: It was decided to introduce new identifiers “entropyF” and “enthalpyF” for entropy and enthalpy including heat of formation. The reference state is defined by the Property Package and has to be consistent for all compounds.

For appropriate flash calculations (PH, PS, HS) only the (overall) identifiers “enthalpy” and “entropy” are to be used for the analogue properties as before.

**Adopted:** list 7.5.5 Non-constant single-phase mixture properties include now:

enthalpy	Enthalpy (may or may not include heat of formation)	E		J	mole/mass	Y
enthalpyF	Enthalpy, including heat of formation	E		J	mole/mass	Y
entropy	Entropy (may or may not include entropy of formation)	E		J/K	mole/mass	Y
entropyF	Entropy, including entropy of formation	E		J/K	mole/mass	Y

### 3.7 Remarks about getting, setting and calculating properties and the use of phase identifiers

Issue: there are some inconsistencies regarding the descriptions of phase identifiers to be passed to the routines for getting and setting properties.

Answer: The consensus is that it should be possible to get properties only for phases that already exist, whereas setting properties should cause the phases to exist. The latter is decided as to facilitate calculation of physical properties on specific phases: e.g. to calculate liquid density, one should only have to set the liquid properties, calculate the density property and get the density property, without having to make sure liquid is an existing phase on the Material Object. The liquid phase however should of course be a phase supported by the Property Package. To summarize:

- To get a property using `GetSinglePhaseProp`, `GetTwoPhaseProp` and `GetTPFraction`, a phase identifier should be passed that appears in the list returned by `ICapeThermoMaterial::GetPresentPhases` (that is, the phase should be present)
- To set a property using `SetSinglePhaseProp` or `SetTwoPhaseProp`, a phase identifier should be passed that is supported by the Property Package or Material Object, i.e. one that appears in the list returned by `ICapeThermoPhases::GetPhaseList`. Setting such a property should cause the phase to be present on the Material Object, as if it were present in a call to `SetPresentPhases` with status `Cape_UnknownPhaseStatus`.
- From the above follows that – as opposed to stated in the description of `SetSinglePhaseProp` – `SetPresentPhases` does not need to be called before `SetSinglePhaseProp` or `SetTwoPhaseProp`.
- `CalcSinglePhaseProp` and `CalcTwoPhaseProp` should be passed phase labels of phases that are present. As usually conditions are set for which to calculate properties, this condition is usually satisfied (no need for `SetPresentPhases`). The phase identifiers that are used must appear in the list that is returned by `ICapeThermoMaterial::GetPresentPhases`. If property calculations are to be performed on phases that are in equilibrium, or if calculations are done on phases for which conditions have been specified using `SetSinglePhaseProp`, this condition is generally automatically fulfilled.
- `CalcAndGetLnPhi` can be performed on any phase that is supported by the Property Package, so the phase identifier must appear in `ICapeThermoPhases::GetPhaseList`.
- A Material Object should always implement `ICapeThermoPhases`, as to provide a list of valid phases for setting properties and performing property calculations.
- `SetPresentPhases` should generally only be used in the context of the equilibrium calculations (specifying which phases are to take part in the equilibrium calculation, and setting the result of the equilibrium calculation). In the context of dynamic simulations `SetPresentPhases` is to be used to completely specify a Material Object leaving a unit operation, equivalent to calculating an equilibrium in steady state simulations.
- `GetPresentPhases` is used in multiple contexts: a) it is used by equilibrium calculations to consider phases that may exist in the equilibrium, b) it is used by PMCs or the PME to obtain the result of an equilibrium calculation, c) it will be used by PMCs in general to see which phases exist in a Material Object.  
The phases returned by `GetPresentPhases` do not necessarily represent phases in equilibrium.

Adopted: in Chapter 6.1, descriptions of `SetSinglePhaseProp` and `SetTwoPhaseProp` methods are now:

## SetSinglePhaseProp

**Interface Name**      **ICapeThermoMaterial**

**Method Name**        SetSinglePhaseProp

**Returns**              CapeError

### Description

Sets single-phase non-constant property values for a mixture.

### Arguments

Name	Type	Description
[in] <i>property</i>	CapeString	The identifier of the property for which values are set. This must be one of the single-phase properties or derivatives. The standard identifiers are listed in sections 7.5.5 and 7.6.
[in] <i>phaseLabel</i>	CapeString	Phase label of the Phase for which the property is set. The phase label must be one of the strings returned by the GetPhaseList method of this interface.
[in] <i>basis</i>	CapeString	Basis of the results. Valid settings are: “Mass” for Physical Properties per unit mass or “Mole” for molar properties. Use UNDEFINED as a place holder for a Physical Property for which basis does not apply. See section 0 for details.
[in] <i>values</i>	CapeVariant	Values to set for the property (CapeArrayDouble) or CapeInterface (see notes).

### Notes

The *values* argument of SetSinglePhaseProp is either a CapeArrayDouble that contains one or more numerical values to be set for a property, *e.g.* temperature, or a CapeInterface that may be used to set single-phase properties described by a more complex data structure, *e.g.* distributed properties.

Although some properties set by calls to SetSinglePhaseProp will have a single numerical value, the type of the *values* argument for numerical values is CapeArrayDouble and in such a case the method must be called with *values* containing an array even if it contains only a single element.

The property values set by SetSinglePhaseProp refer to a single Phase. Properties that depend on more than one Phase, for example surface tension or K-values, are set by the SetTwoPhaseProp method of the Material Object.

To set a property using SetSinglePhaseProp, a *phaseLabel* identifier should be passed that is supported by the Property Package or Material Object, *i.e.* one that appears in the list returned by ICapeThermoPhases::GetPhaseList. Setting such a property should cause the phase to be present on the Material Object, as if it were present in a call to SetPresentPhases with status Cape\_UnknownPhaseStatus.

## Exceptions

ECapeNoImpl – The operation is “not” implemented even if this method can be called for reasons of compatibility with the CAPE-OPEN standards. That is to say that the operation exists but it is not supported by the current implementation. This method may not be required if the PME does not deal with any single-phase properties.

ECapeInvalidArgument - To be used when an invalid argument value was passed, that is a value that does not belong to the valid list described above, for example UNDEFINED for *property*.

ECapeOutOfBounds – one or more of the entries in the *values* argument is outside of the range of values accepted by the Material Object.

EcapeFailedInitialisation - The pre-requisites are not valid. The phase referenced has not been created using SetPresentPhases.

ECapeUnknown – The error to be raised when other error(s), specified for the SetSinglePhaseProp operation, are not suitable.

## SetTwoPhaseProp

**Interface Name**      **ICapeThermoMaterial**

**Method Name**        SetTwoPhaseProp

**Returns**              CapeError

### Description

Sets two-phase non-constant property values for a mixture.

### Arguments

Name	Type	Description
[in] <i>property</i>	CapeString	The property for which values are set in the Material Object. This must be one of the two-phase properties or derivatives included in sections 7.5.6 and 7.6.
[in] <i>phaseLabels</i>	CapeArrayString	Phase labels of the Phases for which the property is set. The Phase labels must be two of the identifiers returned by the <code>GetPhaseList</code> method of the <code>ICapeThermoPhases</code> interface.
[in] <i>basis</i>	CapeString	Basis of the results. Valid settings are: “Mass” for Physical Properties per unit mass or “Mole” for molar properties. Use UNDEFINED as a place holder for a Physical Property for which basis does not apply. See section 0 for details.
[in] <i>values</i>	CapeVariant	Value(s) to set for the property (CapeArrayDouble) or CapeInterface (see notes).

### Notes

The *values* argument of `SetTwoPhaseProp` is either a `CapeArrayDouble` that contains one or more numerical values to be set for a property, *e.g.* `kvalues`, or a `CapeInterface` that may be used to set two-phase properties described by a more complex data structure, *e.g.* distributed properties.

Although some properties set by calls to `SetTwoPhaseProp` will have a single numerical value, the type of the *values* argument for numerical values is `CapeArrayDouble` and in such a case the method must be called with the *values* argument containing an array even if it contains only a single element.

The Physical Property values set by `SetTwoPhaseProp` depend on two Phases, for example surface tension or K-values. Properties that depend on a single Phase are set by the `SetSinglePhaseProp` method.

If a Physical Property with composition derivative is specified, the derivative values will be set for *both* Phases in the order in which the Phase labels are specified. The number of values returned for a composition derivative will depend on the property. For example, if there are *N* Compounds then the *values* vector for the surface tension derivative will contain *N* composition derivative values for the first Phase, followed by *N* composition derivative values for the second Phase. For K-values

there will be  $N^2$  derivative values for the first phase followed by  $N^2$  values for the second phase in the order defined in 7.6.2.

To set a property using SetTwoPhaseProp, *phaseLabels* identifiers should be passed that are supported by the Property Package or Material Object, i.e. one that appears in the list returned by ICapeThermoPhases::GetPhaseList. Setting such a property should cause the phases to be present on the Material Object, as if it were present in a call to SetPresentPhases with status Cape\_UnknownPhaseStatus.

### Exceptions

ECapeNoImpl – The operation is “not” implemented even if this method can be called for reasons of compatibility with the CAPE-OPEN standards. That is to say that the operation exists, but it is not supported by the current implementation. This method may not be required if the PME does not deal with any two-phase properties.

ECapeInvalidArgument – To be used when an invalid argument value was passed, that is a value that does not belong to the valid lists described above, for example if UNDEFINED is used for identifying the property, or the calculation type, or the *phaseLabels* argument contains only one item.

ECapeOutOfBounds – One or more of the entries in the *values* argument is outside of the range of values accepted by the Material Object, for example, negative K-values.

EcapeFailedInitialisation - The pre-requisites are not valid. The Phases referenced have not been created using SetPresentPhases.

### **3.8 Can (Pseudo-)Component Properties be set by a Property Calculator?**

Issue: The idea was to use the Property Calculator (PC) for changing the properties of pseudocomponents and so to avoid the use of the Petroleum Fraction Interface.

Answer: It is not possible to set pseudocomponents properties dynamically by a PC. Typically a PC would get the required compound properties when it is instantiated and keep them fixed for its lifetime. Instead the petroleum fractions interface should be used for changing properties during the object's lifetime.

## 4. Errata

### 4.1 Section 5.5 page 25

“A second Material Object constructed by the Physical Property Package act as intermediary between the Property Package and the Property Calculator. According to this diagram, the lifetime of **the Material Objects** is very short; it only exists for as long as the object that creates it needs to communicate with the external component.”

should be

“A second Material Object constructed by the Physical Property Package acts as intermediary between the Property Package and the Property Calculator. According to this diagram, the lifetime of **a Material Object** is very short; it only exists for as long as the object that creates it needs to communicate with the external component.”

### 4.2 Section 5.5 page 115 (units vanderwaals entities)

The units for vanderwaalsArea and vanderwaalsVolume should be  $\text{m}^2/\text{mol}$  and  $\text{m}^3/\text{mol}$  respectively instead of  $\text{m}^2$  and  $\text{m}^3$ .

So the list of pure compound constant properties should end with:

Identifier	meaning	units
.....		
vanderwaalsArea	van der Waals area	$\text{m}^2/\text{mol}$
vanderwaalsVolume	van der Waals volume	$\text{m}^3/\text{mol}$

### 4.3 Section 6.3 ECapeBadInvOrder instead of EcapeBadInvokationOrder

On pages 64, 66, 67, 68, 70, 71, 73, 74 the wrong item ECapeBadInvokationOrder is used instead of the correct one ECapeBadInvOrder.

## **5. Bibliography**

1. “Thermodynamic and Physical Properties interface specification v1.1”, CO-LaN, 2006.