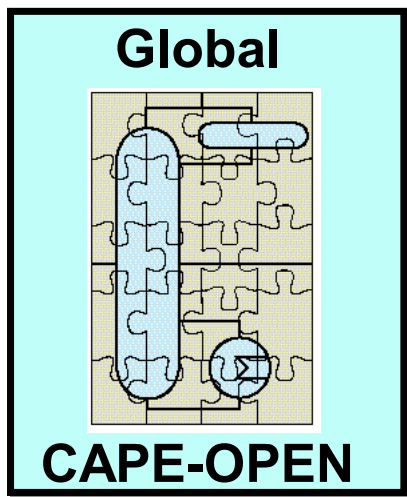


Global CAPE-OPEN

Delivering the power of component software
and open standard interfaces
in computer-aided process engineering



Update of the CAPE-OPEN Thermodynamic and Physical Properties Open Interface Specification

Daniel Piñol, Hyprotech
Juan Carlos Rodriguez, Hyprotech
Michael Halloran, AspenTech
Werner Drewitz, BASF
Richard Szczepanski, Infochem
Michel Pons, TotalFinaElf
Malcolm Woodman, BP
Peter Banks, BP

CO Thermo 0.90 Spec Update.DOC, 3rd July 2001

Archival Information

Reference	CO Thermo 0.90 Spec Update.DOC
Filename (if different)	
Authors	Daniel Piñol, Hyprotech Juan Carlos Rodriguez, Hyprotech Michael Halloran, AspenTech Werner Drewitz, BASF Richard Szczepanski, Infochem Michel Pons, TotalFinaElf Malcolm Woodman, BP Peter Banks, BP
Date	July 3, 2001
Number of Pages	
Version	
Reviewed by (date)	Reviewed by (name) (greyed out means not yet) Interoperability Task Force
Distribution	
Additional Material	
Location on BSCW	

IMPORTANT NOTICES

Disclaimer of Warranty

Global CAPE-OPEN documents and publications include software in the form of *sample code*. Any such software described or provided by Global CAPE-OPEN --- in whatever form --- is provided "as-is" without warranty of any kind. Global CAPE-OPEN 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 Global CAPE-OPEN project --- remains with you.

Copyright © 2000 Global CAPE-OPEN and project partners and/or suppliers. All rights are reserved unless specifically stated otherwise.

Global CAPE-OPEN is a collaborative research project established under BE 3512 "Industrial and Materials Technologies" (Brite-EuRam III), under contract BPR-CT98-9005

Trademark Usage

Many of the designations used by manufacturers and seller to distinguish their products are claimed as trademarks. Where those designations appear in Global CAPE-OPEN 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 is part of a series of deliverables that together describe how to implement the level of interoperability demonstrated in November 2000 in Amsterdam at both the Global CAPE-OPEN Mid-term Meeting and Hyprotech 2000. It lists the changes that have been proposed to the CAPE-OPEN Thermodynamic and Physical Properties Interface Specification, to remove the ambiguities identified during the preparation of this demonstration. The other deliverables are: the example Physical Property Packages, created by Hyprotech; the CO Tester, created by Work Package 8; the Thermo Wizard; and the video of the demonstration. All are available via the CAPE-OPEN Laboratories Network web site (www.colan.org) and have been produced by the Interoperability Task Force in Work Package 4, unless otherwise indicated. None of the changes listed in this document invalidates the current CAPE-OPEN COM type library (version 0.9.0), which will be updated with the new specification later in the year. All of the changes were implemented in the versions of Aspen Plus and HYSYS that were used for the demonstration.

The document also provides recommendations for developers of CO-compliant thermodynamic components and sockets on how to use these deliverables.

Acknowledgements

Contents

A. Introduction.....	10
B. Implementation Resources.....	10
C. CAPE-OPEN Identifiers.....	11
D. List of Changes to Chapter 2 of CAPE-OPEN Thermodynamic and Physical Properties Interface Specification.....	12
2.5 General Remarks on Usage of Interface Methods.....	12
2.5.1 Convention Note on NULL and Empty.....	12
2.5.2 Argument interpretation of Get/Set/CalcProp Standard Methods.....	12
2.5.2.1 Values format.....	13
2.5.2.2 NULL/EMPTY interpretation.....	13
2.5.2.3 Overall interpretation.....	13
2.5.2.4 How to request information of non existing entities (such as components or phases).....	14
2.5.2.5 Fraction and Flow.....	14
2.5.2.6 Get/SetIndependentVar.....	15
2.5.2.7 Specific properties.....	16
2.5.2.8 Validity Check.....	16
2.7 CAPE-OPEN Calling Pattern & Material Object.....	16
2.10 Code Sample of CAPE-OPEN Calling Pattern & Material Object.....	17
2.10.1 Declare Material Object.....	17
2.10.2 Example 2: Calling a flash and then calculating a viscosity:.....	18
2.11.2 ICapeThermoMaterialObject.....	20
2.11.2.3 GetUniversalConstant.....	20
2.11.2.4 GetComponentConstant.....	20
2.11.2.5 CalcProp.....	21
2.11.2.6 GetProp.....	21
2.11.2.7 SetProp.....	22
2.11.2.13 RemoveResults.....	23
2.11.4 ICapeThermoPropertyPackage.....	23
2.11.4.1 GetComponentList.....	23
2.11.4.2 GetUniversalConstant.....	24
2.11.4.3 GetComponentConstant.....	24
2.11.4.5 CalcEquilibrium.....	25
2.11.6. ICapeThermoEquilibriumServer.....	25
2.11.6.1 CalcEquilibrium.....	25
2.11.6.2 PropCheck.....	26
2.14. CAPE-OPEN Properties List.....	27
2.14.1 Constant Properties Identifiers.....	27
Universal constant properties.....	28
2.14.2 Non-constant Properties (or Model Dependent Properties).....	29
Derivatives:.....	30
2.15 CAPE-OPEN Phase List.....	33
2.15.1 Phase Details.....	33
E. SI Units.....	34
F. Notes.....	35
Notes on Configuration of a Material Template.....	35
Notes on 2.10.3 (Convention Note on NULL and Empty).....	35
Notes on 2.10.4 (Argument interpretation of Get/Set/CalcProp Standard Methods).....	35
Non-constant properties.....	35
Notes on 2.11.2.5 CalcProp description.....	35
CalcProp and CalcEquilibrium.....	35
Multiple calculations.....	35
Side-effects during calculation.....	36
Notes on 2.14.2 Non-constant Properties (or Model Dependent Properties).....	36
Notes on 2.14.1 Constant Properties Identifiers.....	36
Components supported by a Property Package.....	36
Notes on 2.11.2.4 GetComponentConstant.....	36

Description of component constants	37
CasRegistryNumber	37
Notes on 2.15 Phases	37
Examples of valid phase equilibrium details.....	37
Existence of a phase.....	38

List of Figures

A. Introduction

This document is part of a series of deliverables that together describe how to implement the level of interoperability demonstrated at the Global CAPE-OPEN Mid-term Meeting in Amsterdam in November 2000. It lists the changes that have been proposed to the CAPE-OPEN Thermodynamic and Physical Properties Interface Specification, to remove the ambiguities identified during the preparation of this demonstration. The other deliverables are: the example Physical Property Packages, created by Hyprotech; the CO Tester, created by Work Package 8; the Thermo Wizard; and the video of the demonstration. All are available from the CAPE-OPEN Laboratories Network (CO-LaN) web site (www.colan.org) and have been produced by the Interoperability Task Force in Work Package 4, unless otherwise indicated.

The Interoperability Demonstration at the Mid-term Meeting showed both unit operation and physical property components being used for steady-state simulation in an industrial flowsheet in Aspen Plus version 10.2-1 (Service Pack 1 or Update 1), plus CAPE-OPEN Hotfix and HYSYS.Process version 2.2 (build 3797), plus HYSYS.CAPE-OPEN 1.03 (build 236). The level of functionality demonstrated is sufficient for useful development work to be possible, but does not yet include error handling or design specifications. This is thus an interim document, which will be superseded when the next versions of Aspen Plus, HYSYS and the CO Thermodynamic and Physical Properties Interface specification are issued later in the year. In the meantime, it provides a useful introduction to the mechanics of implementing a CAPE-OPEN (CO) component. For those who have already been involved in CO component development, none of the changes listed in this document invalidates the current type library (version 0.9.0), which will be updated with the new specification later in the year.

Implementation of a unit component is not dealt with in this document. It is covered by the CAPE-OPEN Unit Operations Interface Specification, plus the example components used in the Interoperability Demonstration (see www.colan.org). Both will be modified later in the year to handle errors and design specifications, but for the moment satisfactorily represent what was implemented in the demonstration.

The remainder of this document recommends an approach to the development of new property components and sockets, lists the changes proposed to the Thermodynamic and Physical Properties Interface and discusses the reasons for the changes in a Notes section at the end.

B. Implementation Resources

We have the following resources available from the CO-LaN web site (www.colan.org) for developers of physical property components and sockets:

1. Physical Property Components

Resources:

- CAPE-OPEN Thermodynamic and Physical Property Interface Specification (CO THRM spec)
- This update document to the CO THRM spec
- The Hyprotech Physical Property Packages (HT PP's) in VB and C++

- The CO Tester
- The Thermo Wizard
- The demonstration video, which shows the current look and feel of open simulation interoperability

We recommend using the HT PP's as templates, with the two specification documents as references to resolve any difficulties. The C++ HT PP is more limited in scope, so is clearer to follow initially. The VB version provides a fuller example of the range of facilities possible. The CO Tester provides a means of testing that the new component conforms to CAPE-OPEN standards and the Thermo Wizard provides an alternative means of generating a thermo component.

In addition to these specific resources, we have background information on COM implementation in a document called "COM Architecture Overview and Basic Principles". See the CO-LaN web site (www.colan.org) for current status and availability of all of these implementation resources.

2. Physical Property Sockets

The same resources are available for socket developers. In this case, the first thing required is to develop a Material Object. A suitable example is in the CO Tester and the HT PP's provide a means to test if the socket works correctly.

C. CAPE-OPEN Identifiers

In this document, we are following CO Methods and Tools recommendations for property identifiers and method names, i.e. property identifiers start with a lower case letter and method names start with an upper case letter. (See: www.colan.org/archive/specs/v09x/doc/03_CO_Methods_and_Tools_Recommendations.pdf)

However, for actual implementation purposes, to avoid uppercase/lowercase problems, all comparisons of CAPE-OPEN identifiers are case insensitive. Namely:

- Properties
- Phase details
- Flash type details
- Calculation type details

We recommend that this should be adopted as CO policy.

D. List of Changes to Chapter 2 of CAPE-OPEN Thermodynamic and Physical Properties Interface Specification

Note: The section numbers used in this chapter refer to those in the original CAPE-OPEN Thermodynamic and Physical Properties Interface Specification. The descriptions below supersede those in the original specification.

(See: http://www.colan.org/archive/specs/v09x/doc/04_CO_Thermodynamics_and_PhysProps.pdf)

The title of section 2.5 has been changed as shown below.

2.5 General Remarks on Usage of Interface Methods

2.5.1 Convention Note on NULL and Empty

(See [Notes](#) for more information.)

Throughout the document NULL refers to a NULL BSTR for string arguments. emptyVariant is a VARIANT argument of the type VT_EMPTY. See below how to implement these values in C++ and in VB.

Type of Data	Declaration and Usage
BSTR	<p>In C++</p> <p>Acceptable <code>BSTR strArg = NULL;</code></p> <p>Not Acceptable <code>BSTR str = ::SysAllocString(L"");</code></p> <p>In VB</p> <p>Acceptable <code>Dim strArg as String strArg = vbNullString</code></p> <p>Not Acceptable <code>Dim strArg as String strArg = ""</code></p>
VARIANT	<p>In C++ // the vt type of the VARIANT is // set to VT_EMPTY VARIANT VarArg; VariantInit(&VarArg); Remember that VariantInit must always be called after declaring a VARIANT</p> <p>In VB <code>Dim VarArg as Variant</code></p>

2.5.2 Argument interpretation of Get/Set/CalcProp Standard Methods

See [Notes](#) for more information.

The original specifications of the GetProp and SetProp methods could be interpreted in several ways. Below is the agreed interpretation of the methods for each property

2.5.2.1 Values format

Although the values argument has VARIANT type in COM and some properties will always return a single value, this argument must always contain an array (possibly with a single element).

2.5.2.2 NULL/EMPTY interpretation

NULL/Empty is only used when one of the arguments is irrelevant for the particular method, such as phases for the Temperature property.

NULL/Empty is never allowed in the property/ies, phases or calcType qualifiers, except for properties fraction, flow, phaseFraction and totalFlow, where calcType should be NULL/Empty.

In general, NULL/Empty cannot be used to express a default value. For instance, someone could make the assumption that basis="" means "mole". We suggest NOT allowing NULL for default values when there are several alternatives for the value (like "mass" or "mole"). Therefore, the NULL/Empty value can only be used when this qualifier has no sense for the requested property.

2.5.2.3 Overall interpretation

If Phases contains "Overall", only the properties that refer to the overall phase will be calculated. To request the values for each particular phase, the particular identifiers of all the phases must be included in the phases argument.

Property	Phases	Comp ID vector	Calculation type	Return value
Temperature Pressure etc.	Typically Overall. (&)	Empty	Mixture	Scalar (1)
Enthalpy Entropy volume density viscosity thermal cond. HeatCapacity		Empty	Mixture	Scalar
		Filled	Pure	Vector (2:all components)
fugacityCoefficient diffusivitycoeff activitycoeffi.		Empty	Mixture(*)	Vector (3: specified comps)
		Filled		Vector (all components)
		Empty	Pure	Vector (specified comp)
		Filled		Vector (all components)
		Empty	Pure	Vector (all components)
		Filled		Vector (specified comp)

Table comments

1. As stated in the specifications, the value is always an array. Scalar means here that the array will only contain 1 value
2. All components means the value of the property for each component of the material object

3. The value of the property for each component specified in argument compIds.

(&) Other phases also allowed, since in the case of a multiphase system which is not thermodynamically in equilibrium (ex: rate-based approach on a distillation column stage), phases of a given system may be at different temperatures

(*)Normally, mixture means that the value is a single scalar that refers to the whole fluid, and pure means that the value is a list of property values for each component. In these particular properties (fugacityCoefficient, . . .), mixture/pure has a special meaning. In them, ‘mixture’ means that the property values refer to the components when they are within a fluid, and “pure” means that the property values refer to the components when they are in pure state (not mixed with other components). Although it could sound appealing to use GetComponentConstant instead of GetProp with “pure”, it is not the case because these properties depend on the physical conditions of the fluid: temperature, pressure,...

2.5.2.4 How to request information of non existing entities (such as components or phases)

Although the CAPE-OPEN standard allows setting a property for a particular phase with the ICapeThermoMaterialObject SetProp method, the Unit Operation developers must be aware that some COSEs may not support this functionality. It can also happen that a CAPE-OPEN-compliant Simulator Executive (COSE) allows setting a property for a particular phase, but only after flashing the stream.

If any GetProp is called for a particular phase where the phase has not been created, the method call will always fail. To check whether a phase exists or not, it is not safe to use the properties totalFlow and phaseFraction. That is because, at bubble point condition, phaseFraction (with phase=vapor), will be identically zero, although the phase exists.

There's another similar case where a GetProp call will fail. For example, imagine a particular stream, where only the molar/mass fractions for some (but not all) of the components of the stream have been set. If the molar fractions are then requested for all the components, the call will fail. That's because, even if the values for some components are available, not all of them can be returned. Therefore, it is recommended to always set the molar/mass fractions for all the components at the same time, assigning a 0 value for the non-existing components.

2.5.2.5 Fraction and Flow

The original specification was ambiguous, since it mentioned properties fraction, flow, molFraction, molFlow. To solve that, the following scheme was agreed.

The Thermo specification mentions “fraction”, but the list of official properties does not. We have agreed to remove molFraction, molFlow & massFlow, leaving only “flow” and “fraction” and forcing the use of the basis argument. This means that now mass fraction can be requested. Since it was not clear how to specify properties such as the total flow or the vapour fraction, two new properties have been added: totalFlow and phaseFractions.

Property	Calc Type	Comp ID	Phase	Return Value
fraction	NULL	Empty	PH	Vector (all components)
flow		Filled	PH	Vector (specified components)
totalFlow	NULL	Empty	PH	Scalar
phaseFraction	NULL	Empty	PH	Scalar

Note: PH means that phase is defined.

Examples:

- To get a vector with the mole fraction of each component within the liquid phase. The sum of all values will be 1.

```
matObj.GetProp("fraction", "liquid", Empty, NULL, "mole")
```

- To get a vector with the mole fraction of some components within the vapor phase. The sum of all values will be the fraction that the set of specified components represent with respect to the whole vapor phase.

```
matObj.GetProp("fraction", "vapor", (vector of components), NULL, "mole")
```

- To get a scalar with the fraction of the fluid that is in the specified phase.

```
matObj.GetProp("phaseFraction", "vapor", Empty, NULL, "mole")
```

- To get a vector with the molar flows of each component within the liquid phase. The sum of all values should be equal to the total molar flow of the fluid

```
matObj.GetProp("flow", "liquid", Empty, NULL, "mole")
```

- To get a vector with the molar flow of some components within the vapor phase.

```
matObj.GetProp("flow", "vapor", (vector of components), NULL, "mole")
```

- To get a scalar with the flow (e.g. molar flow) of the fluid that is in the specified phase.

```
matObj.GetProp("totalFlow", "vapor", Empty, NULL, "mole")
```

2.5.2.6 Get/SetIndependentVar

In the ICapeThermoMaterialObject interface, since the advantage of using the methods Get/SetIndependentVar instead of GetProp and SetProp is not clear, they should be not be used. They will be removed in the next version of the Specification. Moreover, since they lack the basis argument, the units could be ambiguous for state variables such as enthalpy and flow (molflow & massflow have been removed from the standard).

Without Get/SetIndependentVar, the properties listed in "names of state variables/global variables" (p86 of original specification), namely:

temperature	gibbsFreeEnergy
pressure	helmholtzFreeEnergy
volume	MolFraction (deleted)
density	moles
enthalpy	mass
entropy	Molflow (deleted)
energy	Massflow (deleted)

have been moved to the [Non-constant Properties](#) list, so that all of them can be used in Calc/Set/GetProp

Since in COM it's not feasible to remove a method from an interface and retain binary compatibility, the methods will remain, but must not be used.

2.5.2.7 Specific properties

So far, several examples and implementations have used the following property identifiers:

enthalpy, entropy, energy, gibbsFreeEnergy, helmholtzFreeEnergy, volume

to mean intensive properties (which require a specifying “mole” or “mass” for the basis qualifier).

To represent the non-intensive property, such as the energy contained in a given amount of mass, the basis qualifier must have NULL value.

2.5.2.8 Validity Check

The ValidityCheck methods must not be used, since the ICapeThermoReliability interface is not yet defined.

2.7 CAPE-OPEN Calling Pattern & Material Object

The following pattern details the way in which the Material Object is used to execute calls on the corresponding Thermo System. This means that the way in which values are set, calculated and retrieved is consistent for all Thermo System components. This pattern is documented below and detailed in code examples later in the document.

- Step 1: Declare Material Object and set Independent Variables

Independent variables are set on the Material Object for Flash calculations, using the SetProp method. In most cases, two state variables of the material object will be set, for example temperature or pressure.

- Step 2: Set Values

The client of the Material Object adds properties and corresponding values.

- Step 3: Calculate

The appropriate calculations are set as strings on the parameter list and the appropriate generic calculation routine is called.

CalcEquilibrium
CalcProp

- Step 4: Get Results

After results are calculated, the values are then retrieved from the Material Object using the generic Material Object GetProp method. Results are further qualified for phase, components, calculation type, and basis. Property results are in SI units (detailed information available at http://www.bipm.fr/enus/3_SI/si.html).

Specific code examples of this pattern are included in this document and are provided by Werner Drewitz of BASF.

The CAPE-OPEN calling pattern significantly reduces the complexity of the integration with existing native Thermo Systems by reducing the number of calls to the Open System Components. It also allows for the interfaces and contracts between these systems to be modified without addressing software issues. See notes on [Configuration of a Material Template](#) for more information.

2.10 Code Sample of CAPE-OPEN Calling Pattern & Material Object

The following pseudo code example is detailed for the Material Object. However, the pattern by which values are set, calculated and retrieved is consistent for all Thermo System components. Example 1: Calling of liquid enthalpy property of a mixture

2.10.1 Declare Material Object

```
//create a material object  
Set Imo = MaterialTemplate.CreateMaterialObject();
```

□ Step 1: Set Values

```
CapeArrayDouble T[1], P[1], F[100];  
CapeArrayString phaseQualifiers[1];  
  
T[0] = 373; // initialize temperature  
P[0] = 101325; // initialize pressure  
  
F[0] = 0.1; // initialize liquid composition  
F[1] = 0.7; // initialize liquid composition  
F[2] = 0.2; // initialize liquid composition  
  
Strcpy(phaseQualifiers[0], "Liquid"); // set phase qualifier  
  
Imo.SetProp("temperature", "Overall", emptyVariant, NULL, NULL, T);  
Imo.SetProp("pressure", "Overall", emptyVariant, NULL, NULL, P);  
// set temperature and pressure on the material object  
Imo.SetProp("fraction", phaseQualifiers, emptyVariant, NULL, "mole", F);  
  
// set liquid composition on the material object
```

□ Step 2: Calculate Mixture Property

```
CapeArrayString properties[2]; //create array for properties.  
CapeString calculationType;  
  
Strcpy(calculationType, "Mixture"); // set calculation type  
  
Strcpy(properties[0], "Enthalpy"); // set property identifier  
  
Imo.CalcProp(properties, phaseQualifiers, calculationType);  
//calculate properties
```

□ Step 3: Get Results

```
CapeArrayDouble val[2]; //double array created.  
CapeString basisQualifier;  
  
Strcpy(basisQualifier, "mole"); // set basis qualifier  
  
Imo.GetProp("Enthalpy", phaseQualifiers, emptyVariant, calculationType,  
basisQualifier, val);  
//get property enthalpy in the liquid phase
```

2.10.2 Example 2: Calling a flash and then calculating a viscosity:

```
//Declare Material Object
Set Imo      = MaterialTemplate.CreateMaterialObject();
//create a material object
```

□ Step 1: Set Values

```
CapeString phaseQualifiers[1];

T[0] = 373;           // initialize temperature
P[0] = 101325;       // initialize pressure

F[0] = 0.1;          // initialize overall composition
F[1] = 0.7;          // initialize overall composition
F[2] = 0.2;          // initialize overall composition

Strcpy(phaseQualifiers[0], "Overall"); // set phase qualifier

// set temperature, pressure and composition on the material object

Imo.SetProp("temperature", phaseQualifier, emptyVariant, NULL, NULL, T);
Imo.SetProp("pressure", phaseQualifier, emptyVariant, NULL, NULL, P);
Imo.SetProp("fraction", phaseQualifiers, emptyVariant, NULL, "mole", F);
```

□ Step 2 : Calculate Flash

```
CapeString flashTypeQualifier;

// set flash type qualifier
Strcpy(flashTypeQualifier, "TP");

// call equilibrium server, no additional calculation of further
// properties ("NULL")

Imo.CalcEquilibrium(FlashType, emptyVariant);
```

□ Step 3: Calculate Viscosity

```
CapeString calculationType;
// set calculation type
Strcpy(calculationType, "Mixture");

// set phase qualifier
Strcpy(phaseQualifier, "Liquid");

//calculate viscosity
Imo.CalcProp("viscosity", phaseQualifier, calculationType);
```

□ Step 4: Get Results

```
CapeDoubleArray val;  
//double created.  
  
Imo.GetProp("viscosity", phaseQualifier, emptyVariant, calculationType,  
basisQualifier, val);  
  
//get property viscosity for the liquid phase from the  
//Material Object.
```

2.11.2 ICapeThermoMaterialObject

2.11.2.3 GetUniversalConstant

Interface Name ICapeThermoMaterialObject
Method Name GetUniversalConstant
Returns CapeError

Description

Retrieves universal constants from the Property Package.

Arguments

Name	Type	Description
[in] props	CapeVariant (String Array)	List of universal constants to be retrieved
[out, retval] *propvals	CapeArrayVariant	Values of universal constants

2.11.2.4 GetComponentConstant

Interface Name ICapeThermoMaterialObject
Method Name GetComponentConstant
Returns CapeError

Description

Retrieve component constants from the Property Package. See [Notes](#) for more information.

Arguments

Name	Type	Description
[in] props	CapeVariant (String Array)	List of component constants
[in] compIds	CapeVariant (String Array)	List of component IDs for which constants are to be retrieved. emptyVariant for all components in the Material Object.
[out,retval] *propvals	CapeVariant (Variant Array)	Component Constant values returned from the Property Package for all the components in the Material Object It is a variant containing a 1 dimensional array of variants. If we call P to the number of requested properties and C to the number requested components the array will contain C*P variants. The C first ones (from position 0 to C-1) will be the values for the first requested property (one variant for each component). After them (from position C to 2*C-1) there will be the values of constants for the second requested property, and so on.

2.11.2.5 CalcProp

The basic description of this method has not changed, but extensive explanations have been provided to remove ambiguities. See [2.10.4](#) for a more detailed explanation of the arguments and [CalcProp description](#) in the notes for section [2.11.2.5](#) for a general discussion of the method.

2.11.2.6 GetProp

Interface Name **ICapeThermoMaterialObject**
Method Name GetProp
Returns CapeError

Description

This method is responsible for retrieving the results from calculations from the MaterialObject. See [2.10.4](#) for a more detailed explanation of the arguments.

Arguments

Name	Type	Description
[in] property	CapeString	The Property for which results are requested from the MaterialObject.
[in] phase	CapeString	The qualified phase for the results.
[in] compIds	CapeVariant (String Array)	The qualified components for the results. emptyVariant to specify all components in the Material Object. For mixture property such as liquid enthalpy, this qualifier is not required. Use emptyVariant as place holder.
[in] calcType	CapeString	The qualified type of calculation for the results. (valid Calculation Types: Pure and Mixture)
[in] basis	CapeString	Qualifies the basis of the result (i.e., mass /mole). Default is mole. Use NULL for default or as place holder for property for which basis does not apply.
[out, retval] *results	CapeVariant (Double Array)	Results vector containing property values in SI units arranged by the defined qualifiers.

2.11.2.7 SetProp

Interface Name ICapeThermoMaterialObject
Method Name SetProp
Returns CapeError

Description

This method is responsible for setting the values for properties of the Material Object. See [2.10.4](#) for a more detailed explanation of the arguments.

Arguments

Name	Type	Description
[in] property	CapeString	The property for which the values need to be set.
[in] phase	CapeString	Phase, if applicable. Use NULL for place holder.
[in] compIds	CapeVariant (String Array)	Components for which values are to be set. emptyVariant to specify all components in the Material Object. For mixture property such as liquid enthalpy, this qualifier is not required. Use emptyVariant as place holder.
[in] calcType	CapeString	The calculation type. (valid Calculation Types: Pure and Mixture)
[in] basis	CapeString	Qualifies the basis (mole / mass).
[in] values	CapeVariant (Double Array)	Values to set for the property.

2.11.2.13 RemoveResults

Interface Name ICapeThermoMaterialObject
Method Name RemoveResults
Returns CapeError

Description

Remove all or specified property results in the Material Object.

Arguments

Name	Type	Description
[in] props	CapeVariant (String Array)	Properties to be removed. emptyVariant to remove all properties.

2.11.4 ICapeThermoPropertyPackage

2.11.4.1 GetComponentList

Addition to the description:

In order to identify the components of a Property Package, the Executive will use the 'casno' argument instead of the compIds. The reason is that different COSEs may give different names to the same chemical compounds, whereas CAS Numbers are universal. Nevertheless, GetProp/SetProp... will still require their compIds argument to have the usual contents ("hydrogen","methane",...). Be aware that some simulators may have a limitation on the length of the names for pure components. Hence, it is recommended that each identifier returned by the compIds argument should not contain more than 8 characters. See notes on [Description of component constants](#) for more information.

If the package does not return a value for the 'casno' argument, or its value is not recognised by the Executive, then the compIds will be interpreted as the component's English name: such as "benzene", "water",... Obviously, it is recommended to provide a value for the casno argument.

The same information can also be extracted using the ICapeThermoPropertyPackage GetComponentConstant method, using the casRegistryNumber property identifier.

2.11.4.2 GetUniversalConstant

Interface Name ICapeThermoPropertyPackage
Method Name GetUniversalConstant
Returns CapeError

Description

Returns the values of the Universal Constants.

Arguments

Name	Type	Description
[in] *materialObject	CapeInterface	The Material object.
[in] props	CapeVariant (String Array)	List of requested universal constants.
[out,retval] *propvals	CapeArrayVariant	Values of universal constants.

2.11.4.3 GetComponentConstant

Interface Name ICapeThermoPropertyPackage
Method Name GetComponentConstant
Returns CapeError

Description

Returns the values of the Constant properties of the components contained in the passed Material Object.

Arguments

Name	Type	Description
[in] *materialObject	CapeInterface	The material object.
[in] props	CapeVariant (String Array)	The list of properties.
[out,retval] *propvals	CapeVariant (Variant Array)	Component Constant values. See description of return value of the ICapeThermoMaterialObject GetComponentConstant method.

2.11.4.5 CalcEquilibrium

Interface Name **ICapeThermoPropertyPackage**
Method Name **CalcEquilibrium**
Returns **CapeError**

Description

Method responsible for calculating/delegating flash calculation requests.

Arguments

Name	Type	Description
[in] *materialObject	CapeInterface	The MaterialObject
[in] flashType	CapeString	Flash calculation type.
[in] props	CapeVariant (String Array)	Properties to be calculated at equilibrium. emptyVariant for no properties.

2.11.6. ICapeThermoEquilibriumServer

2.11.6.1 CalcEquilibrium

Interface Name **ICapeThermoEquilibriumServer**
Method Name **CalcEquilibrium**
Returns **CapeError**

Description

Calculates the equilibrium properties requested. See [CalcProp and CalcEquilibrium](#) in section F for more information.

Arguments

Name	Type	Description
[in] *materialObject	CapeInterface	MaterialObject of the calculation
[in] flashType	CapeString	Flash calculation type.
[in] props	CapeVariant (String Array)	Properties to be calculated at equilibrium. emptyVariant for no properties.

2.11.6.2 PropCheck

Interface Name **ICapeThermoEquilibriumServer**

Method Name PropCheck

Returns CapeError

Description

Checks to see if a given type of flash calculations can be performed and whether the properties can be calculated after the flash calculation.

Arguments

Name	Type	Description
[in] *materialObject	CapeInterface	The Material Object for the calculations.
[in] flashType	CapeString	Type of flash calculation to check
[in] props	CapeVariant (String Array)	List of Properties to check. emptyVariant for none.
[out, retval] *valid	CapeArrayBoolean	The array of booleans for flash and property. First element is reserved for flashType.

2.14. CAPE-OPEN Properties List

2.14.1 Constant Properties Identifiers

See [Notes](#) for more explanation on the table and [CAPE-OPEN Identifiers](#) for naming conventions. Units of measure can be found in the section on [SI Units](#).

To identify a pure component there are some attributes, which are not really 'properties', but are never the less needed:

Nidentifier	Character string for identification e.g. in a flowsheet or PPS
iupacName	Complete IUPAC Name
casRegistryNumber	Chemical Abstract Sequencing Number
chemicalFormula	Chemical formula (brutto, nomenclature according to Hill)
structureFormula	Chemical structure formula

The name of a component should be the same in the whole flowsheet. There is a need in every flowsheet calculation for a global component list! If one uses different Property Packages, it is very probable that there are different names used for the same component. So a translation list may help in which, for each name of the global component list, there are the names for the correspondent components in the Property Packages being used.

Identifiers	Meaning	SI Units
molecularWeight	Relative molecular mass	
criticalTemperature	Critical Temperature	K
criticalPressure	Critical Pressure	Pa
criticalVolume	Critical Volume	m ³ /mol
criticalCompressibilityFactor	Critical Compressibility Factor	
criticalDensity	Critical Density	mol/m ³
acentricFactor	Pitzer Acentric Factor	
dipoleMoment	Dipole Moment	Cm
parachor	Parachor	m ³ kg ^{0.25} /(s ^{0.5} mol)
gyrationRadius	Radius of Gyration	m
associationParameter	Association-Parameter (Hayden-O'Connell)	
diffusionVolume	Diffusion volume	m ³
diffusionCoefficient	Diffusion coefficient	m ² /s
vanderwaalsVolume	Van der Waals Volume	m ³
vanderwaalsArea	Van der Waals Area	m ²
energyLennardJones	Lennard-Jones Energy	J
lengthLennardJones	Lennard-Jones Length	m
normalBoilingPoint	Temperature at boiling point (1.01325 bar)	K
heatOfVaporizationAtNormalBoilingPoint	Heat of Vaporization at boiling point (1.01325 bar)	J/mol
normalFreezingPoint	Temperature of normal melting point (1.01325 bar)	K
heatOfFusionAtNormalFreezingPoint	Heat of Melting at melting point	J/mol

	(1.01325 bar)	
liquidDensityAt25C	Liquid Density at 25 C	mol/m ³
liquidVolumeAt25C	Liquid Volume at 25 C	m ³ /mol
idealGasGibbsFreeEnergyOfFormationAt25C		J/mol
idealGasEnthalpyOfFormationAt25C		J/mol
standardFormationEnthalpySolid	Standard Formation Enthalpy of Solid	J/mol
standardFormationEnthalpyLiquid	Standard Formation Enthalpy of Liquid	J/mol
standardFormationEnthalpyGas	Standard Formation Enthalpy of Gas	J/mol
standardFreeFormationEnthalpySolid	Standard Free Formation Enthalpy of Solid	J/mol
standardFreeFormationEnthalpyLiquid	Standard Formation Enthalpy of Liquid	J/mol
standardFreeFormationEnthalpyGas	Standard Formation Enthalpy of Gas	J/mol
standardEntropySolid	Standard Entropy of Solid	J/mol
standardEntropyLiquid	Standard Entropy of Liquid	J/mol
standardEntropyGas	Standard Entropy of Gas	J/mol

Standard is at 25 C and 1.01325 bar (= 1 atm).

Universal constant properties

standardAccelerationOfGravity	9.806 65 m s ⁻²
avogadroConstant	6.022 141 99(47) 10 ²³ mol ⁻¹
boltzmannConstant	1.380 6503(24) 10 ²³ J K ⁻¹
molarGasConstant	8.314 472(15) J mol ⁻¹ K ⁻¹

Note: Only the units of measure and the identifiers of the universal constants are specified in the standard, not the values.

2.14.2 Non-constant Properties (or Model Dependent Properties)

See [Notes](#) for more explanation of the revised table and [CAPE-OPEN Identifiers](#) for general naming conventions. Units of measurement can be found in the section on [SI Units](#).

Identifiers	Meaning	SI Units
vaporPressure	Vapor Pressure. Only for Pure calcType	Pa
sublimationPressure	Sublimation Pressure	Pa
meltingPressure	Melting Pressure	Pa
glassTransitionTemperature	Glass Transition Temperature	K
glassTransitionPressure	Glass Transition Pressure	Pa
solidSolidPhaseTransitionTemperature	SolidSolidPhaseTransitionPressure	Pa
triplePointTemperature	Triple Point Temperature	K
triplePointPressure	Triple Point Pressure	Pa
virialCoefficient	Second Virial Coefficient	m ³ /mol
surfaceTension	Surface Tension	N/m
expansivity	coefficient of linear expansion (Expansivity) $\frac{1}{L} \left. \frac{\partial L}{\partial T} \right $	1/K
compressibility	$\frac{1}{V} \left. \frac{\partial V}{\partial P} \right _T$	1/Pa
compressibilityFactor	Compressibility Factor $Z = \frac{PV}{RT}$	
jouleThomsonCoefficient	$\left. \frac{\partial T}{\partial P} \right _H$	K/Pa
heatOfVaporization	**	J/mol
heatOfSublimation	**	J/mol
heatOfFusion	**	J/mol
heatOfSolidSolidPhaseTransition	**	J/mol
volumeChangeUponVaporization	**	m ³ /mol
volumeChangeUponSublimation	**	m ³ /mol
volumeChangeUponMelting	**	m ³ /mol
volumeChangeUponSolidSolidPhaseTransition	**	m ³ /mol
heatCapacity	Heat Capacity (Cp)**	J/(mol K)
idealGasHeatCapacity	Heat Capacity of ideal Gas**	J/(mol K)
idealGasEnthalpy	Enthalpy of ideal Gas*	J/mol
excessEnthalpy	Excess enthalpy*	J/mol
excessEnergy	Excess energy*	J/mol
excessGibbsFreeEnergy	Excess Gibbs Free Energy*	J/mol
excessHelmholtzFreeEnergy	Excess Helmholtz Free Energy*	J/mol
excessEntropy	Excess entropy*	J/(mol K)
excessVolume	Excess volume*	m ³ /mol
partialMolarEnthalpy		J/mol
partialMolarEnergy	Partial molar internal energy	J/mol
partialGibbsFreeEnergy	Partial molar Gibbs energy	J/mol
partialHelmholtzFreeEnergy	Partial molar Helmholtz energy	J/mol
partialMolarVolume		m ³ /mol
viscosity	Viscosity	Pa s
thermalConductivity	Thermal Conductivity	W/(m K)

selfDiffusionCoefficient		m ² /s
fugacity	Fugacity	Pa
fugacityCoefficient	Fugacity Coefficient	
activity	Activity	
activityCoefficient	Activity Coefficient	
dewPointPressure		Pa
dewPointTemperature		K
kvalues	K Factors of a pair of phases in equilibrium	
logFugacityCoefficient	Logarithm of fugacity coefficients	
logkvalues	Logarithm of kvalues	
temperature		K
pressure		Pa
volume	Volume*	m ³ /mol
density	Density **	mol/m ³
enthalpy	Enthalpy*	J/mol
entropy	Entropy*	J/(mol K)
energy	Internal energy*	J/mol
gibbsFreeEnergy	Gibbs Free Energy*	J/mol
helmholtzFreeEnergy	Helmholtz Free Energy*	J/mol
moles	Number of moles of a given amount of matter	mol
mass	Total mass of a given amount of matter	kg
flow	List of the partial molar (or mass) flows of each component within a given phase (or the whole mixture)**	mol/s
fraction	List of the partial molar (or mass) fractions of each component within a given phase (or the whole mixture)	
phaseFraction	The fraction of the fluid that is in the specified phase.	
totalFlow	Matter flow of a phase or the whole mixture**	mol/s
molecularWeight	It is recommended to be used only with CalcType="mixture". For pure, GetComponentConstant is preferred. It is up to the package to calculate it by whatever means it chooses	
boilingPointTemperature	Only supported for "pure" CalcType	K

Notes: * per mole, or kg, or total depending on basis.

** per mole, or kg depending on basis.

Derivatives:

Derivatives are built from the property identifier, a point with a D meaning "Derivative" and the name for the variable:

property.Dtemperature
property.Dpressure

derivative of property according to Temperature
derivative of property according to Pressure

property.DmolFraction derivative of property according to mole fraction
property.Dmoles derivative of property according to mole numbers

Addition to the ICapeThermoPropertyPackage/ICapeThermoMaterialObject.GetPropList interface description:

Although the property identifier of derivative properties is formed from the identifier of another property, the GetPropList method will return the identifiers of all supported derivative and non-derivative properties. For instance, a Property Package could return the following list:

enthalpy, enthalpy.Dtemperature, entropy, entropy.Dpressure.

2.15 CAPE-OPEN Phase List

2.15.1 Phase Details

Permitted phases have been restricted to the following:

Phase	Description
Vapour	Vapour phase
Liquid	Liquid phase
Solid	Solid phase
Overall	All phases

At the moment, only one phase of each type is allowed to be present, in general. See Notes for information on a partial work around and Existence of a phase for information on phase checking.

E. SI Units

The current standards do not specifically state which SI unit has to be used for each dimension. It will be added to the specification documentation in a future revision. For the moment, we have added the units used in the interoperability demonstration implementations to the CAPE-OPEN Properties List in this document. We suggest referring to the *Bureau International des Poids et Mesures* website (http://www.bipm.fr/enus/3_SI/si.html) for more information.

F. Notes

Notes on Configuration of a Material Template

Although the CAPE-OPEN specifications defined the ICapeThermoMaterialTemplate interface, no methodology was defined to access or create instances of objects that implemented these interfaces.

We will add information on using the Material Template interface in future versions of the Specification.

Notes on 2.10.3 (Convention Note on NULL and Empty)

In the original specification, it is impossible to send a NULL as CapeVariant, in order to say that the complete set of components is the set of components on which a property is required. There are two possibilities: either defining the NULL, or not using that default option. So far NULL was defined as empty Variant in the prototypes developed. This means that in the THRM specification, a large number of pages describing the methods have to be changed in order to explain the definition of NULL.

Notes on 2.10.4 (Argument interpretation of Get/Set/CalcProp Standard Methods)

Non-constant properties

There is not enough definition for passing information on vapour fraction, as well as an inconsistency in the specification. There is no way to choose vapour fraction among the state variables, while the specification provides for a pressure-vapour fraction flash. At the moment, Global CAPE-OPEN is not addressing the generalisation of the flash. It could mean changing the attributes of GetIndependentVar and SetIndependentVar in a future version of the specification.

Notes on 2.11.2.5 CalcProp description

CalcProp and CalcEquilibrium

There is no interaction between CalcProp and CalcEquilibrium, 13.d, so CalcProp should never invoke CalcEquilibrium.

CalcProp is used to calculate properties in the specified phase at the current values of T, P and x, it does not perform phase equilibrium calculations.

CalcEquilibrium is used to calculate state variables from others, such as enthalpy, entropy or phase fraction.

Multiple calculations

If a client uses multiple properties in a call and one of them fails then the whole call should be considered to have failed. Therefore, no value should be written back to the material object by the Property Package until it is known that the whole request can be satisfied. For this reason, to

simplify error handling or debugging, it is recommended that clients only request one property at a time to make error handling simpler.

Side-effects during calculation

It is important to note that Property Packages are NOT allowed to calculate and (more important) to store the values of properties that have not been specifically requested.

Notes on 2.14.2 Non-constant Properties (or Model Dependent Properties)

Modifications:

a) Since there exist the “compressibilityCoefficient” and “compressibilityFactor” properties, it is not clear what the “compressibility” property means. On looking in some handbooks (e.g., Perry), compressibility and isothermal compressibility are used to refer to the same quantity. $(1/V)(dV/dP)$ at constant T. So, we suggest:

- Renaming the property identifier “compressibilityCoefficient” to “compressibility” leaving its meaning $(1/V)(dV/dP)$ at constant T
- Removing the “compressibility” property on page 85 of the original specification

b) ‘kvalues’ carries similar information to ‘fugacityCoefficient’, but it’s more logical to publish the value of the former property.

Notes on 2.14.1 Constant Properties Identifiers

The most important role of the component constants is to identify the components supported by a Property Package. The ICapeThermoPropertyPackage GetComponentList method was designed for this purpose.

Components supported by a Property Package

Use GetComponentList for a list of the components supported by a Physical Property Package. See the Glossary of the original Thermodynamic Specification for a definition of a Physical Property Package. It is a specific entity tailored to a specific application, rather than a general physical property system (see the Glossary for a definition of a Physical Property System).

Notes on 2.11.2.4 GetComponentConstant

Equivalences between GetComponentList arguments and component constant properties:

GetComponentList Arguments	Component Constant Property Identifier	Comments
casno	casRegistryNumber	
compIds	--	This string has to be used in all the arguments of the materialObject and Property Package methods which are named compIds.
formulae	chemicalFormula	

Name	iupacName	
BoilTemps	normalBoilingPoint	
Molwt	molecularWeight	

The problem was that “casRegistryNumber” and other properties have values which are not numbers but strings, whereas the specification states that getComponentConstants returns a list of numbers. As stated in the section 2.14.1 of the Thermo Specs, the following constant properties are not supported by the current specification of GetComponentConstant:

iupacName complete IUPAC Name
casRegistryNumber Chemical Abstract Sequencing Number
chemicalFormula Chemical formula (brutto, nomenclature according to Hill)
structureFormula Chemical structure formula

For the same reasons, GetUniversalConstant should also return a CapeArrayVariant.

Description of component constants

CasRegistryNumber

The value of this constant is a variable-length character string that contains a sequence of 3 numbers separated by hyphens. There must be no leading zeros and no leading spaces. The intention is that it should be possible to compare two CAS numbers with a simple string comparison

CAS numbers and other properties are accessible at
<http://webbook.nist.gov/chemistry>

Components can be accessed directly with
<http://webbook.nist.gov/cgi/cbook.cgi?Formula=ch4&Nolon=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.

Notes on 2.15 Phases

The list of phases in 2.15.1 of the original specification assumes that 2 liquid fractions are supported, since, for instance, there is a LiquidLiquid phase detail. However, at present there is no way to refer to each one of the liquid phases separately. For this reason, the permitted phases have been restricted, as shown, pending an improvement to phase treatment in a future update.

Examples of valid phase equilibrium details

VaporLiquid	Vapor-liquid equilibrium
LiquidSolid	Solid-liquid equilibrium
VaporLiquidSolid	3 coexisting phase: vapor, liquid and solid
VaporSolid	Solid-vapor equilibrium, sublimation

However, Type Library 0.9.0 does allow the calculation of single-phase properties for streams with any number of liquid phases, since we can calculate them independently. For example:

```
mo.SetProp("fraction","liquid", ...,liquid1FractionValue)
mo.calcProp("enthalpy","liquid")
mo.GetProp("fraction","liquid", ...,liquid1EnthalpyValue)
```

```
mo.SetProp("fraction","liquid", ...,liquid2FractionValue)
mo.calcProp("enthalpy","liquid")
mo.GetProp("fraction","liquid", ...,liquid2EnthalpyValue)
```

Unfortunately, multi-phase properties are not supported for 2 liquid phases, since the Material Object cannot cope simultaneously with 2 liquid phases. So, calcprop("kvalues","liquidLiquid") is not supported. However, in the particular case of "kvalues", if "fugacityCoefficient" is used instead, it works around the problem, since the phases can be calculated independently:

```
mo.SetProp("fraction","liquid", ...,liquid1FractionValue)
mo.calcProp("fugacityCoefficient","liquid")
mo.GetProp("fraction","liquid", ...,liquid1 fugacityCoefficient)
```

```
mo.SetProp("fraction","liquid", ...,liquid2FractionValue)
mo.calcProp("fugacityCoefficient","liquid")
mo.GetProp("fraction","liquid", ...,liquid2fugacityCoefficient)
```

kvalues = liquid2fugacityCoefficient/liquid1 fugacityCoefficient

Existence of a phase

As already described, that phasefraction or totalflow cannot be used for checking existence of a phase, since for instance in the case of a bubble point both properties would return a 0 value for the vapor phase.

Instead, materialObject.PhaseIDs() must be used, since it returns only the phases existing in the MO at that moment (the COSE knows this information). Note that materialObject.PhaseIDs() does not return the list of phases supported by the Property Package relating to the MO. The latter information is provided by the PropPack.getPhaseList() method.

This approach has the limitation that, currently, the Property Packages don't have any mechanism to change the list phases existing in a material object during the calculation of an equilibrium. Only the COSE could do it. The next version of the standard will add new methods to solve this shortcoming