CAPE-OPEN

Expanding Process Modelling Capability
through Software Interoperability Standards

Errata and clarifications for Thermodynamic Standard specification 1.1

www.colan.org

ARCHIVAL INFORMATION

Filename	Errata_1.1_2.0.0009.doc
Authors	CO-LaN consortium: Thermo SIG
Status	Public document
Date	November 2012
Version	Version 2.0.0009
Number of pages	7
Versioning	Initiated in July 2011 by Thermo SIG (2.0.0001)
	Approved by the Thermo SIG on Sep 20, 2012
	Approved for release by CO-LaN Mgt Board on
Additional material	
Web location	
Implementation specifications version	Version 1.1, update 311
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.

CO-LaN is a non for profit organization established under French law of 1901.

Trademark Usage

Many of the designations used by manufacturers and seller 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.

FOREWORD

The errata and clarifications have been discussed and developed by the Thermo SIG, the CO-LaN Special Interest Group dedicated to CAPE-OPEN Thermodynamic interfaces. This document has been approved for release by CO-LaN Management Board. This Errata and Clarifications document refers explicitely to the interface specification document version 1.1 (update 311 of 10 May 2011) and should be considered as a constituent part of this interface specification document.

SUMMARY

Clarifications are related to the expected behaviour of a Property Package Manager, to the frequency of calls to SetMaterial method, to the request bubble and dew point calculations, to the behaviour of a Material Object around a CalcEquilibrium call, to the order of specifications in a CalcEquilibrium call, to the recommended use of CheckEquilibriumSpec and to the interpretation of UNDEFINED as a CapeString.

ERRATA AND CLARIFICATIONS

1.1 Errata

No errata have been listed.

1.2 Clarifications

1.2.1 GetPropertyPackage

For new instances of Property Packages, the Property Package Manager expects the name of a property package that is defined on the system to be passed to GetPropertyPackage. These names are returned by GetPropertyPackageList.

In case Property Packages can be persisted, the situation is somewhat more complex. The Property Package may have been created at a different computer, or the list of available property packages may have changed since the property package was first created. In this case, GetPropertyPackage returns an uninitialized Property Package, that keeps track of the name that was specified. In case the Property Package had been persisted, a call to Load will follow, and the Property Package receives its configuration data. In case Load is not called, the Property Package still need to receive its configuration data during Initialize. An invalid name passed to GetPropertyPackage leads to an error in Initialize.

So passing an invalid name to GetPropertyPackage may lead to an error immediately for Property Package Managers of which the Property Packages cannot be persisted, or may lead to an error during Intialize if the Property Package if persistence is supported but the Property Package was not loaded from persistence.

In either case, the PME has the responsibility to expose to the user any textual error message provided by the Property Package.

1.2.2 SetMaterial

A PME that uses a Property Package needs to use SetMaterial before any function that requires a Material Object (e.g. CalcSinglePhaseProp, CalcOverallProp, CalcAndGetLnPhi, CalcEquilibrium) only if the Material Object differs from the one passed to the last call to SetMaterial (if it is a different Material Object, or in case the list of compounds on the Material Object has changed). Although it is not an error to call SetMaterial more often (e.g. between property or equilibrium calculations on the same Material Object), it does likely affect performance. Typically, a Property Package will analyse the list of compounds between the last call to SetMaterial and the first operation that requires the list of compounds to be known, which takes a finite amount of time.

1.2.3 Bubble and dew point calculations

In version 1.1 properties can only be calculated for a real phase, not for the “Overall” phase (the “CalcSinglePhaseProperty” method is not applicable to the “Overall” phase). Therefore, the version 1.0

properties “DewPointPressure”, “DewPointTemperature”, “BubblePointPressure” and “BubblePointTemperature” are not supported by the 1.1 specification.

- The recommended method for obtaining the bubble and dew point pressures is to perform a CalcEquilibrium at specified temperature and vapor fraction.

- The recommended method for obtaining the bubble and dew point temperatures is to perform a CalcEquilibrium at specified pressure and vapor fraction.

This approach has the additional advantage that the solution type can be requested, hence distringuishing between the Normal and Retrograde dew and bubble points.

1.2.4 Properties available after CalcEquilibrium

It has been noticed that some Material Object implementations in response to a CalcEquilibrium calculation automatically request calculations of additional properties like Enthalpy from the Property Package. Such calculations result in unnecessary computational overhead.

The properties that should be present on a Material Object after a successfully completed CalcEquilibrium are Overall Pressure, Overall Temperature, and for each of the present phases, Pressure, Temperature, Fraction and PhaseFraction. If a PMC or the PME requires any other properties on the Material Object (e.g. Enthalpy), it is the responsibility of the PMC or the PME to issue the relevant CalcSinglePhaseProperty / CalcTwoPhaseProperty calls on the Material Object. Therefore, the Material Object should not automatically calculate any other properties after a PMC or the PME has requested a CalcEquilibrium to be performed.

1.2.5 Order of equilibrium calculation specifications

The order of the two specification arguments for CalcEquilibrium and CheckEquilibriumSpec is arbitrary. For example, an equilibrium calculation with the specifications of Overall Pressure and Overall Temperature is equivalent to that with the specifications of Overall Temperature and Overall Pressure.

1.2.6 Avoiding unnecessary CheckEquilibriumSpec calls

Efficiency recommendation: to perform a phase equilibrium calculation, CalcEquilibrium should be used and there is no need to call CheckEquilibriumSpec prior to CalcEquilibrium.

The CheckEquilibriumSpec method should be used to establish whether or not a particular phase equilibrium calculation is supported without performing the calculation itself. Typically, a Unit Operation PMC would call CheckEquilibriumSpec from its Validate method, and CalcEquilibrium from its Calculate method.

If a CalcEquilibrium fails, and the caller needs to determine whether it fails because it is not supported or whether it fails due to other reasons, such as non-convergence, one can call CheckEquilibriumSpec after having called the (failed) CalcEquilibrium.

1.2.7 UNDEFINED interpretation for a CapeString

For a CapeString, it is advised to check for both NULL and an empty string when testing for UNDEFINED if COM is used.

UNDEFINED is not the string literal “UNDEFINED”.

UNDEFINED is defined for several data types in section 7.3, however, not for a CapeString. For a CapeString, a NULL BSTR value is meant when using COM. Note that in some cases an empty BSTR is automatically produced, such as by assignment of a vbNullString to a String variable in Visual Basic 6.0.