CAPE-OPEN

Expanding Process Modelling
Capability through Software
Interoperability Standards

Open Interface Specification:

Custom Data Interface

www.colan.org

ARCHIVAL INFORMATION

Filename	CustomDataInterfaceSpec.docx
Authors	CO-LaN consortium
Status	Internal
Date	November 2019
Version	Version 1.1.0
Number of pages	31
Versioning	Version 0.00 by Jasper van Baten, June 2015
	Version 0.20 submitted for RFC in September 2018
	Version 0.23 edited on October 20, 2019 by Thermo SIG
	Version 0.24 edited on October 21, 2019 by Thermo SIG
	Version 1.1.0 proposed for public release to Management Board on November 8, 2019
Additional material	
Web location	www.colan.org
Implementation specifications version	1.1
Comments	TLB/PIA installer release 2.1.0.60, November 2019 and newer

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.

Adobe Acrobat is a registered trademark of Adobe Corporation.

SUMMARY

This document contains the specification of interfaces enabling any subclass of Process Modelling Components implementing *ICapeThermoMaterialContext*, to store and retrieve custom data from a Material Object. The new interfaces manage the lifespan of custom data by storing it on Material Objects, the number of which is unknown to the Process Modelling Component.

ACKNOWLEDGEMENTS

Many individuals and organizations have contributed to this document. The following are among the main contributors:

Jasper van BATEN	AmsterCHEM
Sergej BLAGOV	BASF SE
Mark STIJNMAN	Shell Global Solutions
Michel PONS	CO-LaN

CONTENTS

LIST OF FIGURES

CAPE-OPEN Document Roadmap

This document is a specification document for a set of business interfaces and belongs to the documentation set of CAPE-OPEN interface specifications in its version 1.1.

The document describes an interface on a Material Object and an interface on various classes of Process Modelling Components implementing *ICapeThermoMaterialContext*, referred to as Custom Data Sources. Consequently, readers should refer to the Thermodynamic and Physical Properties Interface specification document (i) for further information.

1. Introduction

A Property Package will often receive calculation requests that are related to previous requests and may require the same intermediate calculation results to be present. For example, an electrolyte Property Package may calculate the true composition of the aqueous phase as part of a phase equilibrium calculation. Sometime later the Property Package may be asked to calculate entropy for the aqueous phase at the same temperature, pressure and composition, which requires the true composition to be known. Although the Property Package can re-calculate the true composition, this calculation may be computationally expensive. In order to speed up the calculations in such situations, it may be beneficial to store intermediate results, like the aqueous true composition, between calculation requests, and avoid recalculation. To facilitate this, the Custom Data interface specification is introduced.

The Property Package could in principle also internally cache Custom Data such as true composition. However, the difficulty with this approach is that the Property Package has no information on how many Material Objects will be associated with the Property Package nor information on the lifespan of any associated Material Object. This makes an internal cache of custom data inefficient on two accounts: firstly, it might be time-consuming for the Property Package to match the Material Object to a cache structure and, secondly, the Property Package has no way to know if and when the data stored is no longer relevant. Storing the custom data directly on the Material Object resolves both issues.

Any Process Modelling Component supporting *ICapeThermoMaterialContext* (Equilibrium Servers, Chemical Reaction Packages, etc...) can store any intermediate result on a Material Object: for example, a Property Package may store the solution to the equation of state for use in subsequent calculations.

Custom Data is advantageous to all PMCs that do not know in advance with how many Material Objects they will be associated. PMCs that deal with a known set of Material Objects are not considered in this document. For example, a Unit Operation will see typically one Material Object on each of its material Ports, and a Flowsheet Monitoring Component will know about all Material Objects in a flowsheet. Such PMCs can internally store any information particular to any Material Object and the proposed Custom Data interface specification would therefore not be useful to such PMCs.

1.1 Audience

This document is solely intended for PME software developers who want to extend the Material Object functionality, and for PMC software developers who want to take advantage of this functionality. Familiarity with CAPE-OPEN basic concepts is presumed.

For any reader an understanding of UML diagrams is assumed.

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

1.2 Special Glossary Terms

Custom Data

Custom Data usually represent private calculation results that can always be recalculated. Custom Data are stored on a Material Object by a Custom Data Source in order to possibly speed up subsequent calculations.

Custom Data Source (CDS)

A Process Modelling Component which deals with any number of Material Objects (such as Property Packages and Chemical Reaction Packages), implementing *ICapeThermoMaterialContext* and stores custom data on Material Objects. The Custom Data Source does not know the number or lifespan of the Material Objects it is associated with.

Custom Data Container (CDC)

The object that stores the Custom Data for the Custom Data Source (CDS). The Custom Data Container is created by the Custom Data Source and stored on the Material Object.

2. Requirements

2.1 Textual Requirements

Scope

This design applies to Custom Data representing intermediate calculation results that can, if required, always be re-calculated by the Custom Data Source.

Custom Data depend on the state of the material (e.g. temperature, pressure and composition). Custom Data Source (CDS) is responsible for checking whether the stored Custom Data are usable or not. If, for example, temperature has changed, Custom Data may need to be discarded and recalculated.

The interface design does not provide any mechanism for the Material Object to invalidate the stored Custom Data, except when Material Object configuration changes.

Custom Data is carried into a new CAPE-OPEN object, a Custom Data Container (CDC) which is secondary to a PMC. Therefore, a Custom Data Container must follow the requirements imposed on any CAPE-OPEN secondary object (iii).

Requirements

REQ-001-CDS: Custom Data Source must implement *ICapeThermoMaterialContext*.

Rationale: the number and lifespan of Material Objects are unknown to the Custom Data Source. Therefore, using *ICapeThermoMaterialContext*, the PME sets the active Material Object on which the Custom Data Source is operating.

REQ-002-CDS: content of the Custom Data Container is managed only by the Custom Data Source.

Rationale: the contents of the Custom Data Container are private to the Custom Data Source. No interface is defined to access (read/write) the data content.

REQ-003-PME: multiple Custom Data Sources can store data on a Material Object.

Rationale: multiple Custom Data Sources, e.g. a Property Package and a Chemical Reaction Package, may be present that want to use Custom Data support provided by the Material Object. The PME is aware of which PMCs may be associated to any given Material Object.

REQ-004-PME: the PME must discard any Custom Data Container if the thermodynamic configuration of the Material Object may have changed.

Rationale: it is reasonable to assume that a change in thermodynamic configuration makes Custom Data no longer valid. It is not possible for any Custom Data Source to verify that the current thermodynamic configuration of the Material Object is the same as the configuration at the time the Custom Data was saved. Therefore, the PME must remove any Custom Data on any Material Object for which the thermodynamic configuration has changed. This includes changes in thermodynamic calculation routines, compound and phase selections, reaction data, etc…

Generally the PME should assume that if a Material Object is associated with a Property Package or a Chemical Reaction Package, the configuration has changed if the Property Package or the Chemical Reaction Package has been modified through *ICapeUtilities::Edit* (unless *Edit* returns S_FALSE). Also, in case the sub-selection or the order of compounds on the Material Object changes, the Custom Data Container must be removed by the PME.

The PME must assume that configuration of the Custom Data Source may have changed between sessions. Therefore, Custom Data Containers must not be persisted between sessions.

REQ-005-CDS: if the Custom Data Source stores a reference to the Custom Data Container belonging to the active Material Object between calls to *ICapeThermoMaterialContext::SetMaterial*, the Custom Data Source must drop this reference upon executing *SetMaterial*.

Rationale: it is allowed for the Custom Data Source to keep a reference to the Custom Data Container between calls to *ICapeThermoMaterialContext::SetMaterial* on the Custom Data Source.

REQ-006-CDS: if the Custom Data depend on Material Object conditions, the Custom Data Source must internally validate these conditions before using the Custom Data.

Rationale: the applicability of Custom Data may depend on Material Object conditions such as temperature, pressure and phase composition. For example, Custom Data stored by the Property Package may consist of true composition matching the chemical equilibrium at given temperature, pressure and overall composition. Therefore, the Custom Data are only useable if the conditions the Custom Data depend on have not changed. No mechanism is defined for the PME to remove the Custom Data in case of a change in Material Object conditions because the PME is not aware of which conditions invalidate the Custom Data. The Custom Data Source is responsible for validating that the conditions under which the Custom Data were stored are still matching the conditions on the Material Object.

To do so, the Custom Data Source can store the relevant conditions on the Material Object within the Custom Data Container and check that these are matching the current conditions.

REQ-007-PME: if the Material Object is requested to provide an instance of a Custom Data Container, this instance must be available.

Rationale: only the Material Object has a function to obtain the Custom Data Container. The Custom Data Source cannot create and set the Custom Data Container at will. The Material Object can either create the Custom Data Container as soon as it is associated with the Custom Data Source, or the Material Object can do so just-in-time when the Custom Data Source requests the instance. To create the instance, the Material Object depends on the Custom Data Source as indicated in **REQ-008-CDS**.

REQ-008-CDS: Custom Data Source provides a method to create a new instance of a Custom Data Container.

Rationale: Custom Data Source is the object that implements the Custom Data Container, and, therefore, is the only object that can create an instance of the Custom Data Container. The Custom Data Source is requested to do so by the Material Object prior to or during the first request to provide the Custom Data Container instance, after creation of the Material Object or after the Custom Data Container was discarded (see **REQ-004-PME**).

REQ-009-PME: the PME stores the Custom Data Container as a reference to a *CapeInterface*.

Rationale: Custom Data Source is free to choose the format of its Custom Data. The PME is only required to store the Custom Data Container and manage its lifetime but does not need to be concerned with the content of the Custom Data Container.

REQ-010-CDS: Custom Data Source must be able to operate even if the Custom Data Container is not present.

Rationale: it is assumed that Custom Data represent intermediate calculation results and that the Custom Data Source can always reconstruct Custom Data.

Furthermore, as the PME can typically not ensure that the thermodynamic configuration has remained unchanged between sessions, the PME must not persist Custom Data between sessions, to comply with **REQ-004-PME**.

REQ-011-CDC: if transfer between threads or processes needs to be facilitated, Persistence Common Interface must be implemented on Custom Data Containers.

Rationale: if PME wants to implement multithreading, PME may need to transfer content of Material Objects between threads or processes. The PME owned data associated with the Material Object (such as temperature, pressure, compositions, flow rates, phase fractions, etc..), can be transferred by mechanisms internal to the PME. Care must be taken when transferring content of Custom Data stored into the Material Object, as the threading model defined by the middleware may not allow to access the Custom Data from another thread. To support transferring Custom Data between threads or processes, the Custom Data Container must implement the Persistence Common Interface (ii). To transfer Custom Data from one thread to another, the Custom Data is persisted in one thread and the new Custom Data Container is created in the other thread. The Custom Data is de-persisted into the new Custom Data Container.

If a Custom Data Container doesn't implement the Persistence Common Interface, when transferring content from a source Material Object to a target Material Object, the target Material Object will return an empty Custom Data Container when asked. The Custom Data Source should then be able to recreate the Custom Data on the next calculation request, as per requirement **REQ-010-CDS**.

Note that requirement **REQ-004-PME** prohibits the PME from using the persistence interface for storing the Custom Data Container content between sessions. This means that the persistence implementation on the Custom Data Container could be much more lightweight, as it doesn't need to consider versioning, backwards and/or forwards compatibility.

REQ-012-PME: the lifespan of the Custom Data Container is coupled to the lifespan of the Material Object to which it is bound.

Rationale: the purpose for the Custom Data interface is to allow for storing Material Object dependent data.

This requirement is fulfilled by middleware functionality: Material Object keeps a reference to the Custom Data Container. When the Material Object stops to exist (or upon discarding a Custom Data Container due to **REQ-004-PME**), the Material Object releases its reference to the Custom Data Container.

REQ-013-PME: copying a Material Object through *ICapeThermoMaterial::CopyFromMaterial* includes copying associated Custom Data Container instances.

Rationale: when copying a Material Object, the PME must call *ICapeCustomDataSource::CopyCustomData* implemented by the Custom Data Source.

REQ-014-PME: to support Custom Data, the Material Object must implement *ICapeThermoMaterialCustomData*.

Rationale: Material Object must provide functionality to access custom data.

REQ-015-CDS: Custom Data Source must implement *ICapeCustomDataSource*.

Rationale: Custom Data Source provides this interface to allow creation and copying of Custom Data Container instances.

2.2 Use Cases

This section lists the Use Cases pertaining to storing and retrieving custom data on Material Objects.

2.2.1 Actors

The following packages can be found in the "Actors" section of the Use Cases.

- **PME**. A Process Modelling Environment, also known as a CAPE-OPEN Simulator Executive (COSE).
- **Custom Data Source**. A Process Modelling Component which deals with an unknown number of Material Objects (such as Property Packages and Chemical Reaction Packages, implementing *ICapeThermoMaterialContext* or taking Material Object as argument to methods) and which stores custom data on Material Objects.

2.2.2 Use Case Priorities

High. Essential functionality for using Custom Data. Functionality without which the usability or performance of Custom Data might be seriously compromised.

Low. Desirable functionality that will improve the performance of Custom Data. If this Use Case is not met usability or acceptance can decrease.

2.2.3 List of Use Cases

UC-CD-001: Create Custom Data Container

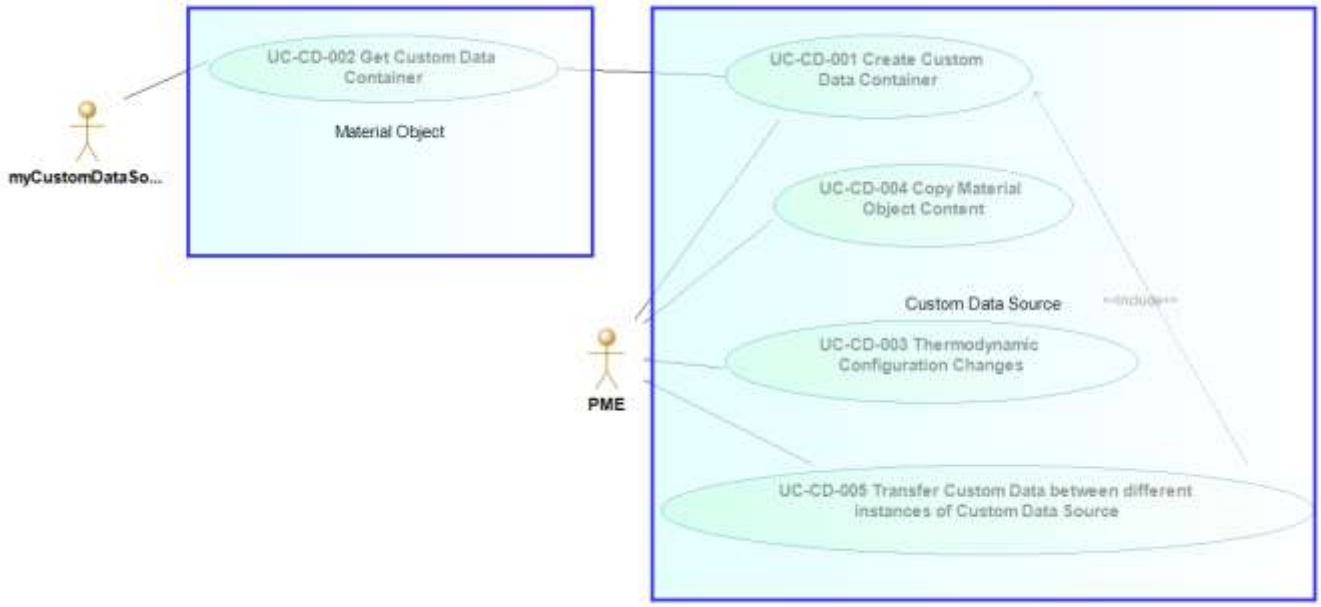
UC-CD-002: Get Custom Data Container

UC-CD-003: Thermodynamic configuration changes

UC-CD-004: Copy Material Object content

UC-CD-005: Transfer Custom Data between different instances of Custom Data Source

2.2.4 Use Cases map



2.2.5 Use Cases

Errors in exercising the Use Cases should not lead to exceptions. The errors should never be critical for the operations of any Custom Data Source or PME. The reason is that the aim of custom data is to improve performance. The unavailability of custom data should not prevent any Custom Data Source to perform what it is intended to do. Custom data are intermediate results that can always be re-calculated.

UC-CD-001: CREATE CUSTOM DATA

Actor: **PME**

Priority: <**High**>

Status:

<This Use Case is fulfilled by *ICapeCustomDataSource::CreateCustomDataContainer*.>

Pre-conditions:

<A Material Object is associated with a Custom Data Source.>

Flow of events:

The PME calls *ICapeCustomDataSource::CreateCustomDataContainer* on the Custom Data Source to create a new instance of a Custom Data Container. A reference to the Custom Data Container is stored on the Material Object with which the Custom Data Container is associated.

Post-conditions:

<A new Custom Data Container instance exists and a reference to it is stored on the Material Object.>

Errors:

None

Uses:

None

UC-CD-002: GET CUSTOM DATA CONTAINER

Actor: **Custom Data Source**

Priority: <**High**>

Status:

<This Use Case is fulfilled by *ICapeThermoMaterialCustomData::GetCustomDataContainer*.>

Pre-conditions:

<The PME supports custom data storage and retrieval.>

Flow of events:

The Custom Data Source, identifying itself to the Material Object, calls *ICapeThermoMaterialCustomData::GetCustomDataContainer* on the Material Object. If the Custom Data Container has not yet been created, the Material Object creates the Custom Data Container by using <**UC-CD-001**>. The Material Object returns the instance of the Custom Data Container that belongs to the Custom Data Source.

Post-conditions:

<The Custom Data Source has retrieved the reference to the Custom Data Container.>

Errors:

None

Uses:

<UC-CD-001>

UC-CD-003: THERMODYNAMIC CONFIGURATION CHANGES

Actor: **PME**

Priority: <**High**>

Status:

<This Use Case is fulfilled by *ICapeCustomDataSource::ThermodynamicConfigurationChanged*.>

Pre-conditions:

<The PME supports custom data storage and retrieval.>

Flow of events:

The PME detects that any aspect of a thermodynamic configuration has changed, for example, the list of Compounds has changed, a thermodynamic property calculation routine has changed, an external Property Package has been edited, etc...

The PME iterates over all Material Objects that are affected by the detected change in thermodynamic configuration: for each Material Object, the PME invokes the method *ICapeCustomDataSource::ThermodynamicConfigurationChanged* on the Custom Data Source, specifying the Custom Data Container.

The Custom Data Source clears all thermodynamic dependent data on the Custom Data Container.

Post-conditions:

<No custom data remains that is inconsistent with the current thermodynamic configuration.>

Errors:

None

Uses:

None

UC-CD-004: COPY MATERIAL OBJECT CONTENT

Actor: **PME**

Priority: <**High**>

Status:

<This Use Case is fulfilled by mechanisms internal to the PME.>

Pre-conditions:

None

Flow of events:

Unit Operations or other PMCs may request the PME to duplicate a Material Object or to copy the content of one Material Object to another using *ICapeThermoMaterial::CopyFromMaterial*. When copying Material Object content, the PME also copies any associated Custom Data Container(s), for improved performance when the Custom Data Source(s) operate(s) on the copied Material Object.

To do so, for each Custom Data Container, the PME calls *ICapeCustomDataSource::CopyCustomData* on the Custom Data Source, passing both the source and target Custom Data Containers as arguments.

Note: Custom Data Container implementations can apply optimization techniques like copy-on-write, where data is not copied, but is instead shared between the original and the copied Custom Data Containers, until the data on either of them is changed. Only then a real (deep) copy will be made. The author of the Custom Data Container is responsible for implementing such optimizations in a way that maintains the same thread-safety guarantees as two fully independent copies.

Post-conditions:

<A copy of the Custom Data Container is available on a copied Material Object.>

Errors:

None

Uses:

None

UC-CD-005: TRANSFER CUSTOM DATA BETWEEN DIFFERENT INSTANCES OF CUSTOM DATA SOURCE

This Use Case applies when

- the Material Object is copied from one application to another;
- the Material Object is copied from one thread to another and the Custom Data Source associated with the target thread is different from the Custom Data Source in the source thread.
- any other scenario in which it cannot be guaranteed that the Custom Data Source associated with the target and source Material Objects are the same.

Actor: **PME**

Priority: <**Low**>

Status:

<This Use Case is fulfilled by the persistence common interface.>

Pre-conditions:

<Source and target Material Objects have the same thermodynamic configuration.>

Flow of events:

For the source Material Object, the PME persists the content of the Custom Data Container(s) by using the persistence interface if implemented on the Custom Data Container(s) (see Persistence Common Interface Specification (ii).

The PME transfers persisted content to the target Material Object using mechanisms internal to the PME.

For each Custom Data Container (of a given Custom Data Source) for which content was persisted:

- If a Custom Data Container for the same Custom Data Source is present on the target Material Object, it is released.

- A new Custom Data Container is associated with the target Material Object using <**UC-CD-001**>.

- The persisted content of the Custom Data Container is de-persisted using the persistence interface implemented on the Custom Data Container.

Post-conditions:

<The Custom Data has been transferred for Custom Data Container(s) that support(s) persistence.>

Errors:

None.

Uses:

<**UC-CD-001**>

2.3 Sequence diagram

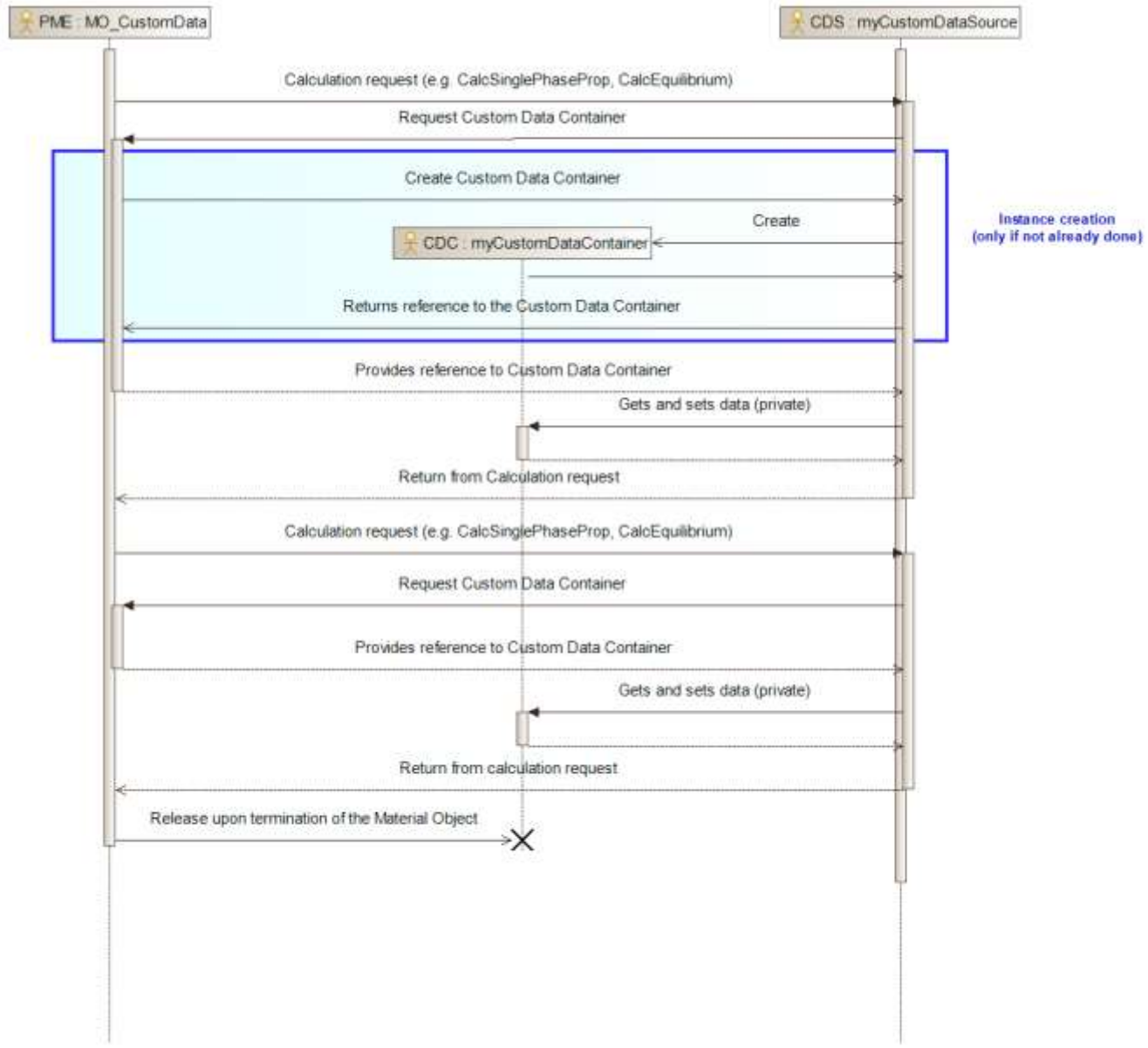


Figure 1 Typical use of a Custom Data Container

3. Analysis and Design

This section describes in detail the two interfaces pertaining to Custom Data functionality. *ICapeThermoMaterialCustomData* is an interface implemented on the Material Object in addition to the interfaces described in the Thermodynamic and Physical Properties Interface (i), the Identification Common Interface, the Error Common Interfaces, etc… specifications. *ICapeCustomDataSource* is an interface implemented on any Custom Data Source, hence qualifying a Process Modelling Component as a Custom Data Source.

3.1 Interface diagrams

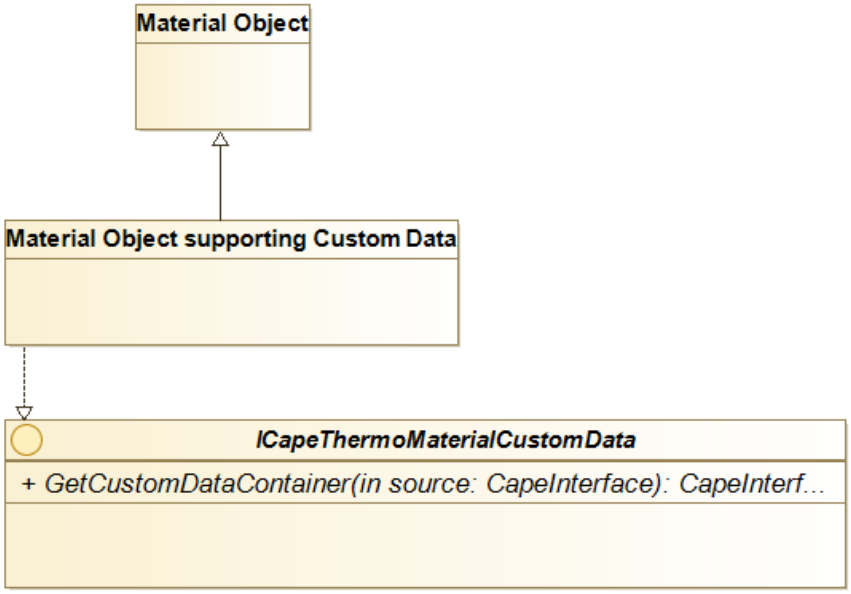


Figure 2 Static diagram of Material Object supporting Custom Data

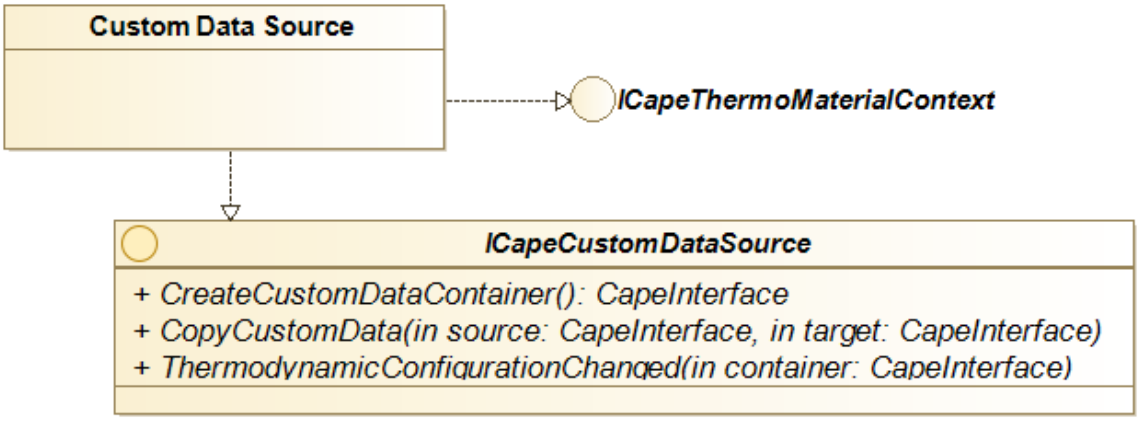


Figure 3 Static diagram of Custom Data Source

3.2 State diagrams

This document does not call for any state diagram.

3.3 Other diagrams

None.

3.4 Interface description

The *ICapeThermoMaterialCustomData* interface is implemented by a Material Object supporting the Custom Data functionality. The *ICapeCustomDataSource* interface is implemented by a PMC acting as a Custom Data Source.

3.4.1 ICapeThermoMaterialCustomData

Interface Name	ICapeThermoMaterialCustomData
Method Name	GetCustomDataContainer
Returns	CapeInterface

Description

Provides Custom Data Container associated with the specified Custom Data Source.

Attributes

Name	Type	Description
[in] source	CapeInterface	The Custom Data Source that is requesting the data.

Errors

There is no specific error for this method.

Notes

The Material Object must return the Custom Data Container corresponding to the Custom Data Source that created the Custom Data Container. The PME may have created the Custom Data Container earlier in the life span of the Material Object (e.g. at the time it was associated with the Custom Data Source), or may do so during the *GetCustomDataContainer* method, by using *ICapeCustomDataSource::CreateCustomDataContainer* on the Custom Data Source.

The PME does not persist the Custom Data Container between sessions. To perform its operations, the Custom Data Source should not depend on the custom data that was previously stored on the Material Object being available; the Custom Data Source should function also if the stored custom data is no longer available. The Custom Data Source chooses the format of the custom data. The PME returns the custom data as it was stored by the Custom Data Source.

3.4.2 ICapeCustomDataSource

Interface Name	ICapeCustomDataSource
Method Name	CreateCustomDataContainer
Returns	CapeInterface

Description

Creates a new instance of the Custom Data Container.

Attributes

None

Errors

There is no specific error for this method.

Notes

The life span of the Custom Data Container is controlled by the Material Object that stores the reference to the Custom Data Container.

Interface Name	ICapeCustomDataSource
Method Name	CopyCustomData
Returns	-

Description

Copies the content of one Custom Data Container to another Custom Data Container.

Attributes

Name	Type	Description
[in] source	CapeInterface	The Custom Data Container from which the content is to be copied.
[in] target	CapeInterface	The Custom Data Container to which the content is to be copied.

Errors

There is no specific error for this method.

Notes

Copying Custom Data Container content is part of copying Material Object content. The Custom Data Source is free to make either a deep copy of the data, or make a shallow copy of the data at this point, and postpone making the deep copy (if required) until the data that is shared between multiple instances of the Custom Data Container (e.g. on the source and target Material Objects) is to be modified (copy-on-write).

The Custom Data Source cannot assume that the Custom Data Container passed as argument belongs to a Material Object of the same type as the current Material Object.

Interface Name	ICapeCustomDataSource
Method Name	ThermodynamicConfigurationChanged
Returns	-

Description

Clears content of Custom Data Container on change of thermodynamic configuration of the Material Object.

Attributes

Name	Type	Description
[in] container	CapeInterface	The Custom Data Container from which the content is to be cleared.

Errors

There is no specific error for this method.

Notes

This method is called by the PME when the thermodynamic configuration associated with the Material Object has changed. This presumably makes that the data contained in the Custom Data Container is inconsistent with the thermodynamic context.

It is not necessary to call this method prior to the final release of the Custom Data Container.

The Custom Data Source cannot assume that the Custom Data Container passed as argument belongs to a Material Object of the same type as the current Material Object.

4. Scenarios

4.1 Example: A Simple Electrolyte Model

In the introduction, an example was given of an electrolyte model, which needs a true composition to calculate the aqueous phase properties. Here we explore in a little more detail how the Custom Data interfaces could be used to improve performance for such an electrolyte model. For this example, we will assume the following:

- The aqueous phase is modelled using activity coefficients, which are functions of temperature and composition only.
- The component list only contains species with neutral charge, which we refer to as the apparent composition.
- Any charged ions are only used internally to the model, which will account for dissociation reactions and the effects of charge. The full list of internal components is referred to as the true composition.
- Properties are reported only in terms of the apparent composition of neutral species but depend on the full true composition.
- Reaction constants are functions of temperature only.
- The PME will often request a calculation of a phase equilibrium, and then requests calculating many phase properties in succession at the same conditions.

The above example is simple enough to cover here but contains enough complexity to extend it to other cases. We hereby explore how it is possible to optimize calculations while using Custom Data.

The author of the Property Package must first implement a Custom Data Container. This is an object that implements both the Custom Data Container interface, as well as some private interface. This private interface gives access to the private data. In this example, the Custom Data Container had a field for the true composition, as well as fields for the temperature and apparent composition at which the true composition was calculated, and a flag to indicate the empty state.

The author of the Property Package must implement the *ICapeCustomDataSource* interface on the Property Package. The primary method of this interface is *CreateCustomDataContainer*, which should create a new Custom Data Container in an empty state.

Finally, the calculation procedure must be changed. When a calculation request comes in, the Property Package performs the following steps:

1. Request the Custom Data Container from the current Material Object, using *ICapeThermoCustomData::GetCustomDataContainer*. The Material Object will create a Custom Data Container if it doesn't exist.
2. Obtain the private interface from the Custom Data Container (for example using *QueryInterface* or some other platform- or language-dependent mechanism).
3. Check if the Custom Data Container is in an empty state. If so, skip to step 5.
4. Check if the input temperature and input composition to the calculation request match the stored temperature and apparent composition. If so, skip to step 6.
5. Calculate the true composition at the current temperature and apparent composition. Store the temperature, apparent and true composition on the Custom Data Container.
6. Now use the true composition stored on the Custom Data Container as input for property calculations.

4.2 Extension to any intermediate thermodynamic results

The above example can be easily extended. For example, when equations of state are used, the true composition will be dependent on pressure as well, so the Custom Data Container will also have to store pressure. In addition, when using a full, multi-phase chemical equilibrium calculation, it might be beneficial to store the true composition of all phases. In fact, any intermediate calculation results, such as compressibility factors solutions of the equation of state, could be stored as well.

4.3 Possible application to Petroleum Properties

Pending the final design of the Petroleum Fractions interface specification, another scenario would be storage of characterization data (e.g. properties of pseudo-compounds such as critical pressures and temperatures) by a Property Package which deals with Petroleum Fractions. It is still unclear if storing this type of Custom Data would be beneficial from a performance point of view. However, sharing characterization data in between streams may be beneficial and could rely on custom data storage using a reference to an object holding the custom data.

5. Prototypes implementation

None so far.

6. Bibliography

(i) CAPE-OPEN Interface Specifications: Thermodynamic and Physical Properties Version 1.1

(ii) CAPE-OPEN Interface Specifications: Persistence Common Interfaces

(iii) Open Interface Specification: Identification Common Interface