CAPE-OPEN

Delivering the power of component software
and open standard interfaces
in Computer-Aided Process Engineering

Open Interface Specification:

Chemical Reactions Interface

www.colan.org

ARCHIVAL INFORMATION

Filename	Chemical Reactions Interface Specification.doc
Authors	CO-LaN consortium
Status	Public
Date	August 2003
Version	version 2
Number of pages	68
Versioning	version 2, reviewed by Jean-Pierre Belaud, August 2003
	version 1, March 2002
Additional material	
Web location	www.colan.org
Implementation specifications version	CAPE-OPENv1-0-0.idl (CORBA) CAPE-OPENv1-0-0.zip and CAPE-OPENv1-0-0.tlb (COM)
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.

SUMMARY

This document contains the specification of the ***CAPE-OPEN Reactions interfaces***. It introduces two new primary CAPE-OPEN components: *Reactions Package* and *Reactions Package Manager*, and introduces the concept of a Reaction object as a mechanism for passing reaction data between a CAPE-OPEN compliant Process Modelling Environment (PME) and a Reactions Package component.

ACKNOWLEDGEMENTS

CONTENTS

LIST OF FIGURES

1. Introduction

This document describes two new CAPE-OPEN components: Reactions Package and Reactions Package Manager. These components are described in terms of: the interfaces that they must support their interaction with a PME and the functionality they are expected to support. The interfaces defined here support both kinetic and electrolyte reactions.

A Reactions Package Manager component is a factory for the set of Reactions Packages that it manages and it may act as tool for constructing new Reactions Packages.

A Reactions Package contains the definition of a set of kinetic reactions for a defined set of compounds and phases. It allows a client to request the calculation of Reaction Properties, for example the reaction rate. A Reactions Package may allow a client to reconfigure the reactions that it defines, for example to change activation energies.

Equilibrium Reactions in electrolyte mixtures are supported by combing the concepts of the Reactions Package and the CAPE-OPEN thermodynamic Property Package. The resulting Electrolyte Property Package supports both the Property Package and the Reactions Package interfaces. The combined set of interfaces allows a client to request the calculation of equilibrium reaction properties and thermodynamic property calculations from a single package. This is the mechanism used to support Electrolyte mixtures within the CAPE-OPEN standard.

1.1 Background Information

CAPE-OPEN covers the definition of standard interfaces for Thermodynamic packages for standard mixtures [1]; UNIT Operations [2] and Numerical Solvers [3]. The CAPE-OPEN Thermo Interfaces, however, do not cover handling of complex and/or ill-defined mixtures, that is, the CO-Interfaces only deal with well-defined molecular species. Specifically, the requirements for handling petroleum assays, petroleum fractions, electrolyte, polymers and particulate materials were not considered in the interface design and definition. Also, interfaces for describing reactive systems were not handled.

There are many similarities between the CAPE-OPEN thermo specifications [1] and the Reaction specifications described here. Both specifications define a manager component, which is used to construct Packages. Both define a package component, which performs calculations and defines a mixture in the case of a Property Package and a set of reactions in the case of a Reactions Package. Both specifications define an object, which is used to pass data between a component and its client.

This document is consistent with version 1.0 of the CAPE-OPEN Thermo Interfaces [1]. The key CAPE-OPEN Thermo Interfaces are:

- ICAPEThermoSystem for Thermo system
- ICAPEThermoMaterialTemplate for the Material template
- ICAPEThermoMaterialObject for the Material object
- ICAPEThermoCalculationRoutine for the Calculation routine
- ICAPEThermoPropertyPackage for the Property Package
- ICAPEThermoEquilibriumServer for the Equilibrium server

1.2 Audience

This document is intended primarily for software developers who want to build CAPE-OPEN Reaction system components. It is also intended for developers of CAPE-OPEN-compliant simulation environments because it describes how the interfaces are to be used to implement communication between the environment and the external software components.

Designers of other CAPE-OPEN interface specifications should read this document to ensure consistency across the various designs.

For any reader an understanding of UML diagrams is assumed. Also, the reader should consult the CAPE-OPEN Thermodynamic and Physical Properties Open Interface Specification document for a description of terms not included in this document.

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

1.3 Glossary

1.3.1 Aqueous Infinite Dilution Heat Capacity

Heat capacity of a solute in an infinitely dilute aqueous solution.

1.3.2 Association/dissociation

When monomeric molecules are capable of forming an aggregate, the result is association and vice versa for dissociation. Association (dissociation) between molecules can take place in gaseous and liquid states and in solutions.

Association of an electrolyte solution refers to the association of ions into non-conducting species such as ion-pairs. Dissociation of an electrolyte solution refers to the dissociation of the electrolyte (solute) to the ionized (dissociated) state in solution.

1.3.3 Born Radius:

The Born model, which prescribes the electrostatic part of the solvation free energy of an ion in a dielectric continuum, has been widely and successfully used to treat ionic solvation phenomena. Despite the fact that the solvent molecules around the ions are highly structured giving rise to molecular phenomena such as dielectric saturation and electrostriction, the Born model yields accurate results compared to experimental data if a proper radius for the ion, the effective Born radius *Reff*, is used in the Born free energy expression.

1.3.4 Chemical Equilibrium

A state in which a chemical reaction and its reverse reaction are taking place at equal velocities, so that the concentrations of reacting substances remain constant.

1.3.5 Chemical Reactions

Refers to the set of reactions for which a state of chemical equilibrium has been attained.

1.3.6 Compound

A chemical substance as defined by a particular set of Physical Properties, calculation methods and data. Compounds can be identified in various ways: by a common name, by a CAS Registry number or by a chemical formula. Examples are water, hydrogen and oxygen.

1.3.7 Component

A piece of executable software whose functionality is accessed via specified interfaces, which can be deployed independently of other software and which can be used in the composition of other software systems without modification.

1.3.8 Dielectric Constant

For a given substance (species), the ratio of the capacity of a condenser with that substance as dielectric to the capacity of the same condenser with a vacuum for dielectric. It is a measure, therefore, of the amount of electrical charge a given substance withstand at a given electric field strength.

1.3.9 Dissociation Constant

The chemical equilibrium constant corresponding to a dissociation reaction.

1.3.10 Electrolyte solution

An electrolyte solution is a mixture of dissociated ions in a mixture of solvents.

1.3.11 Equilibrium Constant

The product of the concentration (or activities) of the substances produced at equilibrium in a chemical reaction divided by the product of concentrations of the reacting substances, each concentration raised to that power which is the coefficient of the substance in the chemical equation.

1.3.12 Equilibrium Constant

The product of the concentration (or activities) of the substances produced at equilibrium in a chemical reaction divided by the product of concentrations of the reacting substances, each concentration raised to that power which is the coefficient of the substance in the chemical equation.

1.3.13 Ionic Species/Radicals

An ion is an atom or group of atoms that is not electrically neutral but instead carries a positive or negative electric charge. Positive ions (anions) are formed when neutral atoms or molecules lose valence electrons; negative ions (cations) are those that have gained electrons.

1.3.14 Ionic Species:

An ion is an atom or group of atoms that is not electrically neutral but instead carries a positive or negative electric charge. Positive ions (anions) are formed when neutral atoms or molecules lose valence electrons; negative ions (cations) are those that have gained electrons.

1.3.15 Kinetic Reactions

A mixture of chemical species, when brought in contact with a catalyst, will react to a greater or lesser extent to form new chemical species at the expense of all or some of the original species present in the system. If a state of chemical equilibrium is not attained, the corresponding set of reactions is referred to as kinetic reactions

1.3.16 Mean (molal) Activity Coefficient

The mean molal activity coefficient of an electrolyte solution is the geometrical mean of the activity coefficients of the ions in the solution.

1.3.17 Osmotic (constant) Coefficient

In dilute aqueous solutions, the water activity and the water activity coefficients are very close to unity. In order to be able to report water activities without a large number of significant digits, the osmotic 9constant) coefficient is commonly used. The practical or molal osmotic coefficient is defined as,

$$\phi = - n_W \ln (x_W f_W)/(n_S \sum \nu_i)$$

where,

n_W is the moles of water; n_S is the moles of solute; x_W is the mole fraction of water; f_W is the symmetric activity coefficient of water; ν_I is the stoichiometric coefficient of component i.

1.3.18 pH & pOH

The term pH derives from a combination of p for the word power and H for the symbol of the element Hydrogen. pH is the negative log of the activity of hydrogen ions.

$$pH = -\log_{10} a_{H^+}$$

pH represents the *'activity'* of hydrogen ions in a *solution*, at a given *temperature*. The term *activity* is used because pH reflects the amount of available hydrogen ions not the *concentration* of hydrogen ions. Replace H with OH to get the activity of OH ions.

1.3.19 Phase

A phase is a stable or metastable collection of compounds with a defined amount of substance and a homogeneous composition. It has an associated state of aggregation, *e.g.* liquid. A given phase can be distinguished from others through the presence of physical interfaces that separate states of matter with different characteristics, such as density.

1.3.20 Physical Equilibrium

Dynamic state in which two opposing physical changes occur at equal rates in the same system. The stable dynamic state is obtained from the Gibbs energy of the system. A system with N phases is stable when the corresponding Gibbs energy of the system is at a minimum. If N =1, then only phase phase exists at a given condition. If N=2, then two coexisting phases are at physical equilibrium at the given condition of the system and so on.

1.3.21 Reaction Rate

A reaction rate is the speed at which reactants are converted into products in a chemical reaction. The reaction rate is given as the instantaneous rate of change for any reactant or product, and is usually written as a derivative (e. g. d[A]/dt) with units of concentration per unit time (e. g. mol L^{-1} s^{-1}).

1.3.22 Solubility Index

The solubility index or degree of saturation of a salt is defined as the activity product of a salt divided by its solubility product (see equation).

1.4 Scope

This document describes two new primary [4] CAPE-OPEN components:

- Reactions Package Manager – Similar in scope to a Thermo System component. A Reactions Package Manager component manages a set of Reactions Packages. It can instantiate a Reactions Package and it may optionally allow new Reactions Packages to be created.

- Reactions Package – A Reactions Package component defines a set of reactions involving a specific set of compounds and phases. The reactions are defined primarily in terms of their stochiometry. A Reactions Package may allow some parameters of its set of reactions to be configured, it may also allow the structure of the reactions to be changed, but both of these behaviors are optional.

This document describes the following interfaces:

- ICapeReactionsPackageManager – Similar in scope to the ICapeThermoSystem. These interfaces will be implemented by a Reactions Package Manager component.

- ICapeReactionsRoutine – Similar in scope to ICapeThermoPropertyPackage. A software component or a PME that can calculate values of reaction (or reaction related) properties will implement this interface. It may also be implemented by a Physical Property package component that deals with electrolytes.

- ICapeReactionChemistry – A component or a PME that needs to describe a set of reactions will implement this interface. A set of reactions is described in terms of the compounds that take part in the reactions and the compounds that are produced. For example, in the case of electrolyte systems, salt complexes and ions. In the case of detailed reaction mechanisms, radicals.

- ICapeReactionProperties – Similar in scope to ICapeThermoMaterialObject. A component or a PME that needs to provide access to the properties of a particular reaction will implement this interface.

- ICapeKineticReactionContext – This interface allows a reaction object to be passed to a component that needs access to the properties of a set of kinetic reactions.

- ICapeElectrolyteReactionContext – This interface allows a reaction object to be passed to a component that needs access to the properties of a set of equilibrium reactions.

- ICAPEThermoContext - which allows a material object to be passed between a PME and the Reactions components it is using so that the Reactions components can make Physical Property calculation calls.

Software that provides Reactions Package functionality and the software that consumes the functionality will use these interfaces to communicate with each other. This specification will describe the following scenarios in which these interfaces will be used:

- A PME behaving as reactions package with respect to a CAPE-OPEN Unit Operation that needs to perform reaction calculations. In this case the Unit Operation is the consumer and the PME the provider.

- A standalone software component behaves as a Reactions Package with respect to a PME that needs to perform reaction calculations. In this case the PME is the consumer and the external component is the provider.

- A PME, which is using an external Reactions Package, behaving as a Reactions Package with respect to a CAPE-OPEN Unit Operation, which needs to perform reaction calculations. In this case the PME is provider with respect to the CAPE-OPEN unit and consumer with respect to the external Reactions package.

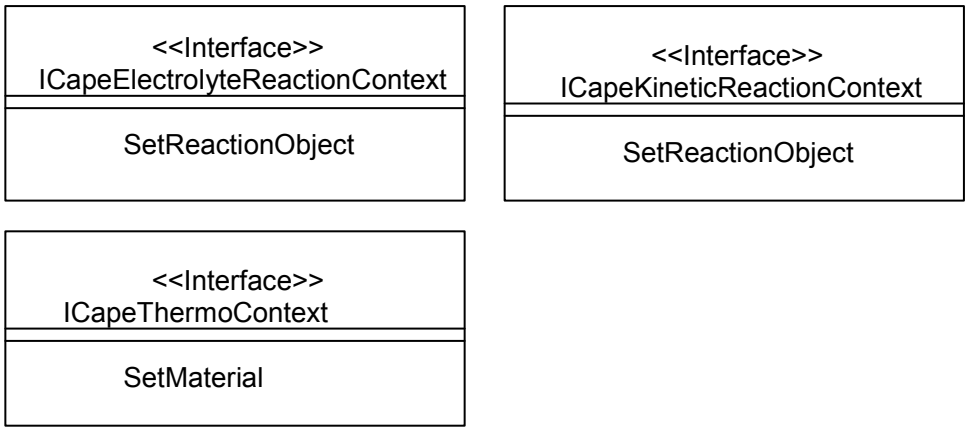
In each of these scenarios the Reactions Package functionality will be provided by a physical property package if an electrolyte reaction calculation is being performed.

In each of these scenarios the consumer may have to deal with both electrolyte and kinetic reactions.

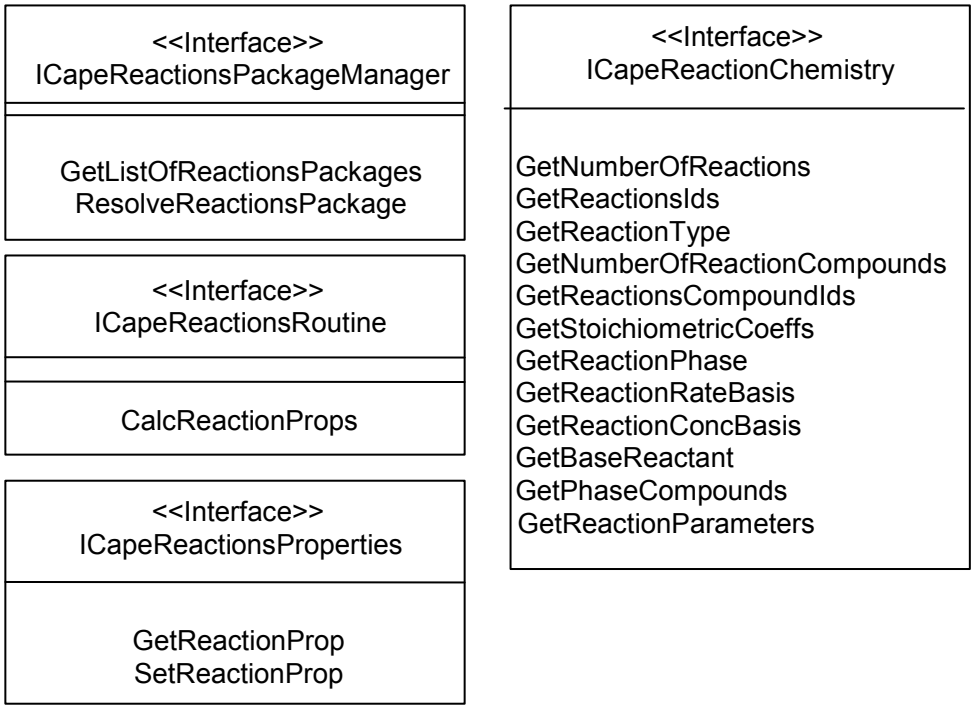
In all cases a consumer and provider of Reactions Package functionality need to be able to exchange values for reaction properties. To facilitate this exchange we define a reaction object, which, following the same pattern as the material object in the thermodynamics specification, holds the values of reaction properties and allows them to be passed between consumer and provider. The reaction object is not a CAPE-OPEN component but the interfaces that a reaction object must support are part of the standard, as is the protocol that consumer and provider must follow when exchanging reaction data.

The ICapeThermoContext interface is defined in order to allow a material object to be passed to consumers that need to perform thermodynamic property calculations. The ICapeKineticReactionContext, plays the same role for reaction property calculations. ICapeElectrolyteReactionContext is also defined.

Contexts



Reaction Interfaces



Interfaces from other packages used by this package:

ICapeUtilities
ICapeEdit
ICapeCollection
parameter interfaces

2. Requirements

2.1 Textual Requirements

Reactions Package scope

The standard must allow creating a Reactions Package component that only contains information relevant for the reaction phenomena. Thus, an equilibrium reaction or a set of equilibrium reactions grouped in a Reactions Package does not necessarily need to know how to calculate the physical properties of the mixture upon which the reaction will take place.

It follows that a Reactions Package has to be able to interact with a Property Package that will serve it with the necessary physical properties if required.

Selection of a Reactions Package

Once a Reactions Package component has been installed in a computer, it will be available to a PME as any other CAPE-OPEN component. The mechanisms by which COM components are installed and registered are described in the COM Architecture document [4]. Thus, a Reactions Package has to be selectable among a list of possible Reactions Packages presented to the user by a PME.

Upon selection of the Reactions Package, the COSE will be responsible for creating the Reactions Package and making it available for the simulation.

Two scenarios are envisioned for the creation/selection of a Reactions Package. These are:

- A Reactions Package belongs to a larger reactions system, and it has been created as a specialisation of the reaction system for a particular process. In this scenario the user will launch the reaction system, and will configure a reaction set using the native reaction system mechanism. Upon completion the reaction system will offer the user the possibility of exporting the reaction set as a CAPE-OPEN component.

- A Reactions Package has been created for a specific process; i.e. does not belong to any reaction system. In this case two sub cases can be distinguished:

    - A) the Reactions Package is pre-configured, i.e. it handles a specific set of reactions with a specific set of products and reactants. Thus, when this Reaction Routine is selected, it is already ready to be used.

    - B) The Reactions Package allows re-configuration, i.e. it can have a default configuration or not, but in either case, the Reactions Package allows the user to specify components participating in the reaction, reactions taking place in the reaction set, and the various parameters involved in the reaction rate expressions. This case is an intermediate between 1) and 2A) in the sense that the Reaction Routine behaves as a small reactions system.

In the case 1), and once the Reactions Package has been exported as a CAPE-OPEN component, it will behave identically to the component created in case 2).

It is expected that when a COSE loads a Reactions Package, or an Electrolyte Property Package, an end-user will be able to view the contents of the package using either a User Interface supported by the package or via a generic interface constructed by the COSE. A COSE is not expected to use the Package to fill in its own data structures, as if the user had entered the data directly. Filling in COSE data structures would require

standardisation of the forms of the reaction models supported by Reactions Packages. This level of standardisation is deliberately not included here because the standard needs to be flexible enough to cover any formulation of the well-known reaction models as well as custom reaction models.

2.2 Use Cases

Packages

The following packages or sub-systems were identified:

- PME. A Process Modeling Environment for example a CAPE-OPEN Simulator Executive (COSE).
- UNIT. A CAPE-OPEN Unit Operation.
- THRM. A CAPE-OPEN Thermodynamic Property Package
- Reactions Package Manager. A software component that allows an actor to select among a list of possible Reactions Package, and optionally to create new reactions.
- Reactions Package. A software component containing all the information and the calculation capabilities to allow a client to solve the material and energy balance for a reactive mixture.

The term "Reaction set" is used here to denote the group of reactions contained in a given Reactions Package.

2.2.1 Actors

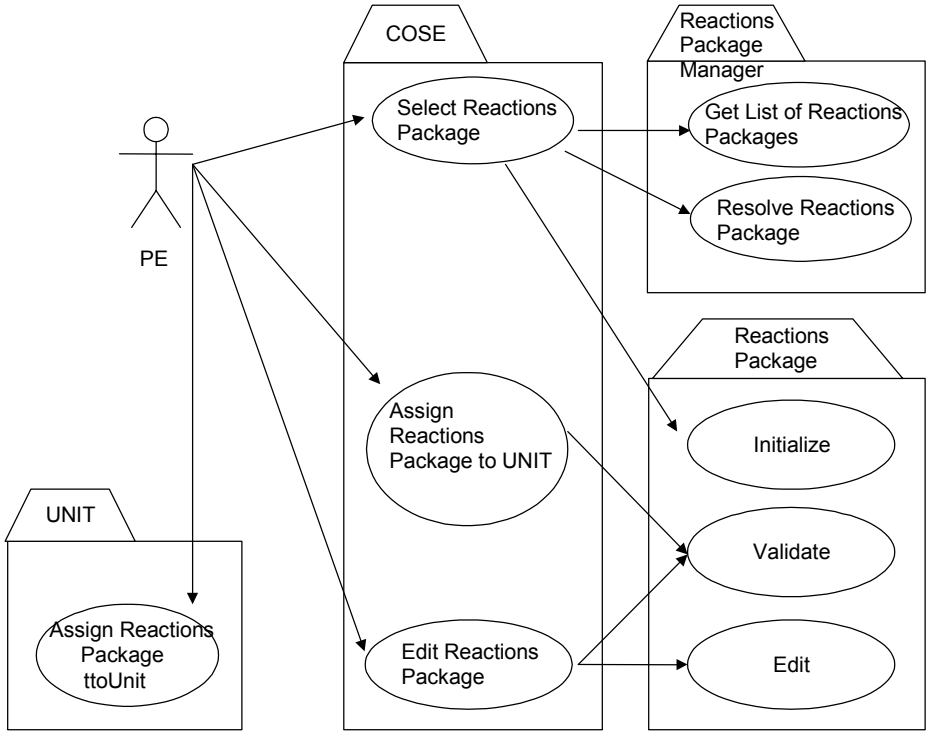
The only human actor identified was the Process Engineer (PE). For the purposes of this document, it was made the assumption that almost any sub-system could be considered an actor. Thus, packages can be found in the "Actors" section of the Use Cases. The reason for this is that when a sub-system instructs another to perform a particular action, the former could be considered as an actor from the viewpoint of the last.

2.2.2 List of Use Cases

2.2.3 Use Cases maps

CONFIGURATION OF THE REACTIONS PACKAGE

The diagram shown in this section is the Use Cases map for the requirements necessary to: Select a Reactions Package, Configure it (if necessary) and Assign it to the simulation. After successful completion of these Use Cases the system is in a valid state and ready to calculate.

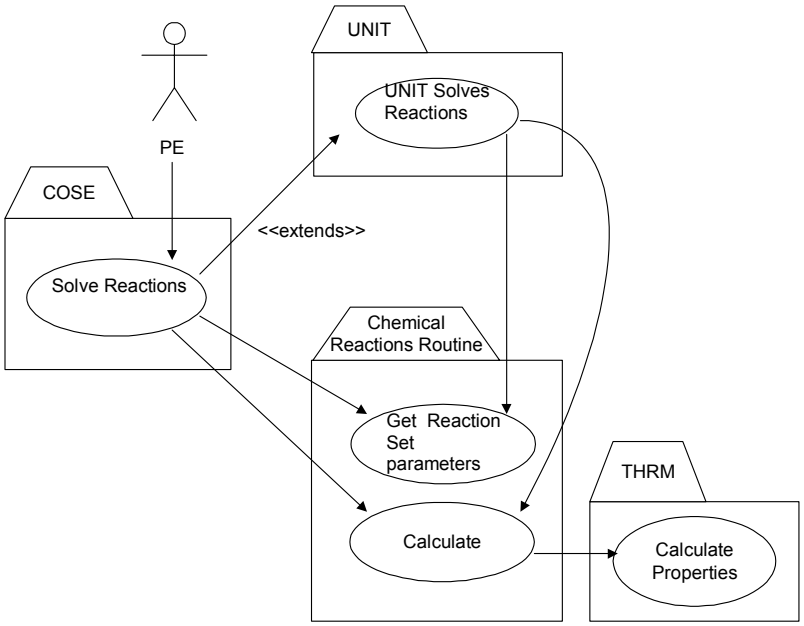


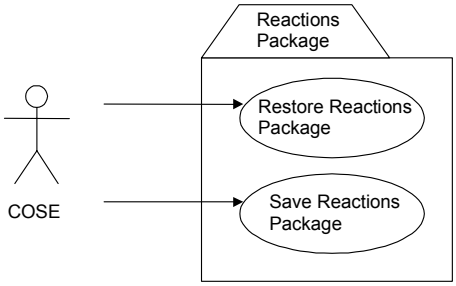
In the case of a combined Reactions Package and Property Package, configuration takes place as indicated in the THRM document

RUNNING A REACTIONS PACKAGE

The diagram shown in this section is the Use Cases map for the requirements necessary to: Execute a Reactions Package. Use Cases for Saving and Restoring a Reactions Package have been also included.

As it is indicated in the diagram two actors can invoke execution of the Reactions Package, these are a PME directly and a CAPE-OPEN UNIT via a PME.





2.2.4 Use Cases

CONFIGURATION OF THE REACTIONS PACKAGE USE CASES

RUNTIME INFORMATION PROVIDED BY A REACTION COMPONENT

A Reactions Package has to be able to provide its clients with enough information to solve the mass and energy balances of the system in which the Reactions Package is operating. This information includes :

For the case of kinetic reactions:

(i) Compounds participating in the reaction/reactions as reactants and products

(ii) Stoichiometry of the participating reactions (reactants stoichimetric coefficients will indicated by negative number, products by positive numbers)

(iii) Base compound of the reaction

(iv) Reaction units (e.g. kgmole/h-kg cat)

(v) Phase to which the reaction rate expressions will be applied

(vi) Reaction rates for the various participating reactions at a given set of conditions (e.g. T, P and z)

(vii) Heat of reaction rates for the various participating reactions at a given set of conditions (e.g. T, P and z)

(viii) Derivatives of the reaction rates with respect to temperature and composition (e.g. expressed as mole fractions, fugacities, partial pressures, etc).

For equilibrium reactions:

(i) Compounds participating in the reaction/reactions as reactants and products

(ii) Stoichiometry of the participating reactions (reactants stoichimetric coefficients will indicated by negative number, products by positive numbers)

(iii) Chemical equilibrium constant for the various participating reactions at a given set of conditions (e.g. T, P and z)

(iv) Heat of reaction rates for the various participating reactions at a given set of conditions (e.g. T, P and z)

CONFIGURING A REACTIONS PACKAGE

A Reactions Package has to be allowed to expose its parameters (e.g. activation energy, pre-exponential factor, etc) in a way that its clients can edit and modify their values. This functionality can be necessary for e.g. a regression package. Some Reactions Packages will not allow this, but it is important that the option to expose these parameters is supported.

UC-001: SELECT REACTIONS PACKAGE (REACS-01)

Actors: PE

Priority: High

Pre-conditions:

A minimum of Reactions Package must exist in the computer in order to trigger this Use Case

Flow of events:

The PE requests from the COSE a list of all installed Reactions Packages.

The COSE has to be able to retrieve Reactions Packages belonging to all installed Reactions Package Managers together with any standalone Reactions Packages which are not owned by a Reactions Package Manager. (From here on these will be termed as stand-alone Reactions Packages to follow the convention Property System/Property Package already proposed in the THRM Package).

The COSE presents the user with the list of available Reactions Packages.

This list is made up of the names of all stand-alone Reactions Packages and all managed Reactions Packages. To present this list the COSE will have to instantiate each Reactions Package Manager and to interrogate them for their available Reactions Packages (i.e. using the <Get List of Reactions Packages> Use Case)

The PE selects one of the Reactions Packages

The COSE "resolves" the Reactions Package by instantiating it or by asking to the corresponding Property System to instantiate it (i.e. using the <Resolve Reactions Package> Use Case)

Once the Reactions Package is available to the COSE, the COSE asks the Reactions Package to initialise itself (i.e. using the <Initialise> Use Case)

Post-conditions:

A Reactions Package is available to the COSE and has been initialised properly.

Errors:

The Reactions Package fails during instantiation

The Reactions Package fails to initialise itself

Uses:

<Resolve Reactions Package >, <Initialise>

UC-002: GET LIST OF REACTIONS PACKAGES (REACS-02)

Actors: COSE

Priority: High

Pre-conditions:

The COSE has access to a Reactions Package Manager and can interrogate it

Flow of events:

As a request from the COSE, the Reactions Package Manager creates a list of all available Reactions Packages, and supplies this list back to the COSE

Post-conditions:

There are 1 or more available Reactions Packages within the Reactions Package Manager, otherwise the returned list will be empty

UC-003: RESOLVE REACTIONS PACKAGE (REACS-03)

Actors: COSE

Priority: High

Pre-conditions:

The COSE has access to a Reactions Package Manager and can interrogate it

Flow of events:

As a request from the COSE, the Reactions Package Manager gets the name of a particular Reactions Package and checks that the selected Reactions Package is recognised. If so, the Reactions Package Manager creates and instance of the Reactions Package and delivers it to the COSE

Post-conditions:

The selected Reactions Package is recognised and it is created and delivered to the COSE correctly

Errors:

A selected Reactions Package is not recognised by the Reactions Package Manager

The selected Reactions Package can not created and delivered to the COSE (e.g. out of memory)

UC-004: INITIALISE (REACS-04)

Actors: COSE

Priority: High

Pre-conditions:

The Reactions Package has been correctly created and is not available for the COSE

Flow of events:

Right after the COSE has created and instance of the Reactions Package, as a response to a PE request, the COSE asks the Reactions Package to initialise itself.

Typically, the Reactions Package will create and initialise its internal structure, that may imply e.g. creating additional entities such as Parameter entities to be exposed outside, or any other action the Reactions Package may consider

This Use Case is triggered by the COSE only once in the life of the Reactions Package, right after its creation

Post-conditions:

The Reactions Package initialises correctly and notifies that to the COSE

Errors:

Failure during initialisation.

UC-005: VALIDATE (REACS-05)

Actors: COSE

Priority: High

Pre-conditions:

A Reactions Package is available to the COSE and has been initialised properly.

Flow of events:

It is an internal Use Case, automatically triggered by the COSE as results of completion of the Use Cases <Select Reactions Package > or <Edit Reactions Package>

The COSE needs to check that the Reactions Package will be able to perform appropriately in the context it is going to be used. Thus, the Reactions Package may need to use a Property Package to evaluate physical properties such as fugacities or activities of the mixture upon which the Reaction takes place. Therefore, the Property Package has to be able to supply this information. This Use Case is for the Reactions Package to check that its associated Property Package can supply such information.

The COSE asks the Reactions Package to Validate itself, passing a Material Object to the Reactions Package

The Reactions Package will check that the properties its needs can be calculated as requested

This Use Case will be triggered by the COSE each time the configuration of the Reactions Package changes or the Reactions Package is assigned with a new Property Package. For instance, each time the <Edit Reactions Package> Use Case is executed the required Physical Properties for the Reactions Package could be different and therefore, <Validate> has to be executed again. Similarly after the <Assign Reactions Package to Property Package> Use Case is executed <Validate> has to be passed again>

In this Use Case the Reactions Package has to match the identifiers of the compounds participating in its Reactions with the ones contained in the associated Property Package.

The Reactions Package will do so by: obtaining the component identifiers and related information such as CAS numbers, formulae, etc from the Property Package, comparing the CAS numbers of the Reactions Packages with the ones provided by the Property Package, and obtaining the corresponding values of the component identifiers. In case of hypothetical components in the Reactions Package, this should open a dialog and prompt the user to make the association between the component defined in the Reactions Package and the ones provided by the Property Package.

The Reactions Package will check as well that the phases it expects are supported by the Property Package.

Note that until <Validate> Use Case has been passed successfully the information that the client will get by executing the Use Case <Get Reaction Parameters> will be incorrect, in particular regarding the component identifiers (i.e. the obtained component identifiers will not match the ones in the rest of the flowsheet).

Post-conditions:

The Reactions Package recognises the Property Package as suitable for its needs and Validation passes successfully.

Errors:

The Property Package associated with the Reactions Package can not provide the required physical properties, therefore <Validate> fails

The Reactions Package is selected before the PE has selected a Property Package, in which case the COSE has to postpone validation until it can construct a material object.

UC-006: EDIT REACTIONS PACKAGE (REACS-06)

Actors: PE

Priority: Medium

Pre-conditions:

The Reactions Package can be edited, e.g. it has its own graphical interface that can be displayed to visualise and/or modify the configuration of the Reactions Package, or alternatively, the COSE can construct a generic graphical interface to serve for the same purpose. Nevertheless, it is recommended that each Reactions Package requiring configuration by the PE is provided with its own interface, since the default interface provided by the COSE will necessarily be less intuitive and user friendly.

Flow of events:

This Use Case may not be strictly necessary for all Reactions Packages. Thus, for those Reactions Packages not requiring any configuration this Use Case is not relevant (as it will be shown here below).

The PE through the COSE request from the Reactions Package to display its graphical means, by using the <Edit> Use Case

After the editing of the Reactions Package is completed, the COSE will request that the Reactions Package validate itself against the assigned Property Package, using the <Validate> Use Case

Alternate Path:

The PE through the COSE requests that the Reactions Package displays its graphical User Interface but this functionality is not supported by the Reactions Package

The COSE attempts to construct a default graphical interface for the Reactions Package, for doing so:

The COSE asks the Reactions Package to provide the list of its Public Parameters

If The Reactions Package is prepared to do so, it supplies to the COSE the list of its Public Parameters. If not, the COSE informs the PE that the Reactions Package can neither be edited nor its configuration changed, and this Use Case terminates

At this point the PE can change the configuration of the Reactions Package by activating or deactivating some Reactions, modifying the Reaction expressions or the Reaction parameters. There is no restriction on how many or which of these items have to be available for the PE to interact with. Each Reactions Package implementor creator will decide what information should be displayed as Public Parameters and the extent to which the parameter values can be modified.

After the editing of the Reactions Package is completed, the COSE will request that the Reactions Package to Validate itself against the assigned Property Package, using the <Validate> Use Case

Post-conditions:

The Reactions Package has been edited by the PE and the changes occurred in the configuration of the Reactions Package left it in a valid state (i.e. Validation passed successfully)

Errors:

Edition of the Reactions Package failed while displaying the Reactions Package graphical interface or while the PE was performing a change in the configuration of the Reactions Package.

The Reactions Package does not have a graphical interface and does not expose any Public Parameters

The changes performed by the PE left the Reactions Package in an invalid state (i.e. the Reaction rates, equilibrium constants, etc, can not be calculated because the associated Property Package is not able to provide the necessary physical properties (i.e. fugacities, activities, etc)

Uses:

<Validate>

UC-007: EDIT (REACS-07)

Actors: PE.

Priority: Medium

Pre-conditions:

The Reactions Package can be edited, e.g. it has its own graphical interface that can be displayed to visualise and/or modify the configuration of the Reactions Package

Flow of events:

The Reactions Package, as a response to a request from the COSE, shows its graphical interfaces and allows the PE to visualise the Reactions Package configuration data (i.e. Reactions participating in the Reaction set defined within the package – each Reactions Package contains only a Reaction set –, components participating in the various reactions, stoichiometry of the reactions, type of each reaction, e.g. equilibrium reaction, kinetic reaction and kinetic parameters of the reactions, i.e. pre-exponential factors, activation energies, adsorption constants, etc

At this point the PE can change the configuration of the Reactions Package by activating or deactivating some reactions, modifying the reaction expressions or the reaction parameters. There is no restriction on how may or which of these items have to be available for the PE to interact with. Each Reactions Package implementor or creator will decide what information is displayed and how much of the information can be modified.

Post-conditions:

The Reactions Package edition terminates successfully and the configuration of the Reactions Package is accepted as valid

Errors:

Failure during edition

UC-008: ASSOCIATE REACTIONS PACKAGE WITH PROPERTY PACKAGE (REACS-08)

Actors: PE

Priority: High

Classification: This Use Case is entirely contained within the Reactions Use Cases Package. There is no overlap with other packages Use Cases.

Status:

Pre-conditions:

The Reactions Package and the Property Package to which the Reactions Package is going to the assigned have been initialised successfully. For the Reactions Package this means that <Initialise> has been executed successfully.

Flow of events:

Either

The PE selects an existing Reactions Package and an existing Property Package and asks the COSE to associate them,

or

The COSE associates a Reactions Package with a Property Package because of the PE's actions – for example the PE associates a Reactions Package with a Unit Operation and the PE associates a Property Package with the same Unit.

The COSE keeps track of the association by using its own internal means. Thus, whenever the Reactions Package is requested to calculate, its thermodynamic server will be the one selected in this Use Case

After the association has been completed, the COSE will ask the Reactions Package to Validate itself against the associated Property Package, i.e. using the <Validate> Use Case

Post-conditions:

The Reactions Package and the Property Package can and have been associated

Errors:

Validation of the Reactions Package against the associated Property Package failed

Uses:

<Validate>

UC-009: ASSIGN REACTIONS PACKAGE TO UNIT (REACS-09)

Actors: PE

Priority: High

Classification: This Use Case is entirely contained within the Reactions Use Cases Package. There is no overlap with other packages Use Cases

Status:

Pre-conditions:

The PE has selected some Reaction Packages using the <Select Reactions Package> use case.

Flow of events:

The PE asks to the COSE to present a list of available Reactions Packages

The COSE creates that list and displays it

The PE chooses one of the Reactions Packages and asks the COSE to assign it to the UNIT

The COSE will use its own internal means to keep record of this association between the UNIT and the Reactions Package (note that the association between the Reactions Package and the Property Packages had been already established in <Assign Reactions Package to Property Package>

The COSE asks the UNIT to Validate itself, using the UNIT <Validate> Use Case.

The UNIT as part of its validation procedure will check that the assigned Reactions Package can provide the information the UNIT needs to solve its equations. For instance, the Reactions Set contained in the Reactions Package can be simply not applicable to a given UNIT because the number of reactions exceed the maximum number the UNIT can solve, the Reaction Set contains equilibrium reactions and the UNIT is not prepared to deal with them, etc

The important aspect of this Use Case is that it is the Use Case where the UNIT will have the opportunity to inspect the assigned Reactions Package, and inform the COSE, and even the PE – through the UNIT graphical means – on the adequacy of the Reaction Package.

Post-conditions:

The UNIT can deal with the assigned Reactions Package and the association is performed correctly

Errors:

The UNIT rejects the Reactions Package

Uses:

The UNIT <Validate>

RUNNING THE REACTIONS PACKAGE USE CASES

UC-010: SOLVE REACTIONS (REACS-10)

Actors: PE through COSE and/or UNIT

Priority: High

Pre-conditions:

A Reactions Package exists and it has been associated to a Property package

The COSE needs to use the Reactions Package for solving its own internal equations, e.g. because the Reactions Package has been applied to a COSE internal Unit Operation such as a reactor

And/or, a UNIT has been assigned a Reactions Package and needs to use it for solving its internal mass and/or energy balances

Flow of events:

This is a container Use Case that will make Use of other more specific Use Cases, as stated in the Use section of this description.

The PE requests from the COSE to initiate the solution of a given flowsheet problem.

The COSE solves the flowsheet using its own internal mechanisms. As part of this process, certain COSE native Unit Operations and/or UNIT components installed in the flowsheet (typically reactors) may use an assigned Reactions Package to complete calculation of their mass and energy balances.

Each native COSE Unit Operation (here termed generically as COSE) and UNIT requiring so, ask their corresponding Reactions Packages to provide the reaction data necessary to solve mass and energy balances – this can comprise:

- Number of reactions participating in the reaction set
- Type of each reaction (e.g. kinetic, equilibrium)
- Reactants and Products
- Base reactant (for kinetic reactions)
- Stoichiometry of each reaction
- Phase on which the reactions will be calculated
- Units in which the reaction rates will be provided (e.g. kgmole/hr-kg cat or kgmole/hr-m3, etc) (for kinetic reactions).
- Enthalpy of reaction for each one of the participating reactions.

- Equilibrium constants (for equilibrium reactions).

The COSE and/or UNIT will construct its internal mass and energy balances on the basis of the information obtained from the Reactions Package

The COSE and/or UNIT will request (in some cases such as plug-flow reactors this will happen repeatedly) that the Reactions Package calculate reaction rates, equilibrium constants, enthalpies of reactions etc, to close its mass and energy balances. The COSE and/or UNIT will typically supply temperature, pressure and molar composition to the Reactions Package (see <UNIT Solves Reactions> Use Case).

Post-conditions:

Errors:

One or more Reactions Package fails to provide the COSE and/or UNIT with the necessary information to solve be mass and/or energy balances (e.g. stoichiometric coefficients, etc).

One or more Reactions Packages fails to calculate the property requested by the client (i.e. COSE or UNIT)

Uses:

<Calculate>, <Get Reaction Set Parameters>

UC-011: GET REACTION SET PARAMETERS (REACS-11)

Actors: UNIT/COSE

Priority: High

Pre-conditions:

The COSE and/or UNIT is solving its internal equations

Flow of events:

The UNIT/COSE requests that a Reactions Package provide the description of its reactions (as described in <Solve Reactions>).

The Reactions Package delivers this information to the client.

Post-conditions:

Errors:

Fails providing information.

UC-012: UNIT SOLVES REACTIONS (REACS-12)

Actors: UNIT

Priority: High.

Flow of events:

As in <Solve Reactions> Use Case.

Extends:

<Solve Reactions>, but the actor is a CAPE-OPEN Unit Operation (i.e. UNIT)

UC-013: CALCULATE (REACS-13)

Actors: UNIT/COSE

Priority: High

Pre-conditions:

The COSE and/or UNIT are/is solving its internal equations.

Flow of events:

The COSE or UNIT requests that the Reactions Package calculate a given Reaction Property (see <Solve Reactions> Use Case)

The Reactions Package is passed the necessary information to perform the requested calculation (e.g. T, P and Molar Fractions) solves its equations to provide the requested information (e.g. reaction rates, equilibrium constants, etc). To do so, the Reactions Package may need to get some physical properties (e.g. fugacities for a non-ideal gas reaction set). If this is the case, the Reactions Package can use its associated Properties Package to retrieve this information.

Post-conditions:

The Reactions Package provided the requested reaction property

Errors:

The Reactions Package failed to provide the requested reaction property

SAVING/RESTORING THE REACTIONS PACKAGE USE CASES

UC-014: SAVE REACTIONS PACKAGE (REACS-14)

Actors: COSE

Priority: Medium

Pre-conditions:

The Reactions Package has been correctly initialised; i.e. the <Initialise> Use Case has been executed successfully.

The flowsheet is being saved by the COSE (e.g. as a request from the PE)

Flow of events:

This Use Case does not consider saving data of those Reactions Packages belonging to a Reactions Package Manager. It is considered that in this case the Reactions Package Manager is responsible for doing this and no CAPE-OPEN mechanism is specified for this use case.

The COSE asks the Reactions Packages installed in the flowsheet to save their data in a location provided by the COSE

The Reactions Packages do so

Post-conditions:

The Reactions Packages have saved their data

Errors:

The Reactions Packages failed to save their data

UC-015: RESTORE REACTIONS PACKAGE (REACS-15)

Actors: COSE

Priority: Medium

Pre-conditions:

The Reactions Package has been correctly initialised; i.e. the <Initialise> Use Case has been executed successfully.

The flowsheet is being recovered from file by the COSE (e.g. as a request from the PE)

Flow of events:

This Use Case does not consider recovering data of those Reactions Packages belonging to a Reactions Package Manager. It is considered that in this case the Reactions Package Manager is responsible for doing this and no CAPE-OPEN mechanism is specified for this use case.

The COSE asks the Reactions Package installed in the flowsheet to recover their data in a location provided by the COSE

The Reactions Packages do so

Post-conditions:

The Reactions Packages have recovered their data

Errors:

The Reactions Packages failed to recover their data

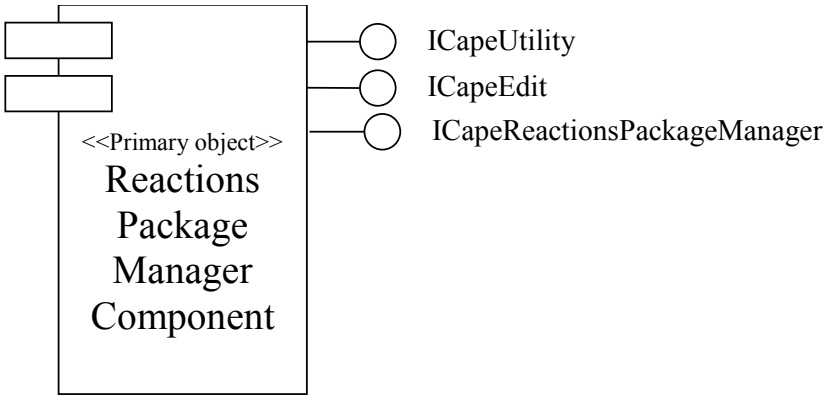
2.3 Sequence diagrams

3. Analysis and Design

This section presents a component view of the Reactions Package and Reaction Package Manager and describes the role of a reaction object in more detail. The interactions between Reaction Packages, reaction object, material object and a PME are expanded upon and related back to the Use Cases described earlier.

3.1 Overview

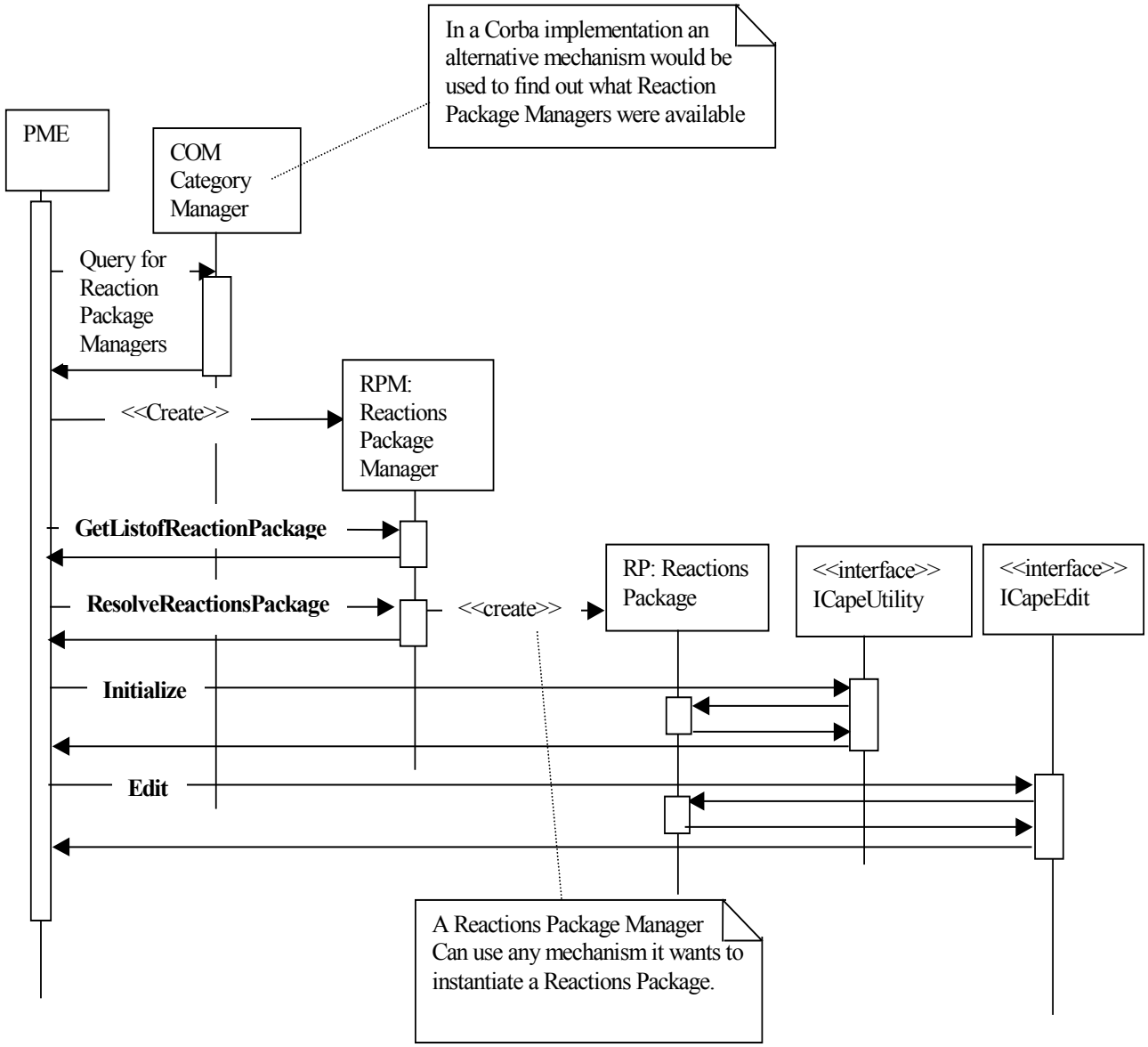
3.1.1 Reactions Package Manager Component



A Reactions Package Manager Component is a primary CAPE-OPEN component. This means that it may support the ICapeUtility, ICapeEdit and ICapeCollection interfaces if it is appropriate to do so. Refer to the Methods & Tools Integrated Guidelines [4] for more information on CAPE-OPEN primary objects.

A Reactions Package Manager must support the ICapeReactionsPackageManager interface so that client can discover what Reaction Packages are managed by the Reactions Package Manager and so that a client can request the instantiation of a particular Reactions Package.

3.1.2 Interactions between a Reactions Package Manager and a PME



This sequence diagram shows the realisation of the following use cases in terms of the components and interfaces defined in this specification:

<Select Reactions Package> REACS-01

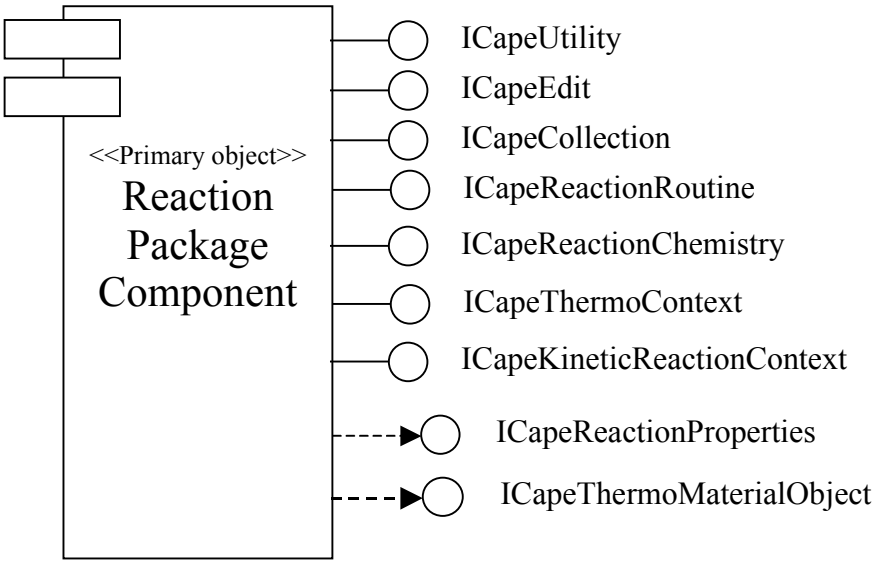
<Get List of Reactions Packages> REACS-02

<Resolve Reactions Package> REACS-03

<Initialise> REACS-04

<Edit Reactions Package> REACS-06

3.1.3 Reactions Package Component

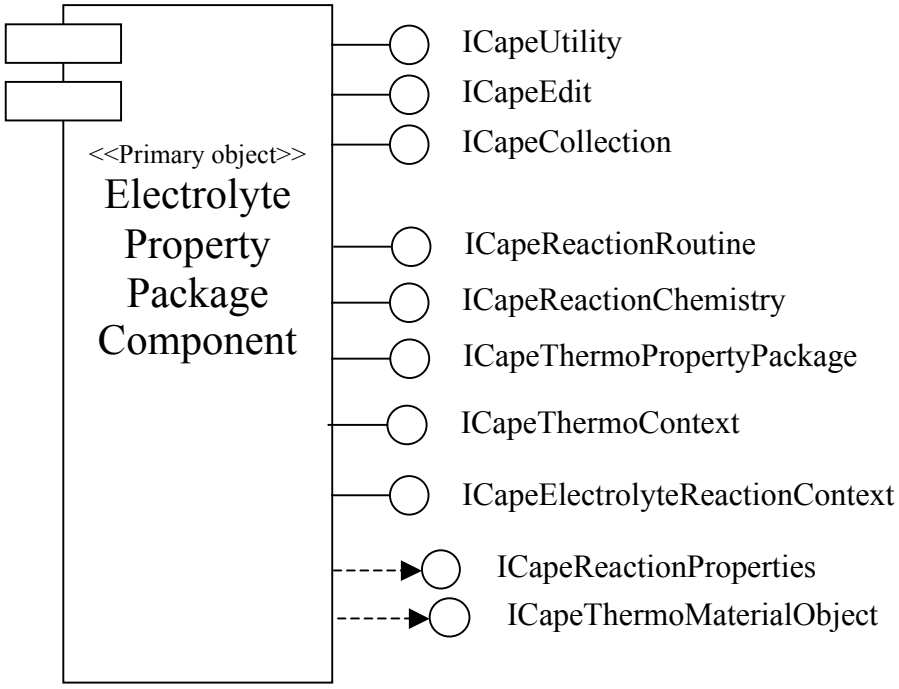


A Reactions Package component is a primary CAPE-OPEN object. This means that it may support the ICapeUtility, ICapeEdit and ICapeCollection interfaces if it is appropriate. See *Methods & Tools Integrated Guidelines* for more information on CAPE-OPEN primary objects.

A Reactions Package component must also support the following interfaces:

- ICapeReactionRoutine so that a client can request Reaction Property calculations
- ICapeReactionChemistry so that a client can query for the description of the system of reactions the package contains
- ICapeThermoContext so that a client can pass an ICapeThermoMaterialObject interface to the package so that the client and the package can exchange thermodynamic property values.
- ICapeKineticReactionContext so that a client may pass an ICapeReactionProperties interface to the package so that the client and the package can exchange reaction property values.

3.1.4 Physical Property Package with Electrolytes



An Electrolyte Physical Property Package Component is a CAPE-OPEN Primary component. It supports the interfaces of a Physical Property Package component but in addition it supports the ICapeReactionRoutine, ICapeReactionChemistry and ICapeElectrolyteReactionContext interfaces. These extra interfaces allow a client to request the calculation of electrolyte reaction properties and to query the property package for the electrolyte reactions it contains.

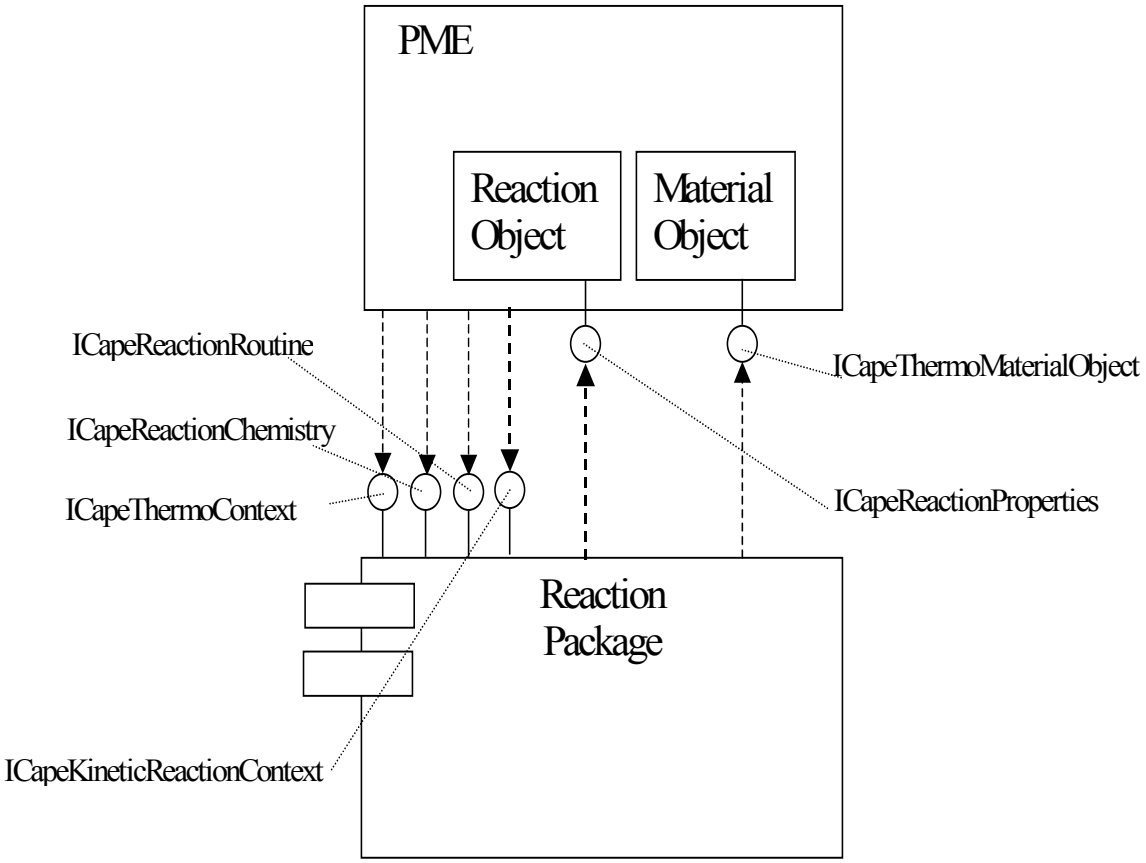
Like a Physical Property Package component an Electrolyte Physical Property Package must be passed an ICapeThermoMaterialObject interface so that property values can be exchanged with a client. It also supports the ICapeElectrolyteReactionContext so that the client can pass it an ICapeReactionProperties interface so that reaction property values can be exchanged.

An Electrolyte Property Package supports the ICapeReactionChemistry interface but it is expected that this interface will only be used where a numerical solution requires the construction of a system of equations to represent the reactions.

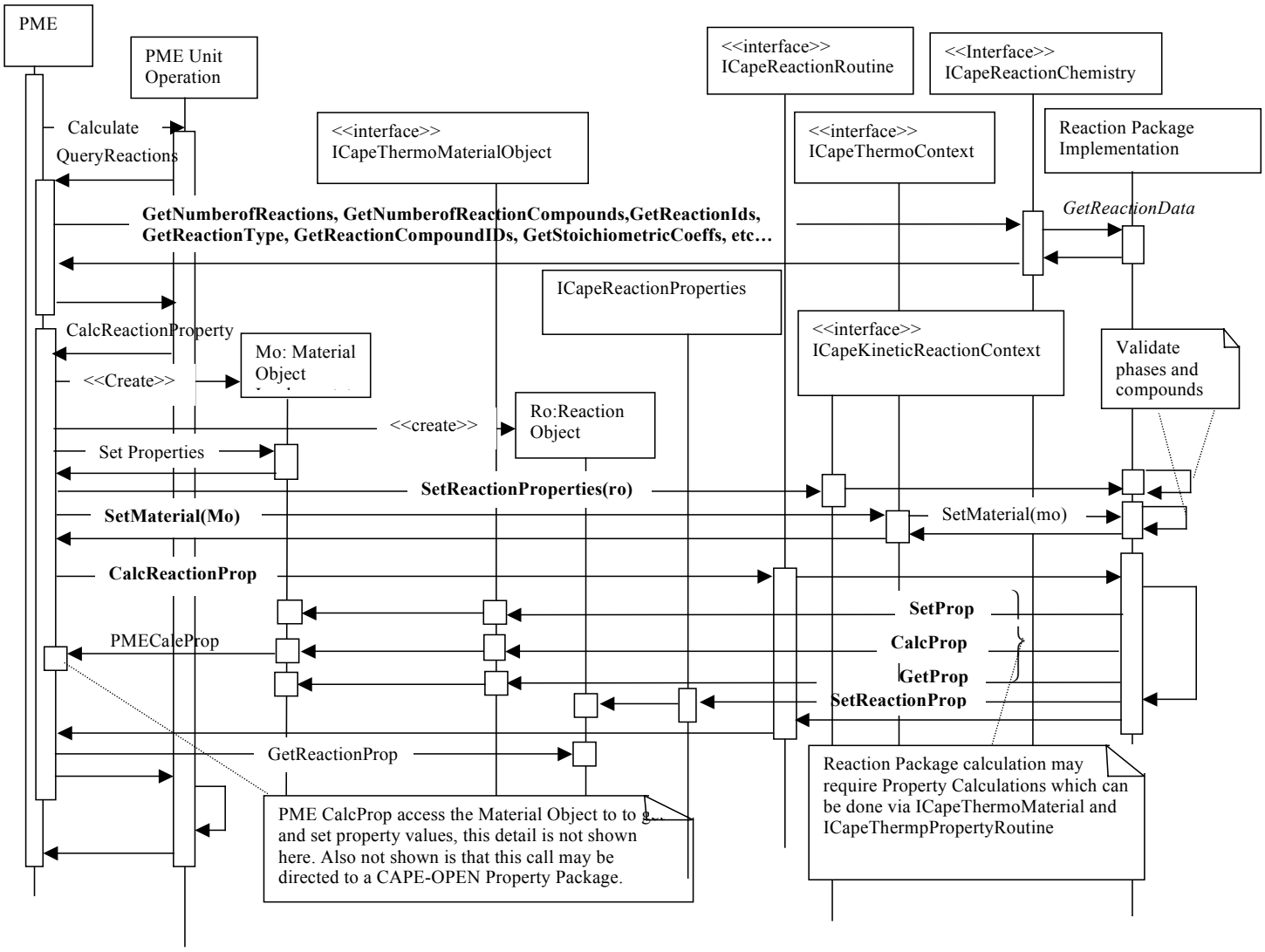
The equilibrium calculation functionality provided by an Electrolyte Property Package is expected to take account of the effect of the electrolyte reactions defined by the package.

3.1.5 Interactions between PME and a Reactions Package

This diagram shows the principal interfaces used in the interaction between a PME and a Reactions Package Component. The PME communicates directly with the Reactions Package component through its CAPE-OPEN interfaces. The Reactions Package component communicates with the PME via the Material Object interfaces and the reaction object interfaces.



The following sequence diagram shows the same interaction in more detail, starting with a request from a PME to a Unit Operation, possibly a reactor, to perform a calculation.



This sequence diagram shows the realisation of the following Use cases:

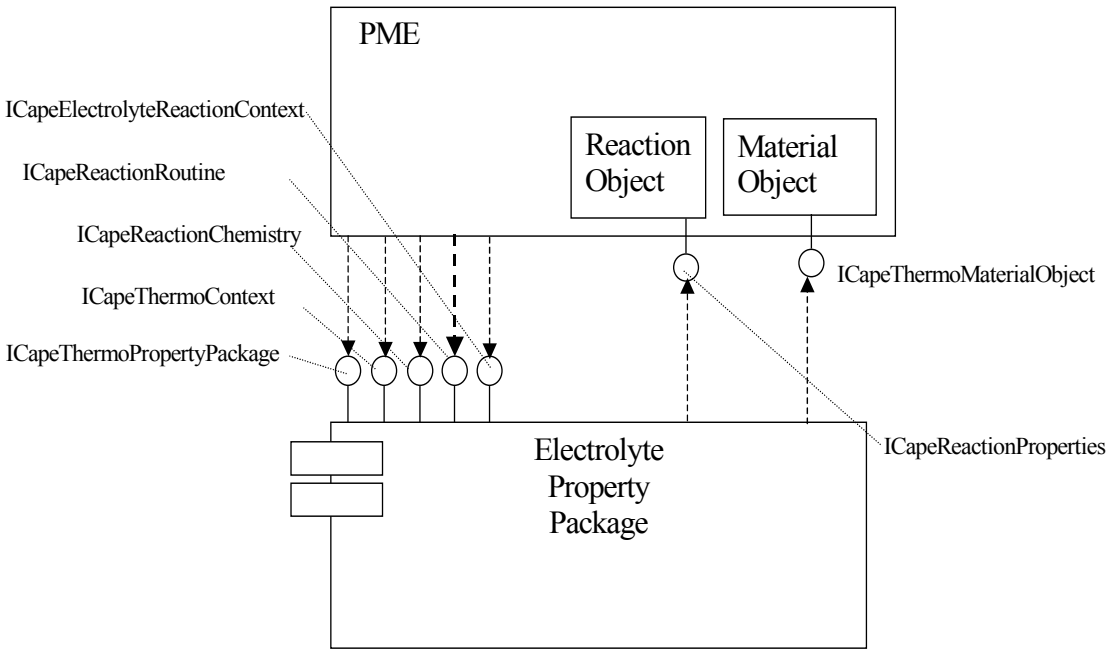
<Solve Reactions> REACS-10

<Calculate> REACS-12

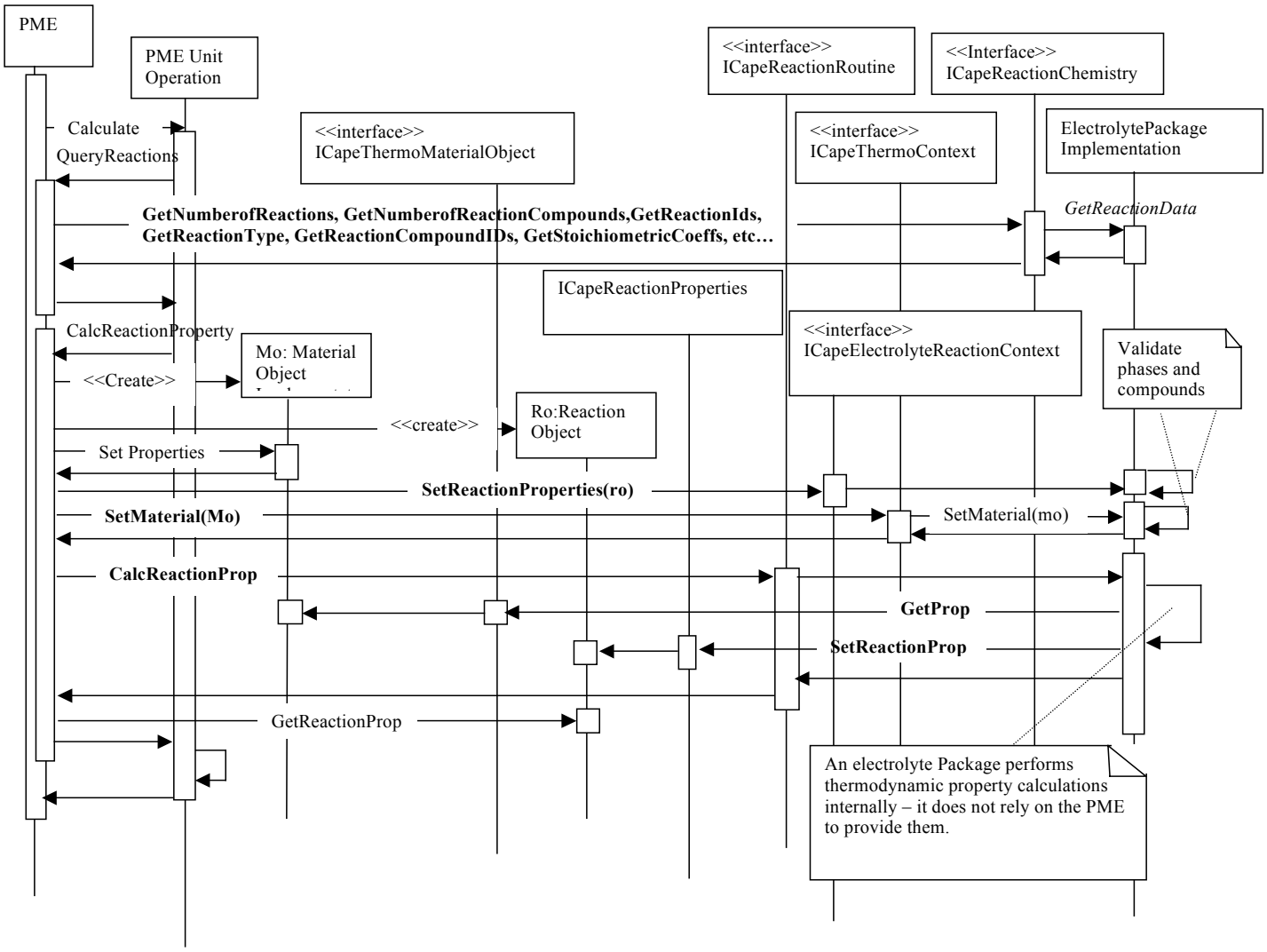
<Validate> REACS-05

3.1.6 Interactions between PME and Physical Property Package with Electrolytes

The interaction between a PME and an electrolyte Property Package is very similar to the interaction between a PME and a non-electrolyte Property Package. The Package takes account of the reactions when calculating a property or performing an equilibrium calculation.



The following sequence diagram shows these interactions in more detail. Note that the next diagram is the same as the one for the interaction between a PME and a Reactions Package component, except that Reactions Package component is replaced by Electrolyte Property Package component. The Electrolyte package is able to perform its own thermodynamic Property calculations, and so does not rely on the PME to provide this service.



This sequence diagram shows the realisation of the following Use cases for the case of a set of electrolyte reactions:

<Solve Reactions> REACS-10

<Calculate> REACS-12

<Validate> REACS-05

3.1.7 Interactions between a PME and a CAPE-OPEN Unit using reactions.

This diagram shows the interfaces used in the interaction between a COSE and a CAPE-OPEN Unit operation that can accommodate both kinetic and equilibrium calculation systems. In this interaction a PME must pass to the CAPE-OPEN Unit, two reaction objects:

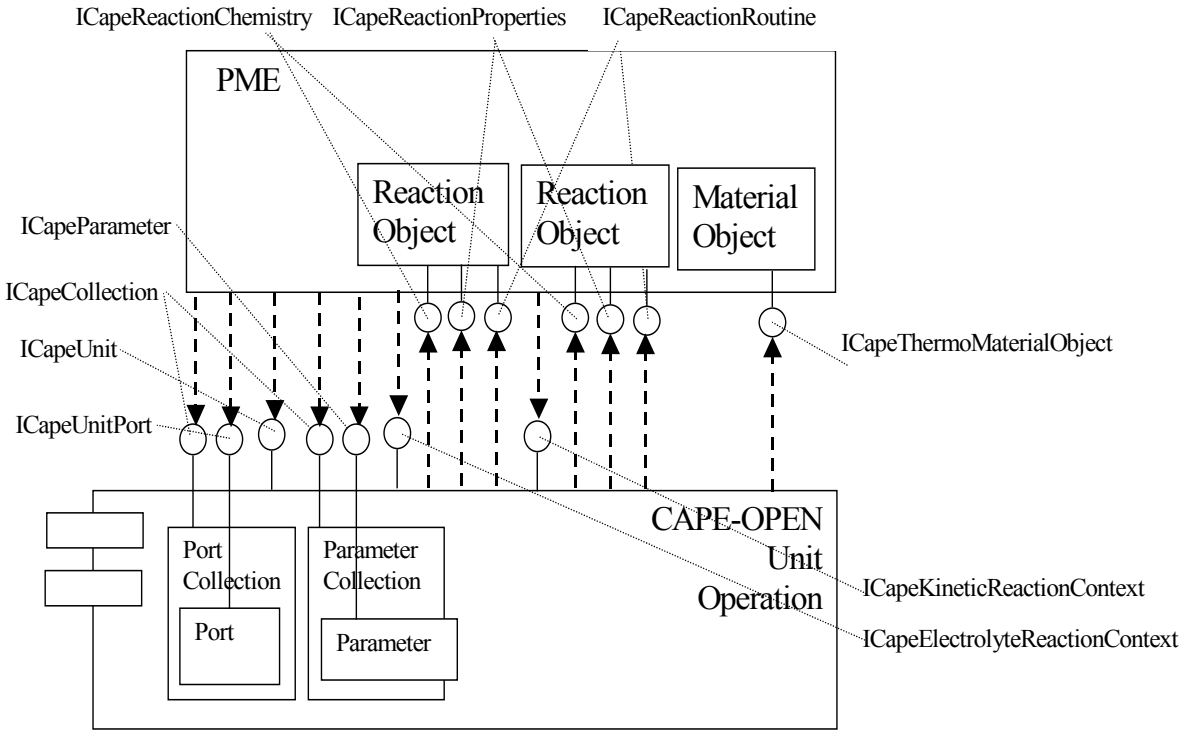
- One reaction object is for the set of kinetic reactions associated with the Unit Operation. This reaction object holds the values of the kinetic reaction properties, gives access to the kinetic reaction stoichiometry and kinetic reaction property calculations. It is passed to the Unit Operation via the ICapeKineticReactionContext interface

- One reaction object is for the set of electrolyte equilibrium reactions associated with the unit operation. This reaction object holds the values of the electrolyte reaction properties; gives access to electrolyte reaction stoichiometry and electrolyte property calculations. It is passed to the Unit Operation via the ICapeElectrolyteReactionContext interface.

Note that a Unit Operation can chose whether to implement the ICapeKineticReactionContext and ICapeElectrolyteReactionContext interfaces or not. If neither is implemented then the PME will not be able to pass reaction objects to the Unit. If kinetic reactions have been associated with the Unit Operation, and the Unit Operation supports the ICapeKineticReactionContext interface then the PME must construct a reaction object for the kinetic reactions and pass it to the Unit before requesting a calculation. Similarly, if electrolyte reactions have been associated with the Unit Operation, and the Unit Operation supports the ICapeElectrolyteReactionContext interface then the PME must construct a reaction object for the electrolyte reactions and pass it to the Unit before requesting a calculation.

A reaction object passed to a Unit Operation must support the ICapeThermoContext interface. This allows the Unit Operation to associate a Material Object (and thereby a Property Package) with a set of reactions in order to support reaction property calculations. It is the PME's responsibility to ensure that the material object associated with a reaction object is passed to the appropriate Reactions Package because there is no method to allow a Reactions Package to query for the material object associated with a reaction object. This design realises the <Associate Reactions Package with Property Package> (REACS-08).

A reaction object passed to a Unit Operation must also support the ICapeReactionProperties, ICapeReactionChemistry and ICapeReactionRoutine interfaces, because it must behave as a Reactions Package as well as a Reaction Object to the Unit. The implementation of these interfaces will forward all calls to the PME. If the PME is using its own Reaction calculation and configuration capabilities then the calls will access them. On the other hand if the PME is using a CAPE-OPEN Reaction Package then all calls will be forwarded to it.



The next sequence diagram shows the interactions between these component and objects in outline. It shows how a CAPE-OPEN Unit operation, uses the interfaces implemented by a Reaction Object to request Reaction Property Calculations from an external Reaction Package as part of its own calculation. It shows the role of the PME in mediating between the two external components, and the transfer of data via a material object and a reaction object. Since the Reaction Package is passed a material object it can request Physical Property calculations if necessary.

The interaction for the electrolyte reaction case is the same except that an Electrolyte Property Package replaces the Reaction Package.

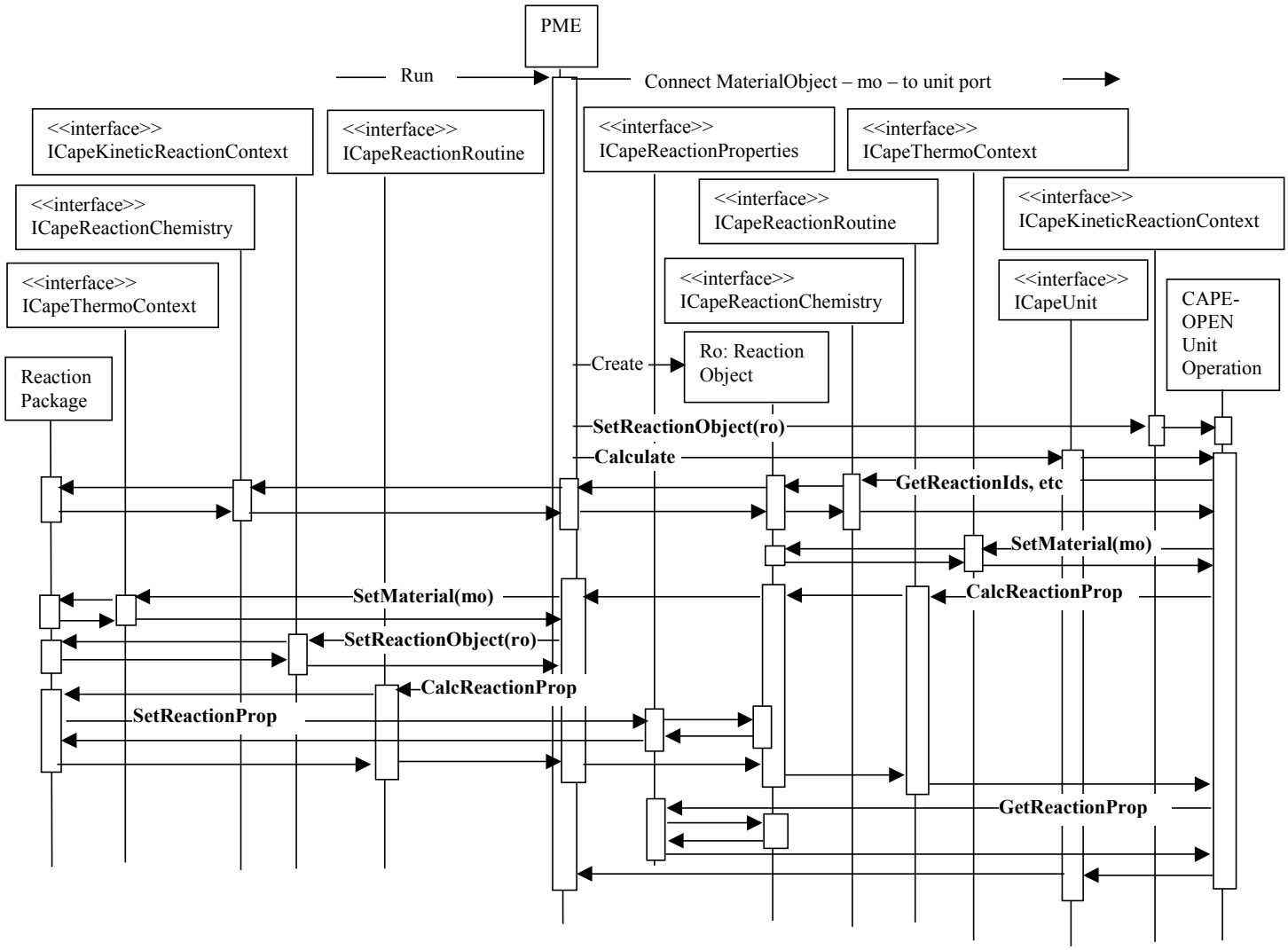
The diagram shows the realisation of the following Use cases:

<Associate Reaction Package with Property Package> REACS-08

<Assign Reaction Package to UNIT> REACS-09 <Solve Reactions> REACS-10

<Get Reaction Set Parameters> REACS-11 <UNIT Solves Reactions> REACS-11

<Calculate> REACS-12



3.2 Sequence diagrams

3.3 Interface diagrams

3.4 State diagrams

3.5 Other diagrams

3.6 Interface desciptions

3.6.1 ICapeReactionsPackageManager

Interface Name	ICapeReactionsPackageManager
Method Name	GetListOfReactionsPackages
Returns	CapeArrayString

Description

Returns a list of the names of all Reactions Packages available within the Reactions Package Manager.

Arguments

None

Errors:

ECapeInvalidArgument – Raised if PackageName does not identify a Reactions Package known to the Reactions Package Manager

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

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

ECapeFailedInitialisation

Interface Name	ICapeReactionsPackageManager
Method Name	ResolveReactionsPackage
Returns	CapeInterface

Description

Returns the Reactions Package specified by the client of the Reactions Package Manager.

Arguments

Name	Type	Description
[in] reactionsPkg	CapeString	The name of the reactions routine to be resolved

Errors:

ECapeInvalidArgument – Raised if PackageName does not identify a Reactions Package known to the Reactions Package Manager

ECapeFailedInitialisation

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

3.6.2 ICapeReactionChemistry

Interface Name	ICapeReactionChemistry
Method Name	GetNumberOfReactions
Returns	CapeLong

Description

Gets the number of reactions contained in the Reactions Package.

Arguments

None

Errors:

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

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

ECapeInvalidArgument

ECapeFailedInitialisation

Interface Name	ICapeReactionChemistry
Method Name	GetReactionsIds
Returns	CapeArrayString

Description

Returns the identifiers of all the reactions contained within the Reactions Package.

Arguments

None

Errors:

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

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

ECapeInvalidArgument

ECapeFailedInitialisation

Interface Name	ICapeReactionChemistry
Method Name	GetReactionType
Returns	CapeReactionType

Description

Returns the type of a particular reaction contained in the Reactions Package. CapeReactionType constants for the various reaction:

CAPE_EQUILIBRIUM = 0,

CAPE_KINETIC = 1,

Arguments

Name	Type	Description
[in] reacId	CapeString	The reaction identifier

Errors:

ECapeInvalidArgument – Raised if the id passed in does not identify any reaction known to the reactions Package.

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

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

ECapeFailedInitialisation

Interface Name	ICapeReactionChemistry
Method Name	GetNumberOfReactionCompounds
Returns	CapeLong

Description

Gets the number of compounds occurring in a particular reaction within a Reactions Package.

Arguments

Name	Type	Description
[in] reacID	CapeString	The reaction identifier

Errors:

ECapeInvalidArgument – Raised if the id passed in does not identify any reaction known to the reactions Package.

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

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

ECapeFailedInitialisation

Interface Name	ICapeReactionChemistry
Method Name	GetReactionCompoundIds
Returns	--

Description

Get the identifiers of the components participating in the specified reaction within the reaction set defined in the Reactions Package.

Arguments

Name	Type	Description
[in] reacId	CapeString	The reaction identifier
[out] compIDS	CapeArrayString	List of compound IDs
[out] compCharge	CapeArrayDouble	The charge for each compound
[out] compCASNumber	CapeArrayString	The CAS Registry numbers for the compounds

Notes:

This method returns both compound name and CAS registry number. The CAS Registry number should be used to identify the compounds for validation purposes because it is unambiguous.

Errors:

ECapeInvalidArgument – Raised if the id passed in does not identify any reaction known to the Reactions Package.

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

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

ECapeFailedInitialisation

Interface Name	ICapeReactionChemistry
Method Name	GetStoichiometricCoefficients
Returns	CapeArrayDouble

Description

Returns the stoichiometric coefficients of the specified reaction (positive numbers indicate products, negative numbers indicate reactants)

Arguments

Name	Type	Description
[in] reacID	CapeString	The reaction identifier

Notes:

The array of coefficients returned by this method is parallel to the array returned by calling GetReactionCompuoundIds meaning that the first coefficient corresponds to the first compound and so on.

Errors:

ECapeInvalidArgument – Raised if the id passed in does not identify any reaction known to the Reactions Package.

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

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

ECapeFailedInitialisation

Interface Name	ICapeReactionChemistry
Method Name	GetReactionPhase
Returns	CapeString

Description

Gets the phase on which a particular reaction contained in the Reactions Package will take place.

Arguments

Name	Type	Description
[in] reacID	CapeString	The reaction identifier

Notes:

The string returned by this method must match one of the phase labels known to the Property Package.

Errors:

ECapeInvalidArgument – Raised if the id passed in does not identify any reaction known to the Reactions Package.

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

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

ECapeFailedInitialisation

Interface Name	ICapeReactionChemistry
Method Name	GetReactionRateBasis
Returns	CapeReactionRateBasis

Description

Gets the phase on which the reactions contained in the package will take place. The reaction rate basis (i.e. "Homogeneous" or "Heterogeneous") Homogeneous reactions will be provided in kgmole/h/m3 and heterogeneous will be provided in kgmole/h/kg-cat.

CapeReactionRateBasis:

CAPE_HOMOGENEOUS = 0,

CAPE_HETEROGENEOUS = 1,

Arguments

Name	Type	Description
[in] reacID	CapeString	The reaction identifier

Errors:

ECapeInvalidArgument – Raised if the id passed in does not identify any reaction known to the Reactions Package.

ECapeNoImpl – The operation is "not" implemented even if this method can be called for reasons of compatibility with the CO standard. That is to say that the operation exists but it is not supported by the current implementation.

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

ECapeFailedInitialisation

Interface Name	ICapeReactionChemistry
Method Name	GetReactionConcBasis
Returns	CapeString

Description

Gets the concentration basis required that will be used by a particular reaction in its rate equation.

Qualifiers defined in the THRM spec can be used here (i.e. “fugacity”, “moleFraction”, etc)

Arguments

Name	Type	Description
[in] reacID	CapeString	The reaction identifier

Errors:

ECapeInvalidArgument – Raised if the id passed in does not identify any reaction known to the reactions Package.

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

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

ECapeFailedInitialisation

Interface Name	ICapeReactionChemistry
Method Name	GetBaseReactant
Returns	CapeString

Description

Returns the name of the base reactant for a particular reaction.

Arguments

Name	Type	Description
[in] reacID	CapeString	The reaction identifier

Errors:

ECapeInvalidArgument – Raised if the id passed in does not identify any reaction known to the reactions Package.

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

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

ECapeFailedInitialisation

Interface Name	ICapeReactionChemistry
Method Name	GetPhaseCompounds
Returns	CapeString

Description

Returns the number and ids of the compounds in the specified phase.

Arguments

Name	Type	Description
[in] phase	CapeString	Label of the required phase
[out] compNo	CapeLong	
[out] compIds	CapeArrayString	The ids of the compounds present in the specified phase.

Errors:

ECapeInvalidArgument – Raised if the id passed in does not identify any phase known to the Reactions Package.

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

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

ECapeFailedInitialisation

Interface Name	ICapeReactionChemistry
Method Name	GetReactionParameters
Returns	(CapeInterface) ICapeCollection

Description

Returns a collection containing the rate expression parameters for a particular reaction.

Arguments

Name	Type	Description
[in] reacId	CapeString	Identifier of a particular reaction

Notes

GetReactionParameters returns a collection of CAPE-OPEN parameters [6] that characterize the rate expression used by the reaction model in a Reaction Package. For a PowerLaw model this collection would contain parameters for activation energy, pre-exponential factor and compound exponents for example. It is up to the Reactions Package implementor to decide whether a client can update the values of these parameters. If this operation is allowed, then the implementor must also provide support for persistence [5] interfaces, so that the updated values can be saved and restored. In this case the COSE is also responsible for calling the persistence methods.

Deliberately, the standard does not define the names of the parameters that may appear in such a collection, even for well-known reaction models, such as PowerLaw and Langmuir – Hinshelwood – Hougen – Watson (LHHW). This is because the formulation of well-known models is not fixed, and because the standard needs to support custom models as well as the well-known models.

This decision is not expected to be restrictive: in most cases the (software) client of a Reactions Package does not need to know what model the package implements and what parameters it has. However, the parameters may be of interest to an end-user who wants to adjust or estimate the parameter values. In these cases the COSE can invoke the Reaction Package's own GUI, or, if it doesn't have one, present the parameters in a generic grid. It is the Reaction Package implementor's responsibility to provide documentation for the parameters so that an end-user can understand how they are used.

Errors

ECapeInvalidArgument – Raised if the id passed in does not identify any phase known to the Reactions Package.

ECapeNoImpl – The operation is "not" implemented even if this method can be called for reasons of compatibility with the CO standard. That is to say that the operation exists but it is not supported by the current implementation.

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

ECapeFailedInitialisation

3.6.3 ICapeReactionsRoutine

Interface Name	ICapeReactionsRoutine
Method Name	CalcReactionProp
Returns	--

Description

The Reactions Package is passed a list of reaction properties to be calculated, the reaction IDS for which the properties are required, and the calculation basis for the reaction properties (i.e. mole or mass). A material object containing the thermodynamic state variables that need to be used for calculating the reaction properties (e.g. T, P and compositions) is passed separately via a call to the setMaterial method of the Reaction Package's ICapeThermoContext interface.

The results of the calculation will be written to the reaction object passed to the Reactions Package via either the ICapeKineticReactionContext interface for a kinetic reaction package, or the ICapeElectrolyteReactionContext interface for an Electrolyte Property Package.

Arguments

Name	Type	Description
[in] props	CapeArrayString	The Reaction Properties to be calculated.
[in] phase	CapeString	The qualified phase for the results.
[in] reacIds	CapeArrayString	The qualified reactions for the results. NULL to specify all reactions in the set.
[in] basis	CapeString	Qualifies the basis of the result (i.e., mass /mole). Default is mole. Use NULL only as a placeholder for properties for which basis does not apply.

Errors:

ECapeInvalidArgument – Raised if any of the properties, phases, reaction ids or the basis is invalid.

ECapeNoImpl – The operation is "not" implemented even if this method can be called for reasons of compatibility with the CO standard. That is to say that the operation exists but it is not supported by the current implementation.

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

ECapeFailedInitialisation

3.6.4

3.6.4

3.6.4 ICapeReactionProperties

Interface Name	ICapeReactionProperties
Method Name	GetReactionProp
Returns	CapeArrayDouble

Description

Gets the value of the specified reaction property within a reactions object. The qualifiers passed in determine the reactions, phase and calculation basis for which the property will be got. The order of the array is the same as in the passed in reacIds array (i.e. property value for reaction reacIds[1] will be stored in property[1])

Arguments

Name	Type	Description
[in] property	CapeString	The Reaction Property to be got.
[in] phase	CapeString	The qualified phase for the Reaction Property.
[in] reacIds	CapeArrayString	The qualified reactions for the Reaction Property. NULL to specify all reactions in the set.
[in] basis	CapeString	Qualifies the basis of the Reaction Property (i.e., mass /mole). Default is mole. Use NULL only as a placeholder for property for which basis does not apply. This qualifier could be extended with values such as activity, fugacity, fractions, molality…This way when an equilibrium constant is requested its basis can be specified

Errors:

ECapeInvalidArgument – Raised if any of the properties, phases, reaction ids or the basis is invalid.

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

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

ECapeFailedInitialisation

Interface Name	ICapeReactionProperties
Method Name	SetReactionProp
Returns	--

Description

Sets the values of the specified reaction property within a reactions object. The qualifiers passed in determine the reactions, phase and calculation basis for which the property will be got

Arguments

Name	Type	Description
[in] property	CapeString	The Reaction Property to be got.
[in] phase	CapeString	The qualified phase for the Reaction Property.
[in] reacIds	CapeArrayString	The qualified reactions for the Reaction Property. NULL to specify all reactions in the set.
[in] basis	CapeString	Qualifies the basis of the Reaction Property (i.e., mass /mole). Default is mole. Use NULL only as a placeholder for property for which basis does not apply. This qualifier could be extended with values such as activity, fugacity, fractions, molality…This way when an equilibrium constant is requested its basis can be specified
[in] propVals	CapeArrayDouble	The values of the requested reaction property. The order of the array is the same as in the passed in reacIds array (i.e. property value for reaction reacIds[1] will be stored in property[1])

Errors:

ECapeInvalidArgument – Raised if any of the properties, phases, reaction ids or the basis is invalid.

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

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

3.6.5 ICapeKineticReactionContext

Interface Name	ICapeKineticReactionContext
Method Name	SetReactionObject
Returns	--

Description

Used to pass the ICapeReactionProperties interface of a reaction object to a component that needs to access the properties of a set of kinetic reactions.

Arguments

Name	Type	Description
[in] reactionObject	CapeInterface	The ICapeReactionProperties interface of a reaction object.

Errors:

ECapeInvalidArgument – Raised if the reaction object is NULL or invalid in some way.

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

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

ECapeFailedInitialisation

3.6.6 ICapeElectrolyteReactionContext

Interface Name	ICapeElectrolyteReactionContext
Method Name	SetReactionObject
Returns	--

Description

Used to pass the ICapeReactionProperties interface of a reaction object to a component that needs to access the properties of a set of electrolyte reactions.

Arguments

Name	Type	Description
[in]reactionObject	CapeInterface	The ICapeReactionProperties interface of a reaction object.

Errors:

ECapeInvalidArgument – Raised if the reaction object is NULL or invalid in some way.

ECapeNoImpl – The operation is "not" implemented even if this method can be called for reasons of compatibility with the CO standard. That is to say that the operation exists but it is not supported by the current implementation.

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

ECapeFailedInitialisation

3.6.7 ICapeThermoContext

This interface should be implemented by all Thermodynamic and Physical Properties components that need an ICapeThermoMaterial interface in order to set and get a material's property values. The following methods are described in this section:

Interface Name	ICapeThermoContext
Method Name	SetMaterial
Returns	--

Description

Allows the client of a component that implements this interface to pass an ICapeThermoMaterialObject interface to the component, so that it can access the properties of a material and request property calculations.

Arguments

Name	Type	Description
[in] materialObject	CapeInterface	The interface of an object support the ICapeThermoMaterialObject interface.

Notes

The SetMaterial method allows a Reactions component to be given the ICapeThermoMaterialObject interface of a Material Object. This interface gives the component access to the description of the material for which Property calculations are required. A component can also use the ICapeThermoMaterialObject interface to to get lists of components and phases.

Errors

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

ECapeInvalidArgument – the input argument is not a valid CapeInterface, or the input argument object passed does not support the ICapeThermoMaterialObject interface.

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

ECapeFailedInitialisation

3.6.8 Reaction properties and qualifiers

This section describes the list of reaction properties whose calculation can be requested through the CalcReactionProp method and whose values can be got through calls to the GetReactionProp method.

Usage of the methods

Function Arguments	**Description**

Property Name	The name of the property
Phase Label	The phase label of the phase upon which the reactions will operate
Reaction Identifier	The identifier of the reaction that needs to be calculated/set/got
Calculation Basis	The basis of the property (mass or mole)

Reaction types

Reaction Types	**Description**
Equilibrium	Equilibrium reactions on a specified phase. This type applies for all electrolyte reactions, and can apply for other reactions.
Kinetic	Kinetic reactions on a specified phase

List of properties

Property Names	**Description**	**Units**
ReactionRate	Reaction rate for kinetic reactions	
ChemicalEquilibriumConstant	The phase label of the phase upon which the reactions will operate	
EnthalpyOfReaction	The identifier of the reaction that needs to be calculated/set/got	J
SolubilityIndex	Solubilty Index (SI) $SI = \frac{\sum_{i=1}^{NC} a_i \nu_{ij}}{K_j}$	
SolubilityProduct	Solubility Product (Kj) $K_j = \prod_{i}^{NC} a_i^{\nu_{ij}} \qquad i = 1, NC, j = 1, N$	

List of Reaction Compounds

Compound Names	**Description**
Compounds	Chemical species as defined in [1]
Ionic Species	+ve and –ve ions from association and dissociation reactions
SolidComplex	Solid (salt) compounds composed of complexes of

	compounds
Radicals	Used in representing kinetic reaction mechanisms

List of Stoichiometric Coefficients

Coefficient Types	**Description**
+ve integer	Product compound
-ve integer	Reactant compound
0	Inert compound

List of Reaction Phases

The list phases that should be supported by a Reactions Package are as defined in [1]

3.7 Scenarios

4. Interface Specifications

4.1 COM IDL

```
// You can get these intructions in Reactions.idl file from CAPE-OPENv1-0-0.zip
```

4.2 CORBA IDL

```
// You can get these intructions in CAPE-OPENv1-0-0.idl within the
CAPEOPEN100::Business::PhyProp::Reactions module
```

5. Notes on the interface specifications

6. Prototypes implementation

AspenTech will prototype the following scenario, which is required for the development of a CAPE-OPEN interface to the FLUENT CFD system.

A CAPE-OPEN Unit operation provides a wrapper around an existing Unit Operation Model. The existing model is capable of doing its own reaction calculations but the CAPE-OPEN Unit Operation wrapper needs to pass to it all the data needed to perform those calculations.

This prototype will allow the CAPE-OPEN Fluent interface to read the complete reaction description of a system of reactions from a PME and to pass it to the FLUENT program. The prototype will require the development of a reaction object and will validate:

- that the specifications covers all the data required to describe a reaction system
- That the interfaces allow the necessary interactions between a Unit operation and a PME to occur.

7. Specific Glossary Terms

8. Bibliography

(i) CAPE-OPEN Interface Specifications: Thermodynamic and Physical Properties Version 1.0

(ii) CAPE-OPEN Interface Specifications: Unit Operation Specification

(iii) CAPE-OPEN Interface Specifications: Numerical Solvers

(iv) Methods & Tools Integrated Guidelines

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

(vi) CAPE-OPEN Interface Specifications: Parameter Common Interfaces

9. Appendices