CAPE-OPEN

Expanding Process Modelling Capability
Through Software Interoperability Standards

Flowsheet Monitoring

Interface Specification

CO-LaN

www.colan.org

ARCHIVAL INFORMATION

Filename	Monitoring Interface Specification.docx
Authors	CO-LaN consortium
Status	Public release
Date	July 2019
Version	version 1.58
Number of pages	76
Versioning	
	Version 1.55 edited on September 27, 2017
	Version 1.56 edited on November 1, 2018
	Version 1.57 edited on May 22, 2019
	Version 1.58 edited on July 8, 2016
Additional material	
Web location	
Implementation specifications version	
Comments	

IMPORTANT NOTICES

SUMMARY

Flowsheet Monitoring is the ability to access all elements in a flowsheet without interfering with the flowsheet. Flowsheet Monitoring Components are software components that can be plugged into a flowsheet and have access to all thermodynamic property calculation methods, all Streams and all Unit Operations.

Flowsheet Monitoring Components gain read-only access to flowsheet elements via Collection interfaces exposed by a new interface, *ICapeFlowsheetMonitoring*, implemented by the Simulation Context. The Flowsheet Monitoring Component can be invoked via the methods of its *ICapeFlowsheetMonitoringComponent* interface, or by means of responding to events. The event handlers are methods on a new interface, *ICapeFlowsheetMonitoringEventSink*, that are invoked by the PME.

ACKNOWLEDGEMENTS

The initial Flowsheet Monitoring interface specification was designed by Jasper van Baten (AmsterCHEM) and Bill Barrett (United States Environmental Protection Agency). Subsequent revisions by the Methods & Tools Special Interest Group of the CAPE-OPEN Laboratories Network (CO-LaN) led to the current document.

CONTENTS

LIST OF FIGURES

CAPE-OPEN Document Roadmap

This document belongs to the documentation set of CAPE-OPEN interface specifications in its version 1.1. This imposes that for Microsoft COM Flowsheet Monitoring Components implementing the interfaces defined in this document, the *CapeVersion* registry key must be set at the value “1.1”.

The interfaces specified in this document are intended to work with version 1.1 of the Thermodynamic and Physical Properties interface specification and with future versions of CAPE-OPEN interfaces in general including UNIT. Version 1.0 of the Thermodynamic and Physical Properties interfaces have been deprecated, and support for them is not considered by this specification.

This document is intended primarily for software engineers, who are interested in developing CAPE-OPEN compliant Flowsheet Monitoring Components (FMCs), or providing support for these components in Process Modelling Environments (PMEs).

All other readers need not go beyond **Section 2 Requirements.**

1. Introduction

Steady-state flowsheet simulation is a computer technique used by process engineers for modeling entire (chemical) processes or parts thereof. Such flowsheet simulations typically consist of several process model units (unit operations) linked together by material, energy, and information streams. Underlying the unit operation models and streams, there are thermodynamic and physical property models. Flowsheet simulations are used to model a complete chemical process to - for example - assess process design, train process operators, and to close mass and energy balances. The unit operation models are combined in a Process Flow Diagram, referred to as a Flowsheet, showing connections between unit operations. Throughout this document the connections between unit operations will be referred to as Streams.

This document describes the interface requirements for a broad class of process-based applications that require process information that can most effectively be obtained directly from the Flowsheet. Typically, these applications do not modify the Flowsheet itself, but rather acquire information about the process being investigated directly from objects in the Flowsheet. Flowsheet monitoring is a mechanism for providing this direct access to the objects contained in the Flowsheet. These applications typically perform analysis of the Flowsheet after it has been calculated to a converged state. Examples of applications that can benefit from the direct access to the Flowsheet include:

- applications that perform overall Flowsheet calculations, including component-, mass-, and energy balances. This category includes environmental monitoring applications such as the WAR algorithm [1, 2];

- applications that perform generic thermodynamic and physical property calculations in the context of a Flowsheet, without the need for any information on the Flowsheet structure. Example applications include phase envelope calculators and residue curve calculators;

- applications that perform thermodynamic and physical property calculations in the context of specific Flowsheet elements. These applications perform post-processing and validation at the found solution. Example applications include wax formation calculations or hydrate deposition calculations in pipes represented by specific Streams;

- applications that conduct performance rating calculations for a particular class of unit operation;

- applications that analyze the structure of the Flowsheet and the specific coupling of Flowsheet elements, for example for heat integration by pinch technology [3];

- archiving applications that store each found solution of a particular Flowsheet to a database (goal could be generation of short-cut models by linearizing between previously found rigorous solutions);

- applications that make use of previously archived solutions to advise the user on an initial starting point for a Flowsheet solution. Such an application can be executed by the Flowsheet User prior to solving the Flowsheet;

- model-based control applications that perform a simulation of the current state of an actual physical process and want to report new control targets to an external application. A separate mechanism is required for this class of applications to provide the simulation with measurements of the current state of the process (e.g. feed conditions);

- applications that perform synthesis studies on a Flowsheet and advise the User on possible optimal structure. These applications act on the structure of the Flowsheet and possibly not on any other piece of information so these applications can be executed by the Flowsheet User prior to solving the Flowsheet.

In the past, a common practice for obtaining information about material flows within a Flowsheet has involved creating a collection of pass-through unit operations that collect stream information without performing operations on the Flowsheet Streams. These unit operations are typically placed strategically within the

Flowsheet to collect flow information and can only collect information from the Streams that they are placed within. This approach has severe drawbacks: if information is required from various places of the Flowsheet, this includes inserting multiple monitoring unit operations, convoluting the actual flow diagram. Moreover, each unit operation must perform an equilibrium calculation on each of the outlet Streams, which can substantially increase the time it takes to calculate the Flowsheet. Additionally, the Flowsheet User cannot change the monitored Streams without invalidating the Flowsheet. As a result, the solution status of the Flowsheet is affected by the action of inserting such Unit Operations. Flowsheet Monitoring provides a generic solution for a broad class of process evaluation and improvement applications to gain access to all parts of the Flowsheet, without changing the state of the Flowsheet (in a read-only manner). Such components will be called Flowsheet Monitoring Components.

Flowsheet Monitoring Components gain read-only access to Flowsheet elements via collection interfaces exposed by a new interface, *ICapeFlowsheetMonitoring*, implemented by the Simulation Context. The Flowsheet Monitoring Component can be invoked via its *Monitor* method, or by means of responding to events. The event handlers are methods on a new interface, *ICapeFlowsheetMonitoringEventSink*, that are invoked by the PME.

2. Requirements

This section lists the functionalities expected from Flowsheet Monitoring Components then gathers textual requirements for Flowsheet Monitoring Components, PMEs and Unit Operations. The relationship between Flowsheet elements and Flowsheet Monitoring Components is also listed and described.

2.1 Flowsheet monitoring

The following functionality is available to a Flowsheet Monitoring Component:

- access to the thermodynamic and physical property system (e.g. enumeration of compounds, phases, …);
- ability to perform physical and thermodynamic property calculations (on duplicated Material Objects or on Material Objects obtained from the Material Template System);
- access to the Stream Collections of a Flowsheet;
- for each of the material Streams, access to all its properties (pressure, temperature, composition, flow, phase equilibrium);
- for each of the Information Streams access to the parameter-based data;
- for each of the Energy Streams, access to the parameter-based data;
- access to a Collection describing all Unit Operations present in the Flowsheet;
- ability to determine connectivity of Unit Operations (determine which Unit Operations are connected by which Streams);
- access to all Public Unit Parameters exposed by a Unit Operation;
- ability to perform event driven calculations (viz. automatically performing a calculation when a new solution of the Flowsheet is found);
- ability to be invoked manually;
- ability to be invoked by the Flowsheet Solver as part of solving the Flowsheet.
- ability to be configured by the Flowsheet User through a GUI.
- ability for storing and loading (persistence) along with the Flowsheet.

To maintain a modular design, existing interfaces already defined by CAPE-OPEN are used to get information from the PME. For instance, Stream data is acquired using the appropriate thermodynamic interfaces, while Unit Operation Ports and Parameters are obtained through the Unit Operation's Port and Parameter Collections.

The Flowsheet Monitoring Component is only allowed to present a modal dialog during *ICapeUtilities::Edit*. This restriction includes the use of message boxes.

ICapeFlowsheetMonitoringComponent::Monitor may present results in the form of output Public Parameters, or through a non-CAPE-OPEN defined mechanism (file Input/Outputs for example), or may not present any output at all but merely update its internal state for later presentation of the results.

When a PME supports Flowsheet Monitoring, it is expected that at least invocation of *ICapeUtilities::Edit* is supported by the PME. During *Edit*, monitoring may be exercised so Stream and Unit Collections should be available to the Flowsheet Monitoring Component and functional at that time. If the task of monitoring is itself interactive and requires always a GUI, then *Edit* is the only place where monitoring is exercised and *ICapeFlowsheetMonitoringComponent::Monitor* may remain un-implemented.

If the Flowsheet Monitoring Component has output Public Parameters, then the PME may choose to take the values of these Output Parameters as variables in the Flowsheet equations. As a consequence, the Flowsheet Monitoring Component must then be evaluated using *ICapeFlowsheetMonitoringComponent::Monitor* during the resolution of the Flowsheet. Although to the PME it is clear which parts of the Flowsheet depend on the output Parameters of the Flowsheet Monitoring Component, the Flowsheet Monitoring Component may depend on any other pieces of information provided by the Flowsheet.

2.2 Textual requirements

Textual requirements are listed hereafter and referenced by mentioning the software component to which each requirement applies and its number in the global list of requirements. The acronym "FMC" in a requirement reference relates to a Flowsheet Monitoring Component, the acronym "PME" refers to a Process Modelling Environment and the acronym "UO" refers to a Unit Operation.

REQ-FMC-01: a Flowsheet Monitoring Component is a PMC Primary Object.

Rationale: A Flowsheet Monitoring Component is created directly by the PME and consequently is a PMC Primary Object.

REQ-FMC-02: Flowsheet Monitoring Components must not modify any Flowsheet element.

Rationale: Flowsheet Monitoring Components are intended to obtain information about elements in the Flowsheet, and to make use of the information contained within these elements directly without making changes to the Flowsheet itself. By making the PMCs on the Flowsheet available to the Flowsheet Monitoring Components through existing CAPE-OPEN interfaces, it is therefore possible to modify these objects. It is important to stress that although the Flowsheet Monitoring Components will have access to interfaces that allow modification of the Flowsheet elements, the access to these elements *is restricted to read-only*.

For example, Parameter values can be obtained, but must not be set or reset, Unit Operation Ports may not be disconnected, Unit Operations may not be calculated, Stream values may not be modified and no thermodynamic or physical property calculations or Phase Equilibrium calculations may be performed directly on the Material Objects that represent material Streams. Modifications (and the consequences thereof) made by Flowsheet Monitoring Components to the Flowsheet elements cannot be foreseen by the PME and may disrupt normal operations of the Flowsheet. Further, changes of these values will invalidate the Flowsheet and the Flowsheet will no longer be in a solved state.

REQ-FMC-03: Flowsheet Monitoring Components must not connect or disconnect Streams from Ports of Unit Operations.

Rationale: disconnecting Streams from Unit Operations would constitute a change to the Flowsheet. Connecting a Stream to a Unit Operation would likewise constitute a change to the Flowsheet. Since such changes are prohibited, Flowsheet Monitoring Components must not connect or disconnect Streams from Unit Operations.

REQ-FMC-04: Flowsheet Monitoring Components must request thermodynamic and physical property calculations only on duplicates of Material Objects.

Rationale: thermodynamic and physical property calculations can be performed by Flowsheet Monitoring Components but must not be performed on the original Material Objects, but rather on duplicates thereof.

REQ-PME-05: all Material Objects must implement support for CAPE-OPEN Thermodynamic and Physical Properties interface standard 1.1 or higher.

Rationale: version 1.1 supersedes version 1.0.

REQ-FMC-06: all Flowsheet Monitoring Components that use thermodynamics must support CAPE-OPEN Thermodynamic and Physical Properties interface standard 1.1 or higher.

Rationale: version 1.1 supersedes version 1.0.

REQ-PME-07: all Stream objects must implement *ICapeIdentification* interface.

REQ-PME-08: all Unit Operations must implement *ICapeIdentification* interface.

REQ-PME-09: PME must enforce that no two items of the Stream Collection carry the same name.

REQ-PME-10: PME must enforce that no two items of the Unit Operation Collection carry the same name.

REQ-PME/UO-11: all Ports of any Unit Operation must implement *ICapeIdentification* interface.

REQ-PME/UO-12: no two items within the Port Collection of any single Unit Operation may carry the same name.

Rationale (for **07** *through* **12***): ICapeIdentification* is used to identify Streams, Unit Ports and Unit Operations. Therefore, it is important that Stream names are unique (no two Streams with the same name should be present in a Flowsheet), that Unit Operation names are unique (no two Unit Operations with the same name should be present in a Flowsheet) and that Port names within a given Unit Operation are unique. The Flowsheet User can generally change names of Streams and Unit Operations in the Flowsheet. The Collection Owner has the responsibility of disallowing duplicate names within a Collection. The PME owns the Collections of Streams and Unit Operations. A Unit Operation owns its Collection of Ports. Internal Unit Operations are solely managed by the PME.

The Streams are made available to the Flowsheet Monitoring Component as a Collection of CAPE-OPEN objects and the Unit Operations are made available to the Flowsheet Monitoring Component as a Collection of Unit Operations.

REQ-PME-13: all Stream objects must implement *ICapeStream* interface.

Rationale: type of Streams as well as connectivity need to be identifiable by the Flowsheet Monitoring Component. The *ICapeStream* interface gives access to the Stream type and to upstream as well as downstream connections.

REQ-PME-14: all Streams must be represented as CAPE-OPEN objects.

Rationale: generally, when a PME has CAPE-OPEN support, it already has an implementation to represent a material stream as a CAPE-OPEN Material Object; these CAPE-OPEN Objects can be exposed as-is, without the need for an additional CAPE-OPEN implementation to represent the Streams. For any Stream type for which support of CAPE-OPEN is not already present, the CAPE-OPEN representation may be partial; methods that modify the Stream content do not need to be implemented. See table 1 for a list of methods that do not need to be implemented.

For more information on which interface(s) must be implemented on any Stream present in Collections provided by PME, see the relevant CAPE-OPEN interface specification documents.

REQ-PME-15: Flowsheet Monitoring Components must be able to request thermodynamic calculations.

Rationale: the ability to use the thermodynamic sub-system of the PME is necessary for any Flowsheet Monitoring Component in order to perform a full range of operations as part of the monitoring process.

Although streams may have a partial CAPE-OPEN implementation, this requirement implies that Material Objects obtained through *ICapeThermoMaterial::CreateMaterial* must have a fully functional implementation of a Material Object.

The Material Object should be functional to the point it has been configured. This implies that property calculations must be available for a Material Object that corresponds to a thermodynamic sub-system for which compounds, phases and property methods are defined.

REQ-PME-16: all Unit Operations must be represented as CAPE-OPEN Unit Operations.

Rationale: the Flowsheet Monitoring Component requests access to the Port and Parameter Collections of the Unit Operation. Furthermore, the Flowsheet Monitoring Component requests access to the members of these Collections, represented as appropriate CAPE-OPEN objects. Therefore, a CAPE-OPEN representation of the Unit Operation object must be exposed by the PME. In accordance with REQ-FMC-02, not all methods of this

CAPE-OPEN representation need to be functional. For example, *ICapeUnit::Calculate* does not need to be implemented since *Calculate* modifies the state of the Unit Operation. See table 1 for a list of methods that do not need to be implemented.

CAPE-OPEN Unit Operations may be exposed directly to the Flowsheet Monitoring Component without any additional container.

For more information on which interface must be implemented on any Unit Operation present in Collections provided by PME, see the relevant CAPE-OPEN interface specification documents [4].

REQ-FMC-17: a Flowsheet Monitoring Component must not call methods modifying objects in any Collection provided by PME.

Rationale: this is a direct consequence of REQ-FMC-02. Methods that must not be called by the Flowsheet Monitoring Component consequently need not be implemented on objects that are *only* exposed to a Flowsheet Monitoring Component. Since in this case the methods are not called, the return code is inconsequential.

The following table lists methods that do not need to be implemented and must not be called by any Flowsheet Monitoring Component. Please note the explicit exception mentioned in REQ-PME-15.

Object	Interface	Methods
Material Object in Stream Collection	*ICapeThermoMaterial*	*ClearAllProps*
		SetOverallProp
		SetPresentPhases
		SetSinglePhaseProp
		SetTwoPhaseProp
	ICapeThermoPropertyRoutine	*CalcSinglePhaseProp*
		CalcTwoPhaseProp
	ICapeThermoEquilibriumRoutine	*CalcEquilibrium*
Unit Operation	*ICapeUnit*	*Calculate*
		Validate
	ICapeUnitPort	*Connect*
		Disconnect
Parameter	*ICapeParameter*	*put_value*
		Reset
		Validate
		put_Mode
Any object	*ICapeIdentification*	*put_ComponentName*
		put_ComponentDescription
	ICapeUtilities	*Terminate*
		Initialize
		Edit
		setSimulationContext
	*IPersist**	*Load*
		InitNew
		Save (with fClearDirty flag)

Table 1

Other interfaces have the potential to modify the Material Stream content. Examples are the Petroleum Fractions and the Chemical Reaction interfaces. The same requirement holds: no function needs to be implemented that modifies the content of a Stream since a Flowsheet Monitoring Object is not allowed to call them.

For Unit Operations that have been inserted as a CAPE-OPEN object into the Flowsheet, a CAPE-OPEN interface is readily available as implemented by the Unit Operation software component itself. Such an interface can be readily exposed via the Unit Operation Collection. For Unit Operations in the Flowsheet that are not CAPE-OPEN objects, a layer needs to be present around any non-CAPE-OPEN Unit Operation to expose the Unit Operation as an *ICapeUnit* interface (with a Collection of Port objects and Parameter objects).

The implementation of this CAPE-OPEN layer around non-CAPE-OPEN Unit Operations however can be limited; because of the read-only nature of Flowsheet Monitoring, many methods of the Unit Operation interfaces need not be implemented. For example, *ICapeUnit::Calculate* does not require an implementation. The functionality that the CAPE-OPEN layer around non-CAPE-OPEN Unit Operations must implement is:

- Expose a name (and possibly description) via *ICapeIdentification*
- Expose a Collection of Parameters via *ICapeUtilities*
- Expose a Collection of Ports via *ICapeUnit*
- Optionally expose reports via *ICapeUnitReport* (see REQ-FMC-18 below)
- Expose error interfaces if an error can occur in any of the implemented functionality

Such Unit Operation implementations should not be registered as CAPE-OPEN Unit Operations that can be instantiated.

The Parameter implementation requires obtaining the Parameter specification and the Parameter value; the Flowsheet Monitoring Component should not change the value of any Parameter of any Unit Operation. Consequently, *SetValue* and *Reset* do not require to be functional for Parameter implementations that are only accessed by Flowsheet Monitoring Components.
The *ICapeUnitPort* implementations require (read-only) implementation of *ICapeIdentification* and exposing the connected object (*get_ConnectedObject*). The Flowsheet Monitoring Component should not attempt to rename, connect or disconnect a Port. Consequently *ICapeUnitPort::Connect* and *ICapeUnitPort::Disconnect* do not require implementation for Port implementations that are only accessed by Flowsheet Monitoring Components.

REQ-FMC-18: the Flowsheet Monitoring Component must maintain the selected report on any Unit Operation.

Rationale: obtaining a report from *ICapeUnitReport* implemented by a Unit Operation involves modifying the Unit Operation to select the current report. As an exception to REQ-FMC-02, a Flowsheet Monitoring Component is allowed to make this change on the Unit Operation in order to ask the Unit Operation to produce any report. However, if a Flowsheet Monitoring Component changes the selected report, the PME is not aware of it. Consequently, the Flowsheet Monitoring Component must restore the originally selected report prior to returning control to the PME, if the Flowsheet Monitoring Component changed the selected report.

REQ-FMC-19: Flowsheet Monitoring Component registers for the events it requires.

Rationale: A Flowsheet Monitoring Component may need one or several events to be fired by the PME in order to perform its activity. Consequently, there is a need for the Flowsheet Monitoring Component to figure out if any given event is available or not. If the Flowsheet Monitoring Component would not function because the event(s) it requires are not supported, *ICapeUtilities::Initialize* will not return successfully.

REQ-PME-20: Flowsheet Monitoring Component should be able to get the list of events supported by the PME.

Rationale: this allows the Flowsheet Monitoring Component to query for event support without subscribing to events.

2.3 Architecture

A Flowsheet Monitoring Component is a primary Process Modeling Component (PMC) that can be loaded into a Process Modeling Environment (PME). Flowsheet Monitoring Components that are installed on a system are registered using the following COM Category ID [13]:

```
CATID_MONITORING_COMPONENT = {7BA1AF89-B2E4-493d-BD80-2970BF4CBE99}
```

As a PMC and Primary Object, a Flowsheet Monitoring Component implements the common CAPE-OPEN interfaces *ICapeUtilities* [5] and *ICapeIdentification* [6], and optionally interfaces allowing for object persistence

[7]. The *ICapeIdentification* interface allows the component to be identified by name and description. The *ICapeUtilities* interface allows for general operations such as configuration of the component (*Edit*), initialization (*Initialize*) and termination (*Terminate*), exposing a Parameter Collection [8], and access to the Simulation Context. If configuration changes can be made through the *Edit* method, persistence should be supported for the configuration changes to be saved. Figure 1 shows an overview of objects accessible by Flowsheet Monitoring Components via the Simulation Context. Each of the objects shown is identified by name using *ICapeIdentification*.

The Flowsheet Monitoring Component gains access to the *ICapeFlowsheetMonitoring* interface of the PME through the Simulation Context. The *ICapeFlowsheetMonitoring* interface exposes the Stream Collections (*GetStreamCollection*) of any (material-, information-, energy-) streams in the Flowsheet, as well as the Unit Operation Collection (*GetUnitOperationCollection*) of the Flowsheet. Each of these Collections can be enumerated using the *ICapeCollection* interface. The Stream Collections allows enumeration of the material-, energy- and information-Streams available in the Flowsheet, using the interfaces appropriate to the type of Stream[4, 10, 11, 12]. This allows the Flowsheet Monitoring Component to duplicate Material Objects and perform physical and thermodynamic property calculations through any duplicated Material Object. The Flowsheet Monitoring Component can also enumerate the Unit Operations, access their Public Unit Parameters and Ports, and obtain the Streams connected to Ports. This enables the Flowsheet Monitoring Component to identify attached Streams by name, and by comparison with the names of the enumerated Streams, determine the connectivity of the Flowsheet elements. The PME also exposes the validity and calculation status of the Flowsheet to the Flowsheet Monitoring Component, which can be used to determine whether it is worthwhile for the Flowsheet Monitoring Component to perform its evaluation of the Flowsheet.

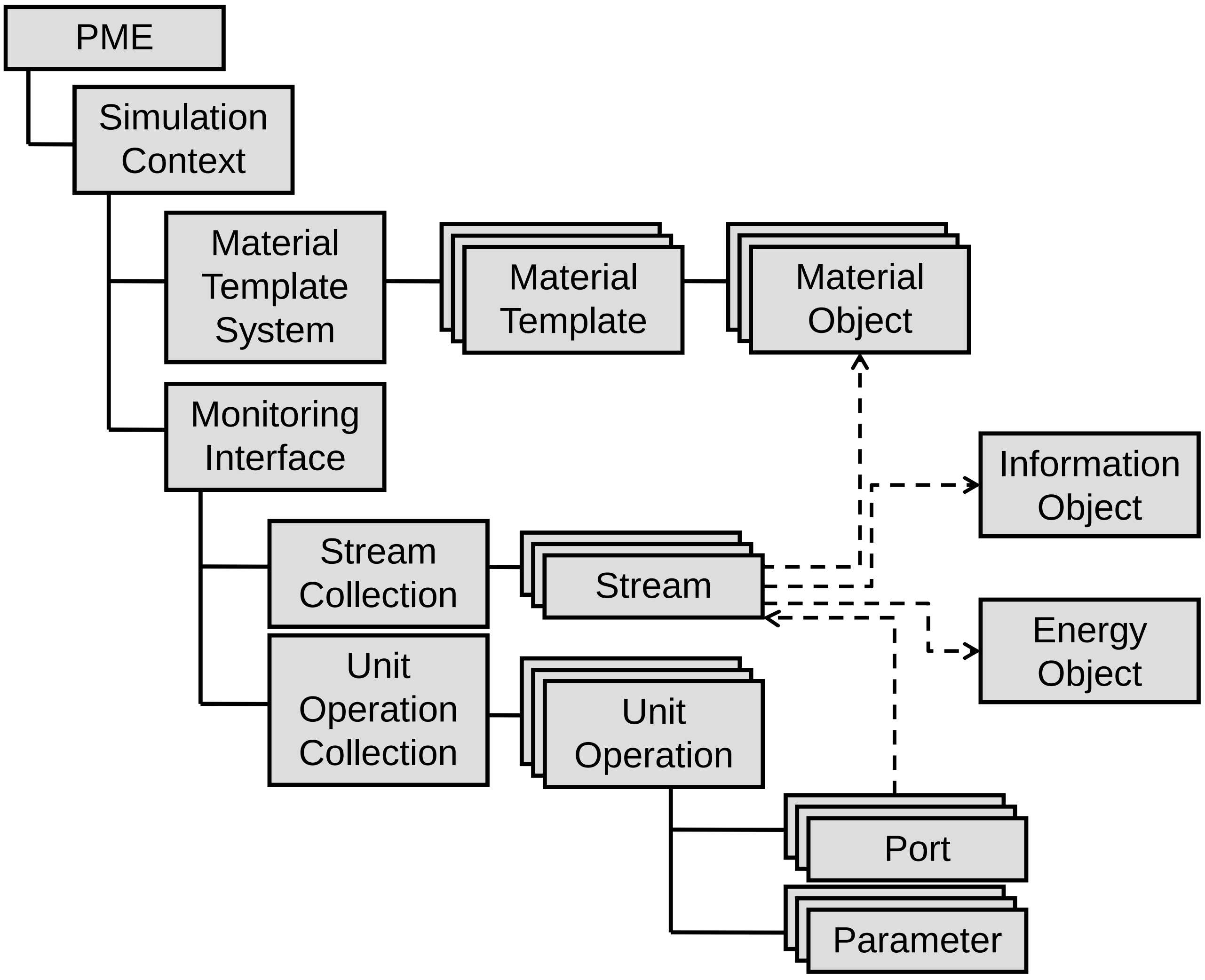


Figure 1: hierarchy of objects accessible by Flowsheet Monitoring Components via the Simulation Context.

The above satisfies all requirements for flowsheet monitoring, as stated earlier, except for the requirement to perform event driven calculations. The only event available from the above discussion is user invocation by means of the *ICapeFlowsheetMonitoringComponent::Monitor* method. The applications of online control and storage systems exemplify the requirement for events fired by the PME and handled by the Flowsheet Monitoring Component.

To this purpose, another new interface definition is required: *ICapeFlowsheetMonitoringEventSink*. This interface is implemented by the Flowsheet Monitoring Component, only if the Flowsheet Monitoring Component has a need to handle events. For each of the available events, the *ICapeFlowsheetMonitoringEventSink* interface has a specific entry point that is to be called by the PME. The events include changes to the Flowsheet structure, changes with respect to the state of a Flowsheet element, and changes in the solution state of the Flowsheet. Events include:

- *A Unit Operation has been added*; this event occurs after insertion of a Unit Operation into the Flowsheet.
- *A Unit Operation has been removed*; this event occurs after removal of a Unit Operation from the Flowsheet.
- *A Unit Operation has been renamed*; this event occurs after a Unit Operation’s name has changed (which may be a result of invoking *ICapeUtilities::Edit* on the Unit Operation). The name change event is relevant because Unit Operations are identified by name.
- *A Stream has been added*; this event occurs after insertion of a Stream into the Flowsheet.
- *A Stream has been removed*; this event occurs after removal of a Stream from the Flowsheet.
- *A Stream has been renamed*; this event occurs after a Stream’s name has changed. The name change event is relevant because Streams are identified by name.
- *A Stream has been connected or disconnected*; the event arguments identify the Stream, the Unit Operation and the Port where connection or disconnection took place. This event occurs after successfully connecting or disconnecting a stream.
- *The solution status of the Flowsheet has changed*; this event occurs after the solution status has changed, e.g. when the Flowsheet has solved or if the solved Flowsheet has been changed by the Flowsheet User.
- *The time integration of the Flowsheet has progressed to the next time step*; this event occurs upon completion of each time step in a dynamic simulation.

The two interfaces described in this document to facilitate flowsheet monitoring are *ICapeFlowsheetMonitoring* and *ICapeFlowsheetMonitoringEventSink*. *ICapeFlowsheetMonitoring* is implemented by the PME and exposed via the Simulation Context object. *ICapeFlowsheetMonitoringEventSink* is optional and may be implemented by the Flowsheet Monitoring Component. If implemented, PMEs that support event driven operation call the appropriate *ICapeFlowsheetMonitoringEventSink* methods.

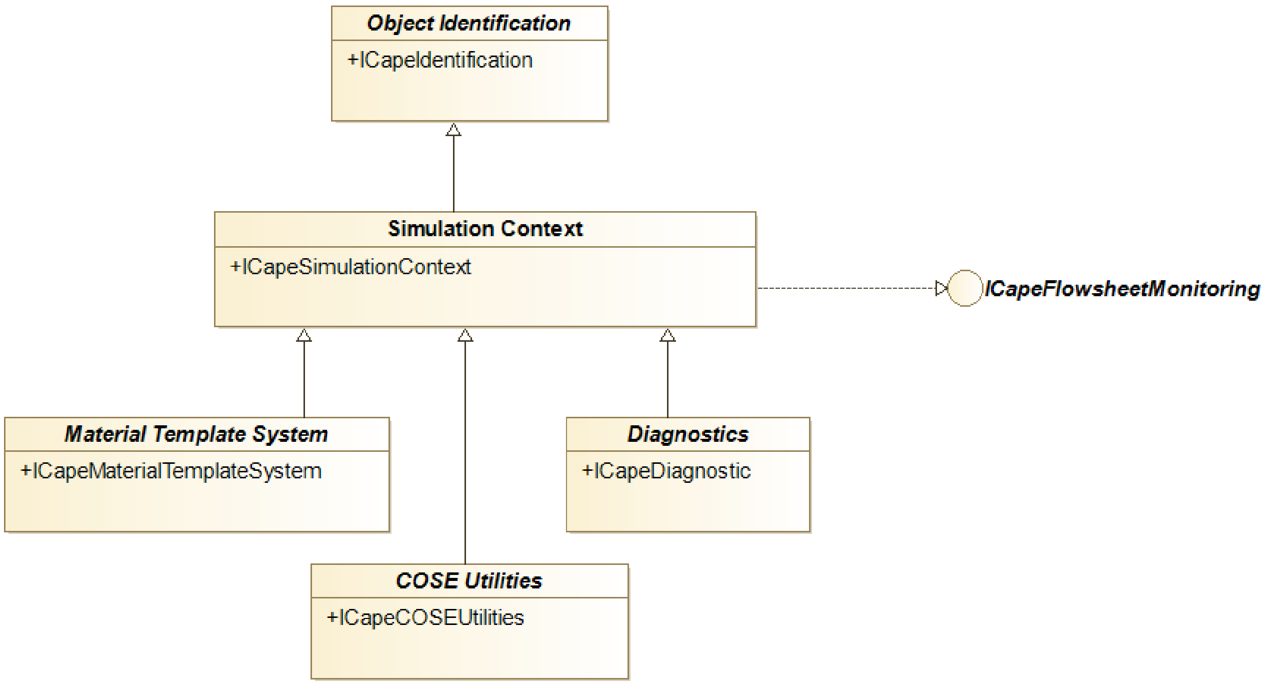


Figure 2: Interfaces implemented by the Simulation Context. Not all PMEs implement ICapeMaterialTemplateSystem.

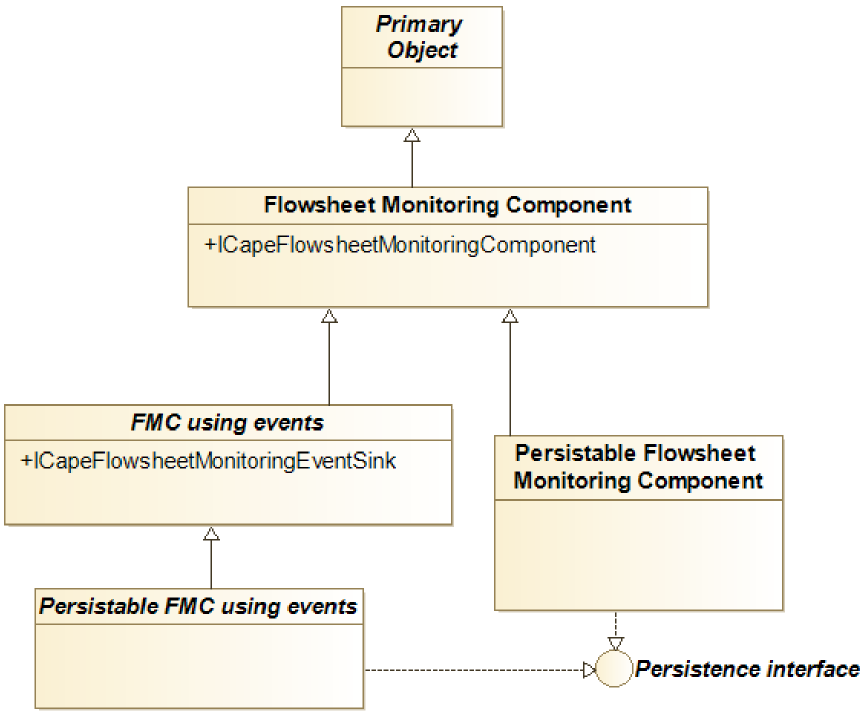

Primary Object
Flowsheet Monitoring Component
+ICapeFlowsheetMonitoringComponent
FMC using events
+ICapeFlowsheetMonitoringEventSink
Persistable Flowsheet Monitoring Component
Persistable FMC using events
Persistence interface

Figure 3: Class diagram of the Flowsheet Monitoring Component

Whereas Unit Operations are generally present in the Flowsheet, it is envisioned that Flowsheet Monitoring Components are not. Depending on the simulation environment application front-end, the PME can make the Flowsheet Monitoring Component available for example from a Plug-in or Add-in menu.

Multiple Flowsheet Monitoring Components can be present in a single Flowsheet.

2.4 Use Cases

This section and the following present the requirements in a more formal way using UML models: Use Cases, Use Case Maps and Sequence Diagrams.

2.4.1 Use Case Priorities

- **High**. Essential functionality for a Flowsheet Monitoring Component. Functionality without which the operation usability or performance of a Flowsheet Monitoring Component might be seriously compromised

- **Medium**. Very desirable functionality that will make Flowsheet Monitoring Components more usable, transportable or versatile. The essence of a Flowsheet Monitoring Component is not compromised by this Use Case, although the usability and acceptance of the component can be.

- **Low**. Desirable functionality that will improve the performance of Flowsheet Monitoring Components. If this Use Case is not met, usability or acceptance can decrease.

2.4.2 Actors

- **PME**. Process Modeling Environment, i.e. the simulation application

- **Flowsheet Builder**. The person who sets up the Flowsheet, the structure of the Flowsheet, chooses thermodynamic models and unit operation models that are in the Flowsheet. This person hands over a working Flowsheet to the Flowsheet User. The Flowsheet Builder can act as a Flowsheet User.

- **Flowsheet User**. The person who uses an existing Flowsheet. This person will put new data into the Flowsheet, rather than change the structure of the Flowsheet.

- **Flowsheet Monitoring Component**. Software component (PMC) inserted in the Flowsheet to perform operations that require information on some or all Flowsheet elements.

- **Flowsheet Monitoring Component Manager**. The part of a PME that provides a list of available Flowsheet Monitoring Components, allows the instantiation of a Flowsheet Monitoring Component and enables it to be maintained and activated.

2.4.3 List of Use Cases

UC-001 Select a Flowsheet Monitoring Component

UC-002 Create a Flowsheet Monitoring Component

UC-003 Restore a Flowsheet Monitoring Component

UC-004 Initialize a Flowsheet Monitoring Component

UC-005 Get Streams of the Flowsheet

UC-006 Get Unit Operations of the Flowsheet

UC-007 Determine Validation Status of the Flowsheet

UC-008 Determine Flowsheet Solution Status

UC-009 Validate the Flowsheet Monitoring Component

UC-010 Monitor Flowsheet

UC-011 Show Graphical User Interface

UC-012: Save A Flowsheet Monitoring Component

UC-013 Get Supported Events

UC-014 Register for Events

UC-015 Unregister for Events

UC-016 Fire Event

UC-017 Handle Event

UC-018 Delete Flowsheet Monitoring Component

2.4.4 Use Cases Map

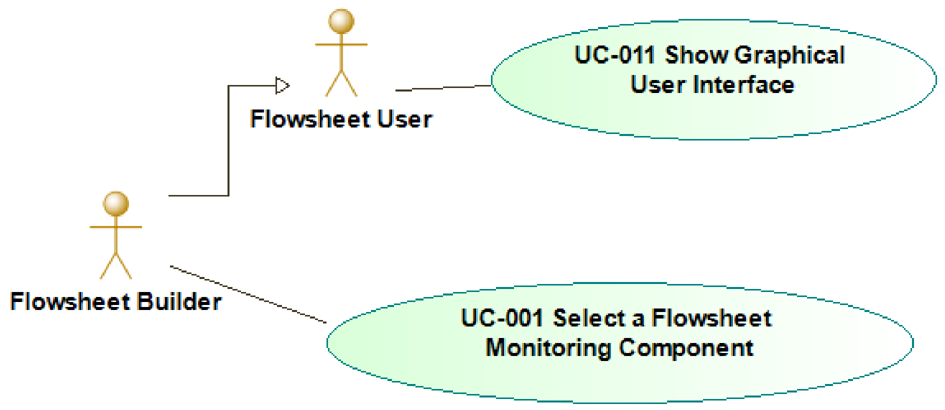

Flowsheet User
UC-011 Show Graphical User Interface
Flowsheet Builder
UC-001 Select a Flowsheet Monitoring Component

Figure 4. Use Cases with Human Actors

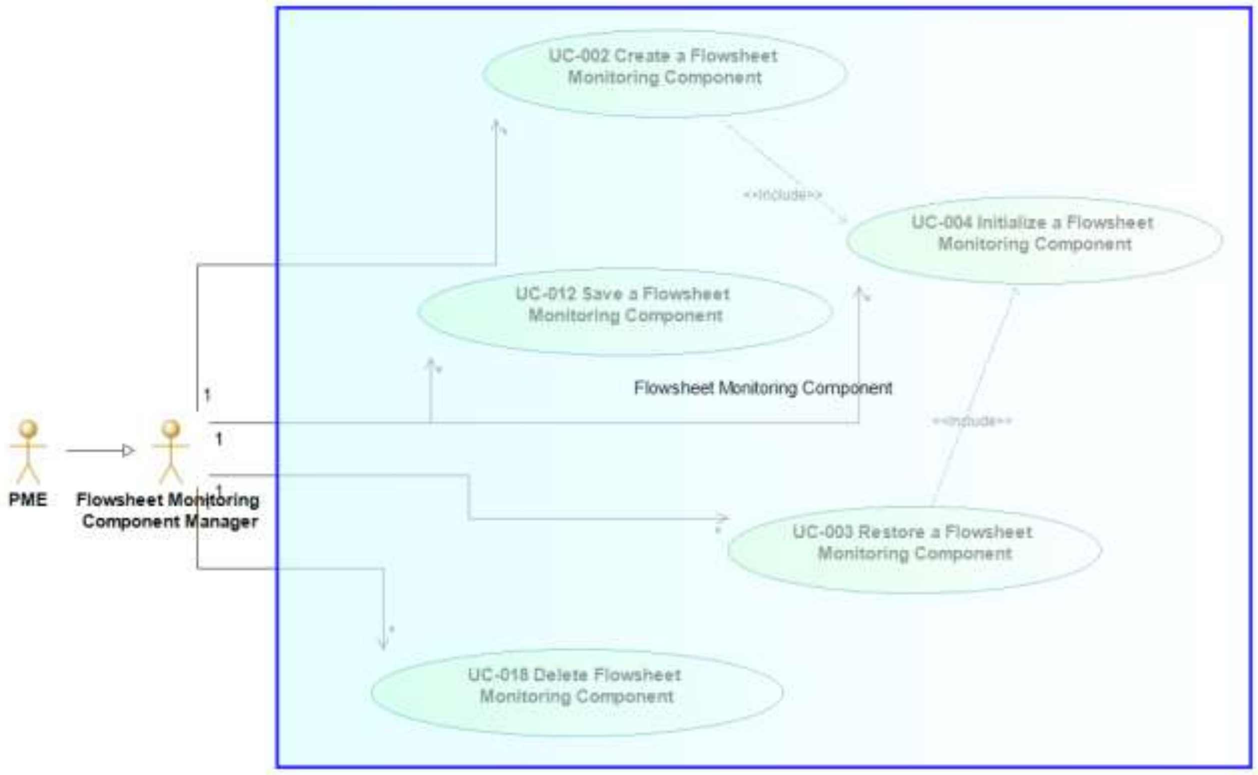

UC-002 Create a Flowsheet Monitoring Component
<<include>>
UC-004 Initialize a Flowsheet Monitoring Component
UC-012 Save a Flowsheet Monitoring Component
Flowsheet Monitoring Component
<<include>>
1
1
1
PME
Flowsheet Monitoring Component Manager
UC-003 Restore a Flowsheet Monitoring Component
UC-018 Delete Flowsheet Monitoring Component

Figure 5. Use Cases with Flowsheet Monitoring Component Manager as Actor

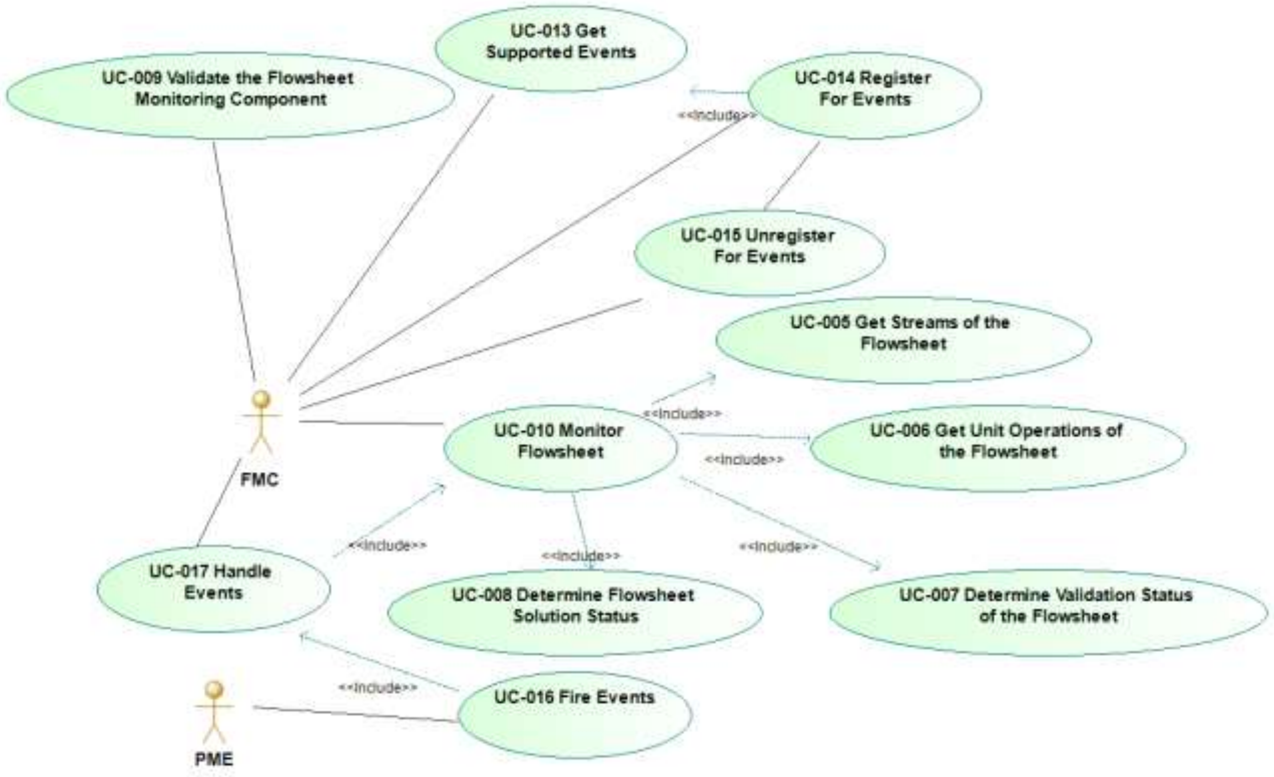

UC-013 Get Supported Events
UC-009 Validate the Flowsheet Monitoring Component
UC-014 Register For Events
<<include>>
UC-015 Unregister For Events
UC-005 Get Streams of the Flowsheet
<<include>>
UC-010 Monitor Flowsheet
UC-006 Get Unit Operations of the Flowsheet
<<include>>
FMC
<<include>>
<<include>>
<<include>>
UC-017 Handle Events
UC-008 Determine Flowsheet Solution Status
UC-007 Determine Validation Status of the Flowsheet
<<include>>
UC-016 Fire Events
PME

Figure 6. Use Cases with PME and Flowsheet Monitoring Component as Actors

2.4.5 Use Cases

UC-001: SELECT A FLOWSHEET MONITORING COMPONENT

Actor: <**Flowsheet Builder**>

Priority: <**High**>

Status: <This requirement is fulfilled by mechanisms provided by software technology.>

Pre-conditions: <There is at least one registered Flowsheet Monitoring Component on the system.>

Flow of events:

The Flowsheet Builder asks the Flowsheet Monitoring Component Manager for the list of available Flowsheet Monitoring Components. The Flowsheet Monitoring Component Manager provides the list to the Flowsheet Builder who selects one from the list.

Post-conditions: <A Flowsheet Monitoring Component has been selected.>

Errors: <None>

Uses: <None>

UC-002: CREATE A FLOWSHEET MONITORING COMPONENT

Actor: <**Flowsheet Monitoring Component Manager**>

Priority: <**High**>

Status: <This requirement is fulfilled by mechanisms provided by software technology and by persistence interfaces.>

Pre-conditions: <The Flowsheet Builder has selected a Flowsheet Monitoring Component using UC-001.>

Flow of events:

The Flowsheet Builder requests the Flowsheet Monitoring Component Manager to create an instance of the selected Flowsheet Monitoring Component. The Flowsheet Monitoring Component Manager creates an instance of the selected Flowsheet Monitoring Component.

If using COM, the Flowsheet Monitoring Component Manager calls the *InitNew* method (if implemented on the selected persistence interface).

The Flowsheet Monitoring Component Manager initializes the Flowsheet Monitoring Component using UC-004.

Post-conditions: <Creation of the Flowsheet Monitoring Component has succeeded.>

Errors: <Creation of the Flowsheet Monitoring Component has failed.>

Uses: <UC-004 Initialize a Flowsheet Monitoring Component>

UC-003: RESTORE A FLOWSHEET MONITORING COMPONENT

Actor: <**Flowsheet Monitoring Component Manager**>

Priority: <**High**>

Status: <This requirement is fulfilled by mechanisms provided by software technology and persistence interfaces.>

Pre-conditions: <The Flowsheet User requests the PME to load a previously stored Flowsheet.>; <All Streams have been loaded.>; <All Units have been loaded and initialized.>; <All stream connections have been restored.>; <The Flowsheet Solution status is available.>

Flow of events:

The Flowsheet Monitoring Component Manager creates an instance of the previously saved Flowsheet Monitoring Component.

The Flowsheet Monitoring Component Manager invokes *Load* on the persistence interface that was earlier used to persist the Flowsheet Monitoring Component.

The Flowsheet Monitoring Component Manager initializes the Flowsheet Monitoring Component using UC-004.

Post-conditions: <Flowsheet Monitoring Component has been successfully restored.>

Errors: <Flowsheet Monitoring Component fails to create.>; < Flowsheet Monitoring Component fails to load.>

Uses: <UC-004 Initialize a Flowsheet Monitoring Component>

UC-004: INITIALIZE A FLOWSHEET MONITORING COMPONENT

Actor: <**Flowsheet Monitoring Component Manager**>

Priority: <**High**>

Status: <This Use Case is only used by UC-002 and UC-003.>; <This requirement is fulfilled by methods of *ICapeUtilities*.>

Pre-conditions: <The Flowsheet Monitoring Component has been created but not yet initialized.>

Flow of events:

Basic path:

The Flowsheet Monitoring Component Manager sets the Simulation Context invoking the *ICapeUtilities::put_SimulationContext* method of the Flowsheet Monitoring Component. The Flowsheet Monitoring Component must have access to the Simulation Context in order to obtain the Stream and Unit Operation Collections.

The Flowsheet Monitoring Component Manager then calls the *ICapeUtilities::Initialize* method of the Flowsheet Monitoring Component. During *Initialize*, the Flowsheet Monitoring Component obtains the *ICapeFlowsheetMonitoring* interface from the Simulation Context.

Optional steps:

The Flowsheet Monitoring Component may obtain the list of events supported by the PME as described in UC-013. The Flowsheet Monitoring Component may register for events as described in UC-014 (Register for Events) and may fail initialization if events required for proper operation are not supported by the PME. Events will not be fired while the Flowsheet Monitoring Component is initialized. The Flowsheet Monitoring Component may execute any monitoring action according to UC-010.

Post-conditions: <Flowsheet Monitoring Component is initialized.>

Errors: <Initialization failed because required events are not supported.>; <Initialization has failed (generic failure).>

Uses: <UC-010 Monitor Flowsheet>; <UC-013 Get Supported Events>; <UC-014 Register for Events>

UC-005: GET STREAMS OF THE FLOWSHEET

Actors: <**Flowsheet Monitoring Component**>

Priority: <**High**>

Status: <This Use Case is fulfilled by using *ICapeFlowsheetMonitoring::GetStreamCollection*.>

Pre-conditions: <The Flowsheet Monitoring Component is processing UC-010.>

Flow of events:

The Flowsheet Monitoring Component invokes *ICapeFlowsheetMonitoring::GetStreamCollection* on the PME, specifying which types of streams (Material Streams, Energy Streams, Information Streams or all Streams) are to be included in the Stream Collection returned by the PME. The PME returns the requested Stream Collection. The Flowsheet Monitoring Component then uses the *ICapeCollection* interface to obtain the number of Streams in the Stream Collection and to access the individual Streams in the Stream Collection.

The Stream Collection obtained from the PME should be considered by the Flowsheet Monitoring Component as a temporary object, and consequently the Stream Collection must not be stored by the Flowsheet Monitoring Component between method calls on the Flowsheet Monitoring Component.

The PME must not change the Stream Collection while the Flowsheet Monitoring Component is executing any method call.

Any change in any Stream Collection can be tracked by the Flowsheet Monitoring Component by handling appropriate events if supported.

Any object in a Stream Collection implements at least the *ICapeIdentification* and *ICapeStream* interfaces. The type of stream determines which additional interfaces are required on the object.

Post-conditions: <Flowsheet Monitoring Component has access to the requested Stream Collection.>

Errors: <An argument is invalid.>

Uses: <None>

UC-006: GET UNIT OPERATIONS OF THE FLOWSHEET

Actor: <**Flowsheet Monitoring Component**>

Priority: <**High**>

Status: <This Use Case is fulfilled by using *ICapeFlowsheetMonitoring::GetUnitOperationCollection*.>

Pre-conditions: <The Flowsheet Monitoring Component is processing UC-010.>

Flow of events:

The Flowsheet Monitoring Component invokes *ICapeFlowsheetMonitoring::GetUnitOperationCollection* on the PME. The PME returns the Unit Operation Collection. The Flowsheet Monitoring Component then uses the *ICapeCollection* interface to obtain the number of Unit Operations in the Unit Operation Collection and to access the individual Unit Operations in the Unit Operation Collection.

The PME must not change the Unit Operation Collection while the Flowsheet Monitoring Component is executing any method call. The Unit Operation Collection obtained from the PME should be considered by the Flowsheet Monitoring Component as a temporary object, and consequently the Unit Operation Collection must not be stored by the Flowsheet Monitoring Component between method calls on the Flowsheet Monitoring Component. Any change in the Unit Operation Collection can be tracked by the Flowsheet Monitoring Component by handling appropriate events if supported.

Post-conditions: <PME has access to the Unit Operation Collection.>

Errors: <None>

Uses: <None>

UC-007: DETERMINE VALIDATION STATUS OF THE FLOWSHEET

Actor: <**Flowsheet Monitoring Component**>

Priority: <**Medium**>

Status: <This Use Case is fulfilled by requesting value of the following property: *ICapeFlowsheetMonitoring::get_ValStatus*.>

Pre-conditions: <The Flowsheet Monitoring Component is processing UC-010.>

Flow of events:

The Flowsheet Monitoring Component requests the value of the Flowsheet validation status from the PME. The PME returns the value of the Flowsheet validation status.

Post-condition: <The Flowsheet validation status is known to the Flowsheet Monitoring Component.>

Errors: <None>

Uses: <None>

UC-008: DETERMINE FLOWSHEET SOLUTION STATUS

Actor: <**Flowsheet Monitoring Component**>

Priority: <**Medium**>

Status: <This Use Case is fulfilled by requesting value of the following property: *ICapeFlowsheetMonitoring::get_SolutionStatus*.>

Pre-conditions: <The Flowsheet Monitoring Component is processing UC-010.>

Flow of events:

The Flowsheet Monitoring Component requests the value of the Flowsheet solution status from the PME. The PME returns the value of the Flowsheet solution status.

Post-conditions: <The Flowsheet solution status is known to the Flowsheet Monitoring Component.>

Errors: <None>

Uses: <None>

UC-009: VALIDATE THE FLOWSHEET MONITORING COMPONENT

Actor: <**Flowsheet Monitoring Component**>

Priority: <**Low**>

Status: <This Use Case is fulfilled by mechanisms internal to the Flowsheet Monitoring Component and methods of *ICapeFlowsheetMonitoringComponent* interface.>

Pre-conditions: <The Flowsheet Monitoring Component has been created (UC-002) or restored from persistence (UC-003).>; <Flowsheet Monitoring Component is requested to validate by the PME.>

Flow of events:

Flowsheet Monitoring Component checks status and consistency of its Public Parameters (if any) and any configuration data. Purpose of validation of the Flowsheet Monitoring Component is not to assert that the Flowsheet is in a state where monitoring can be executed. Flowsheet Monitoring Component returns whether it is valid or not. If not valid, Flowsheet Monitoring Component returns a text describing the reason.

Afterwards the validation status of the FMC, which can be obtained by the PME through *ICapeFlowsheetMonitoringComponent::GetValidationStatus*, cannot be CAPE_NOT_VALIDATED.

Post-conditions: <Flowsheet Monitoring Component has been validated.>; <Validation status is not CAPE_NOT_VALIDATED.>

Errors: <None>

Uses: <None>

UC-010: MONITOR FLOWSHEET

Actor: <**Flowsheet Monitoring Component**>

Priority: <**High**>

Status: <This Use Case may be fulfilled by invoking either the *ICapeFlowsheetMonitoringComponent::Monitor* method or any of the *ICapeFlowsheetMonitoringEventSink* methods.>

Pre-conditions: <The Flowsheet Monitoring Component has been created (UC-002) or restored from persistence (UC-003) or is in the process of being initialized (UC-004).>; <The Flowsheet Monitoring Component registered itself for invocation via events (UC-003) (if needed).>; <Stream Collections reflect the current state of the Flowsheet.>; <Unit Operation Collection reflects the current state of the Flowsheet.>; <Each Material Object exposes the functionality of the underlying thermodynamic sub-system as currently configured.>

Flow of events:

The Flowsheet Monitoring Component performs its monitoring operation, due to one of the following:

❶ *ICapeFlowsheetMonitoringComponent::Monitor* is invoked

❷ An event handler is invoked that requires calculation of the Flowsheet Monitoring Component (UC-017)

❸ Monitoring is required through the Graphical User Interface (UC-011)

❹ The Flowsheet Monitoring Component determines the state of the Flowsheet during initialization (UC-004)

Unless invoked from UC-011, no modal dialog box should be shown by the Flowsheet Monitoring Component. Any result from the monitoring sequence that requires interaction via a modal dialog should be accessed through *ICapeUtilities.Edit* (UC-011).

During monitoring the Flowsheet Monitoring Component updates its internal status and its results which may include output parameters. The Flowsheet Monitoring Component may also update results which may reside outside the Flowsheet Monitoring Component itself (for example file or database) or report execution activity through *ICapeDiagnostic.LogMessage*. During monitoring, the Flowsheet Monitoring Component may access, in a read-only manner, the Stream Collections and Unit Operation Collection of the Flowsheet, the flowsheet solution status and the flowsheet validation status, as outlined in UC-005, UC-006, UC-007 and UC-008.

Post-conditions: <Flowsheet has been monitored.>

Errors: <Invalid Operation>

Uses: <UC-005 Get Streams of the Flowsheet>; <UC-006 Get Unit Operations of the Flowsheet>, <UC-007 Determine Validation Status of the Flowsheet>; <UC-008 Determine Flowsheet Solution Status>

UC-011: SHOW GRAPHICAL USER INTERFACE

Actor: <**Flowsheet User**>

Priority: <**High**>

Status: <This Use Case is fulfilled by invoking the *ICapeUtilities::Edit* method>

Pre-conditions: <The Flowsheet Monitoring Component has been created (UC-002) or restored from persistence (UC-003).>; <Stream Collections reflect the current state of the Flowsheet.>; <Unit Operation Collection reflects the current state of the Flowsheet.>; <Each Material Object exposes the functionality of the underlying thermodynamic sub-system as currently configured.>

Flow of events:

ICapeUtilities::Edit may be invoked for three different reasons: to configure the Flowsheet Monitoring Component, to perform monitoring (UC-010) or to inspect monitoring results. Flowsheet Monitoring Component typically responds to *ICapeUtilities::Edit* by showing a modal dialog to guide the Flowsheet User through any of these tasks.

Post-conditions: <Validation status may be changed to any CapeValidationStatus value.>

Errors: <Edit is not implemented.>

Uses: <UC-010 Monitor Flowsheet>

UC-012: SAVE A FLOWSHEET MONITORING COMPONENT

Actor: <**Flowsheet Monitoring Component Manager**>

Priority: <**High**>

Status: <This Use Case is fulfilled by mechanisms provided by software technology and persistence interfaces [7].>

Pre-conditions: <The Flowsheet Monitoring Component has been created (UC-002) or restored from persistence (UC-003).>

Flow of events:

The Flowsheet Monitoring Component Manager invokes *Save* on its preferred persistence interface of the Flowsheet Monitoring Component. The Flowsheet Monitoring Component saves all its relevant data.

Post-conditions: <Save has succeeded.>

Errors: <The Flowsheet Monitoring Component fails to save.>

Uses: <None>

UC-013 GET SUPPORTED EVENTS

Actor: <**Flowsheet Monitoring Component**>

Priority: <**High**>

Status: <This Use Case is fulfilled by *ICapeFlowsheetMonitoring::GetSupportedEvents*.>

Pre-conditions: <The Flowsheet Monitoring Component has been created (UC-002) or restored from persistence (UC-003) or is in the process of being initialized (UC-004).>; <Simulation Context has been set.>

Flow of events:

The Flowsheet Monitoring Component obtains the *ICapeFlowsheetMonitoring* interface through the Simulation Context. The Flowsheet Monitoring Component, eventually during its initialization step, asks the PME to provide the list of events it supports. The PME returns the list.

Post-conditions: <The Flowsheet Monitoring Component has obtained the list of events supported by the PME.>

Errors: <None>

Uses: <None>

UC-014 REGISTER FOR EVENTS

Actor: <**Flowsheet Monitoring Component**>

Priority: <**Medium**>

Status: <This Use Case is fulfilled by *ICapeFlowsheetMonitoring::RegisterForEvents*.>

Pre-conditions: <The Flowsheet Monitoring Component has been created (UC-002) or restored from persistence (UC-003) or is in the process of being initialized (UC-004).>; <Simulation Context has been set.>

Flow of events:

Basic path

The Flowsheet Monitoring Component obtains the list of supported events using <UC-013>. The Flowsheet Monitoring Component calls *RegisterForEvents* specifying all events it wishes to handle.

Alternative path

The Flowsheet Monitoring Component calls *RegisterForEvents* specifying all the events it requires. If *RegisterForEvents* raises an *ECapeLimitedImpl* error, the Flowsheet Monitoring Component concludes that at least one event required by the Flowsheet Monitoring Component is not supported by the PME.

Post-condition: <The Flowsheet Monitoring Component is registered only for the specified events. Any prior event registration has been cancelled.>

Errors: <A required event is not supported by the PME.>

Uses: <UC-013 Get Supported Events>

UC-015 UNREGISTER FOR EVENTS

Actor: <**Flowsheet Monitoring Component**>

Priority: <**Medium**>

Status: <This Use Case is fulfilled by *ICapeFlowsheetMonitoring*::*RegisterForEvents*.>

Pre-conditions: <The Flowsheet Monitoring Component has been created (UC-002) or restored from persistence (UC-003) or is in the process of being initialized (UC-004).>; <Simulation Context has been set.>; <UC-014 has been used.>

Flow of events:

Using <UC-014> the Flowsheet Monitoring Component registers only those events it wishes to continue to handle. All other event registrations are automatically cancelled.

Post-condition: <Same as UC-014 Register For Events.>

Errors: <None>

Uses: <UC-014 Register For Events>

UC-016 Fire Event

Actor: <**PME**>

Priority: <**Medium**>

Status: <This Use Case is fulfilled by methods internal to the PME and by *ICapeFlowsheetMonitoringEventSink*.>

Pre-conditions: <The PME supports events.>; <An event has taken place.>

Flow of events:

The PME iterates in arbitrary order over all Flowsheet Monitoring Components which have previously registered for this event (UC-014). The PME calls the method of *ICapeFlowsheetMonitoringEventSink* corresponding to this event on each registered Flowsheet Monitoring Component. The Flowsheet Monitoring Component exercises UC-017.

Post-condition: <The event has been processed.>

Errors: <None>

Uses: <UC-017 Handle Event>

UC-017 HANDLE EVENT

Actor: <**Flowsheet Monitoring Component**>

Priority: <**Medium**>

Status: <This Use Case is fulfilled by methods of *ICapeFlowsheetMonitoringEventSink* and by mechanisms internal to the Flowsheet Monitoring Component.>

Pre-conditions: <The Flowsheet Monitoring Component has registered for the event.>; <UC-016 is being exercised.>

Flow of events:

The Flowsheet Monitoring Component performs monitoring tasks as the Flowsheet Monitoring Component sees fit using UC-010.

Post-condition: <The Flowsheet Monitoring Component has handled the event.>; <The Validation status may have changed.>

Errors: <The Flowsheet Monitoring Component tries to register or unregister for events while handling the event.>

Uses: <UC-010 Monitor Flowsheet>

UC-018: DELETE FLOWSHEET MONITORING COMPONENT

Actor: <**Flowsheet Monitoring Component Manager**>

Priority: <**High**>

Status: <This Use Case is fulfilled by using middleware-defined functionality as well as the following method: *ICapeUtilities::Terminate*.>

Pre-conditions: <The Flowsheet Monitoring Component has been created or restored from persistence.>

Flow of events:

The Flowsheet Builder requests that the Flowsheet Monitoring Component Manager deletes the selected Flowsheet Monitoring Component instance. The Flowsheet Monitoring Component Manager calls the *ICapeUtilities::Terminate* method of the Flowsheet Monitoring Component. During Terminate, the Flowsheet Monitoring Component releases all external references including the Simulation Context. Using middleware-defined functionality, the Flowsheet Monitoring Component Manager then releases all references to the Flowsheet Monitoring Component instance. As a result, the Flowsheet Monitoring Component deletes itself.

Post-conditions: <Terminate proceeded successfully.>; <All references to the Flowsheet Monitoring Component have been released and the Flowsheet Monitoring Component has been deleted.>

Errors: <None>

Uses: <None>

3. Analysis and design

3.1 Overview

This section provides component diagrams explaining the interfaces implemented on new software objects introduced by this specification or added to software objects described in other CAPE-OPEN interface specifications. A collaboration diagram is provided to explain the relationship between the PME and a Flowsheet Monitoring Component. Then a State Diagram describes the different possible ways to perform monitoring. Subsequently interface diagrams display the methods supported by each of the new interfaces introduced before giving a detailed description of each method of each of these interfaces.

3.2 Class diagrams

The Flowsheet Monitoring interface specification introduces additional objects in the form of new Collections: the Unit Operation Collection and the Stream Collection. The Flowsheet Monitoring interface specification also introduces additional interfaces on objects previously defined within the UNIT interface specification 4.

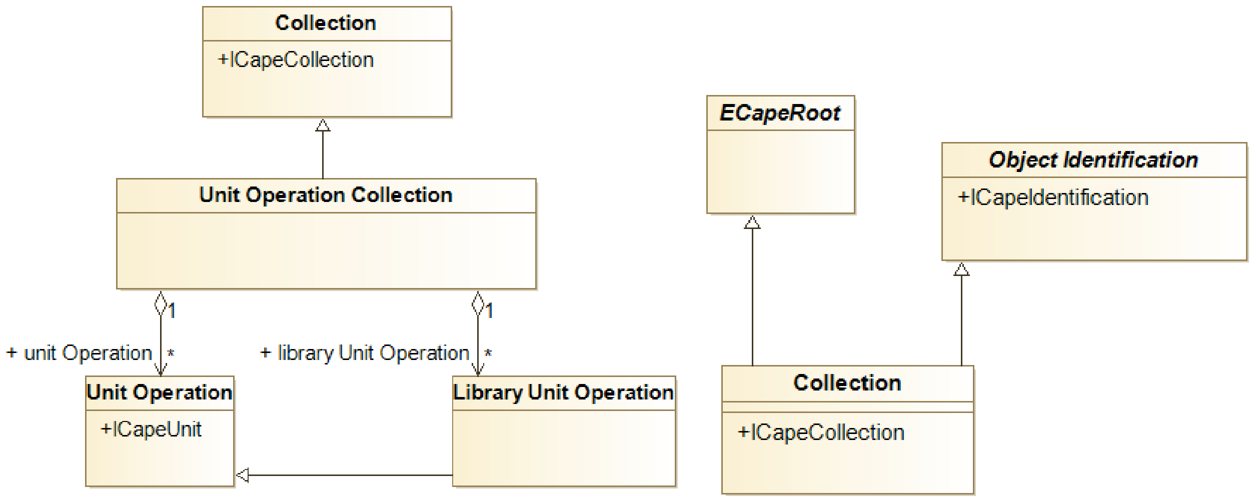


Figure 7 Unit Operation Collection class diagram

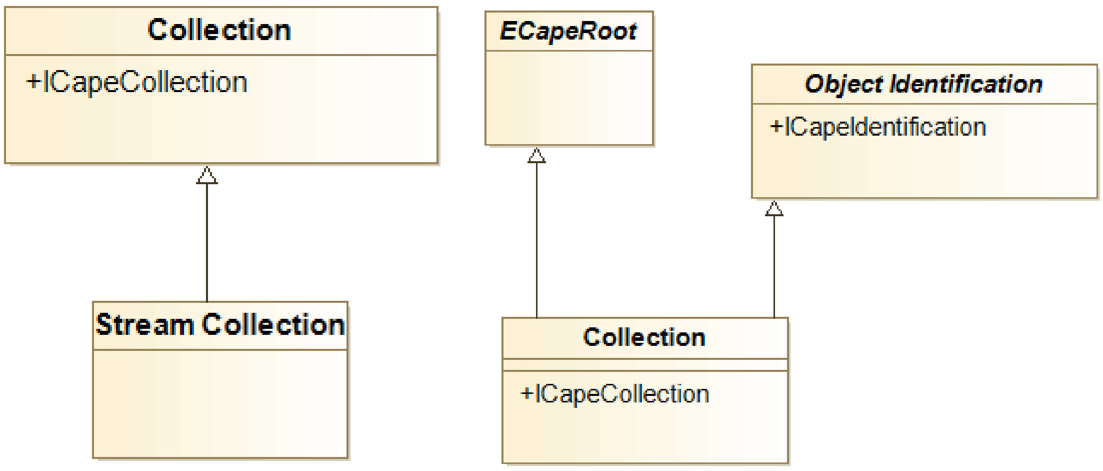


Figure 8 Stream Collection class diagram

The design chosen introduces also modifications to the Material Object, Energy Stream and Information Stream objects through the introduction of the *ICapeStream* interface.

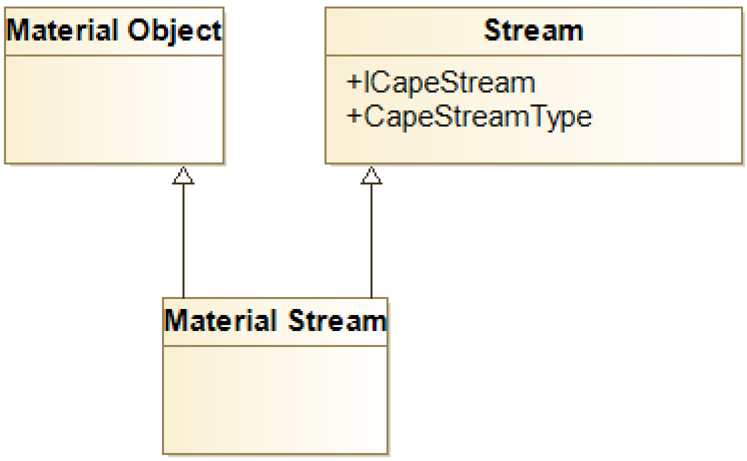


Figure 9 Material Stream class diagram

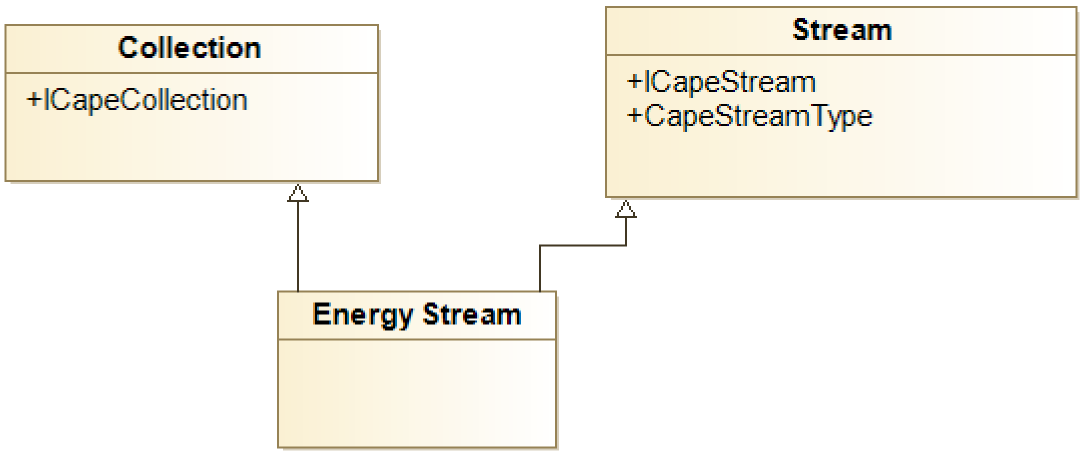


Figure 10 Energy Stream class diagram

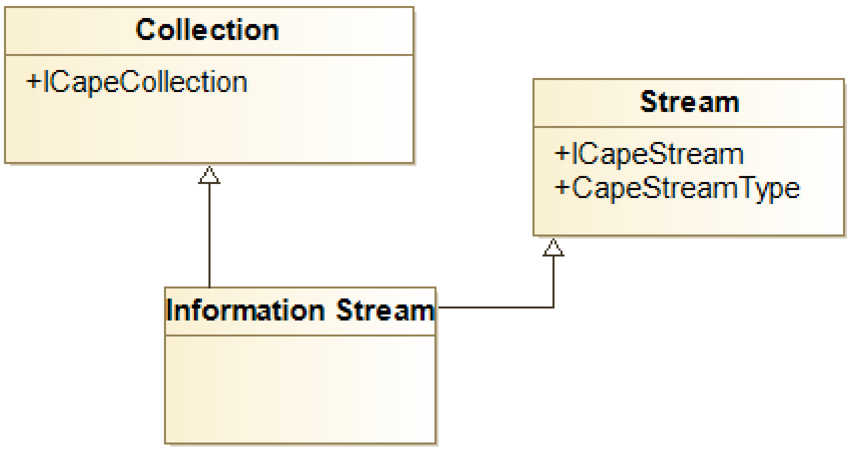


Figure 11 Information Stream class diagram

3.3 Collaboration diagram

A collaboration diagram has been drawn in order to describe the relationships between the different objects. It has a similar objective as Figure 3-9 of the Unit Operation interface specification.

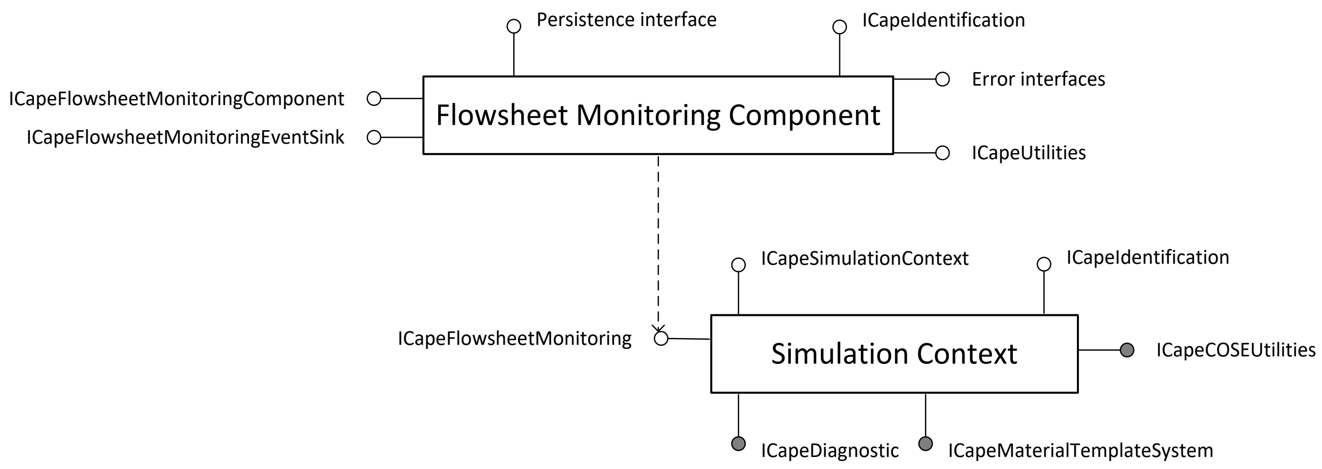


Figure 12 Interfaces implemented by Simulation Context and Flowsheet Monitoring Component

3.4 State diagrams

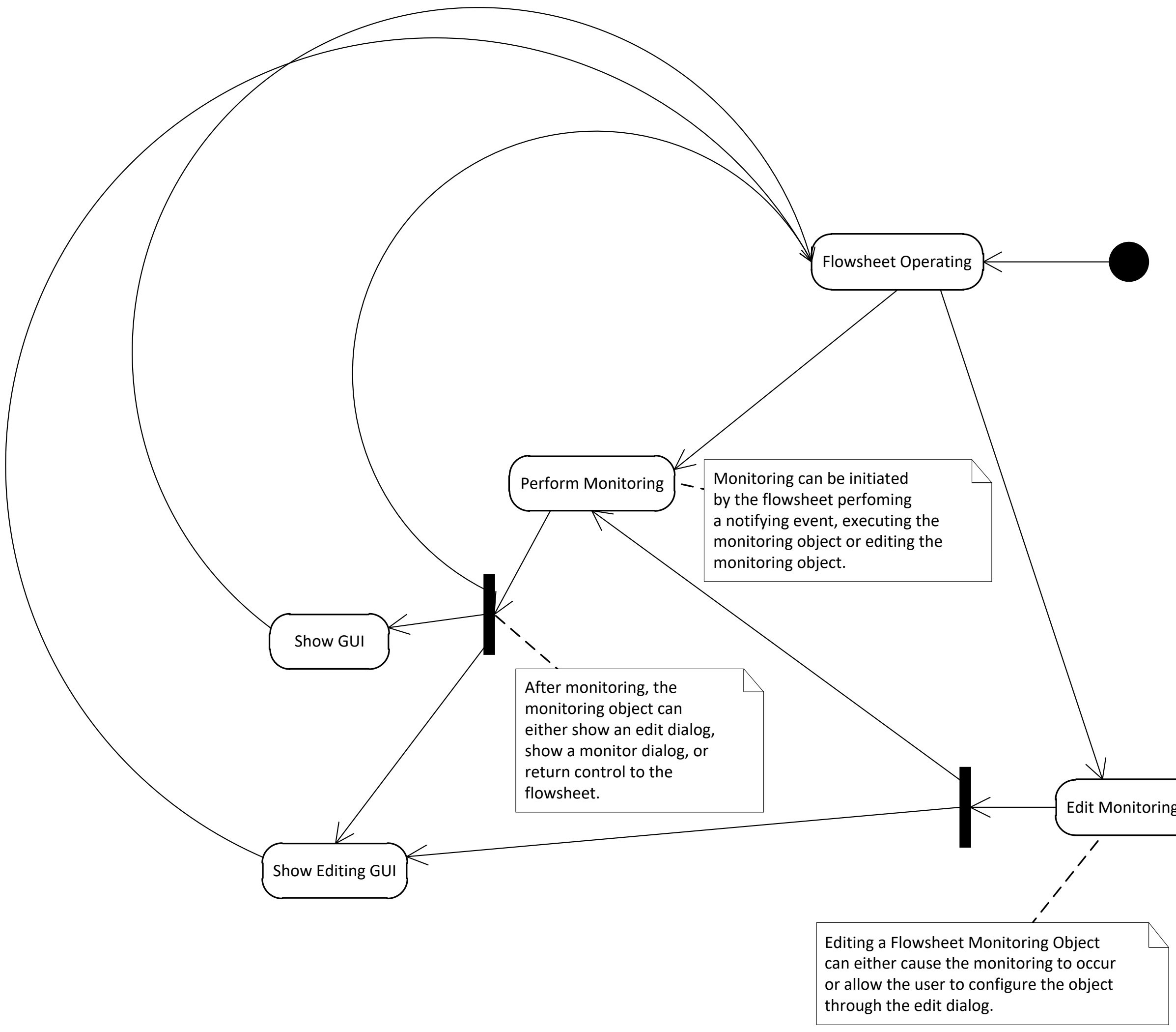
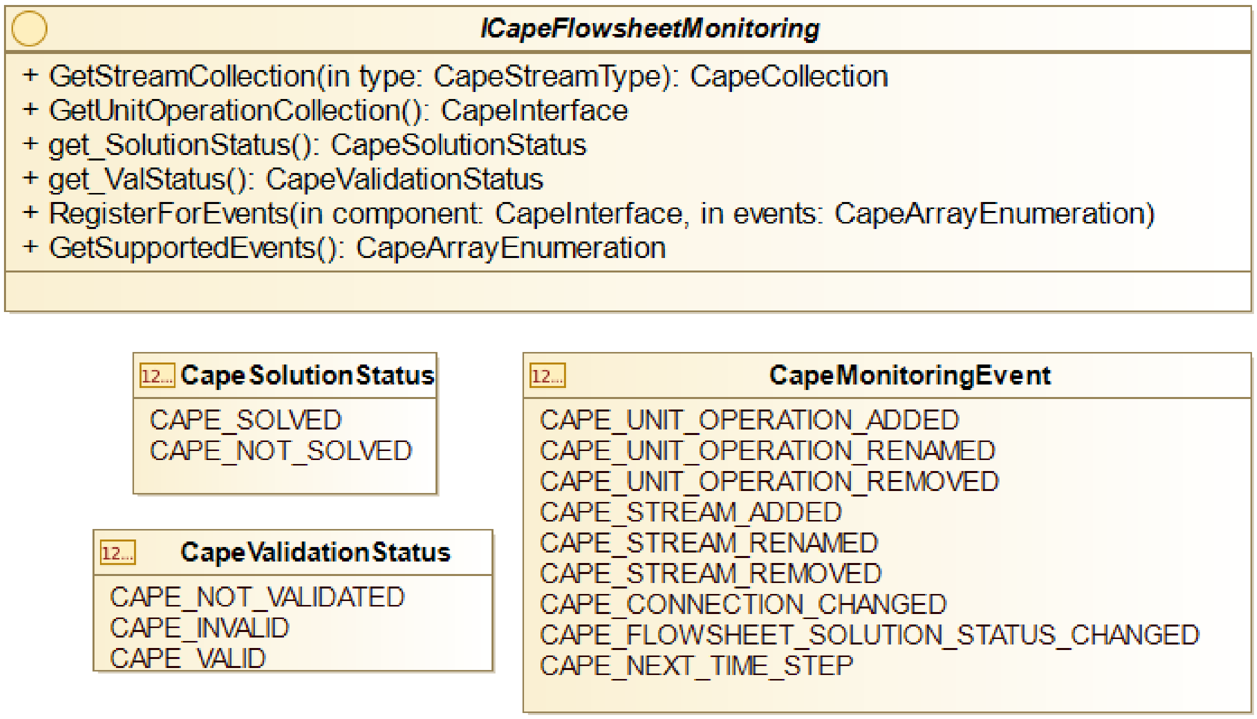

Flowsheet Operating
Perform Monitoring
Monitoring can be initiated by the flowsheet perfoming a notifying event, executing the monitoring object or editing the monitoring object.
Show GUI
After monitoring, the monitoring object can either show an edit dialog, show a monitor dialog, or return control to the flowsheet.
Edit Monitoring Object
Show Editing GUI
Editing a Flowsheet Monitoring Object can either cause the monitoring to occur or allow the user to configure the object through the edit dialog.

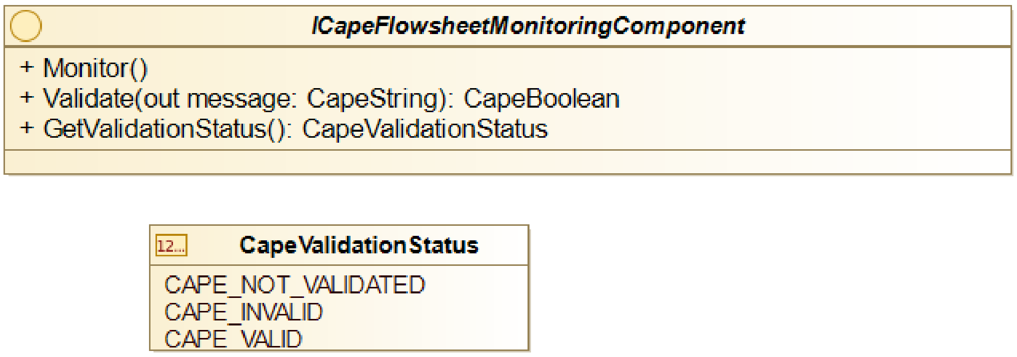
Figure 13. Monitoring Component State Diagram.

3.5 Interface diagrams

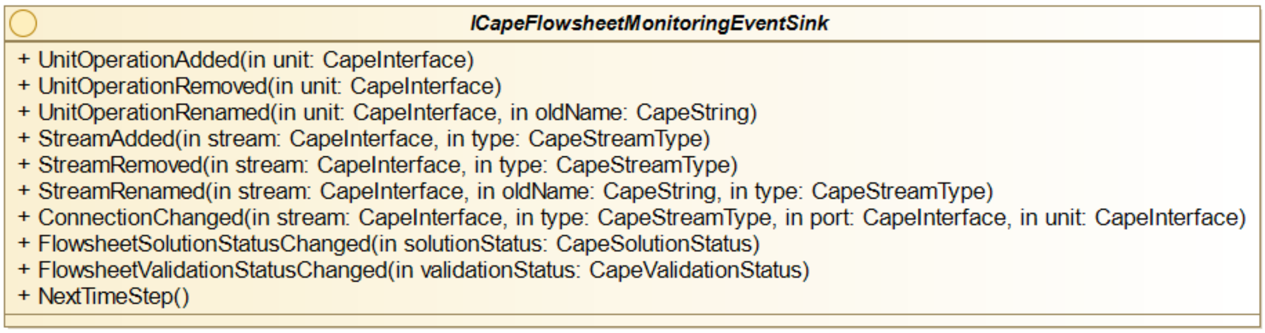
3.5.1 Interface ICapeFlowsheetMonitoring


ICapeFlowsheetMonitoring
+ GetStreamCollection(in type: CapeStreamType): CapeCollection
+ GetUnitOperationCollection(): CapeInterface
+ get_SolutionStatus(): CapeSolutionStatus
+ get_ValStatus(): CapeValidationStatus
+ RegisterForEvents(in component: CapeInterface, in events: CapeArrayEnumeration)
+ GetSupportedEvents(): CapeArrayEnumeration
12...
CapeSolutionStatus
CAPE_SOLVED
CAPE_NOT_SOLVED
12...
CapeValidationStatus
CAPE_NOT_VALIDATED
CAPE_INVALID
CAPE_VALID
12...
CapeMonitoringEvent
CAPE_UNIT_OPERATION_ADDED
CAPE_UNIT_OPERATION_RENAMED
CAPE_UNIT_OPERATION_REMOVED
CAPE_STREAM_ADDED
CAPE_STREAM_RENAMED
CAPE_STREAM_REMOVED
CAPE_CONNECTION_CHANGED
CAPE_FLOWSHEET_SOLUTION_STATUS_CHANGED
CAPE_NEXT_TIME_STEP

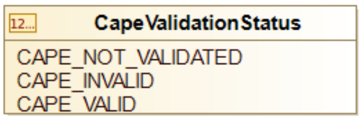
3.5.2 Interface ICapeFlowsheetMonitoringComponent


ICapeFlowsheetMonitoringComponent
+ Monitor()
+ Validate(out message: CapeString): CapeBoolean
+ GetValidationStatus(): CapeValidationStatus
12...
CapeValidationStatus
CAPE_NOT_VALIDATED
CAPE_INVALID
CAPE_VALID

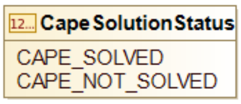
3.5.3 Interface ICapeFlowsheetMonitoringEventSink



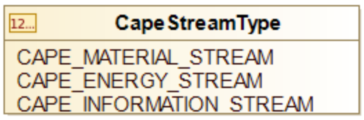
```
ICapeFlowsheetMonitoringEventSink
+ UnitOperationAdded(in unit: CapeInterface)
+ UnitOperationRemoved(in unit: CapeInterface)
+ UnitOperationRenamed(in unit: CapeInterface, in oldName: CapeString)
+ StreamAdded(in stream: CapeInterface, in type: CapeStreamType)
+ StreamRemoved(in stream: CapeInterface, in type: CapeStreamType)
+ StreamRenamed(in stream: CapeInterface, in oldName: CapeString, in type: CapeStreamType)
+ ConnectionChanged(in stream: CapeInterface, in type: CapeStreamType, in port: CapeInterface, in unit: CapeInterface)
+ FlowsheetSolutionStatusChanged(in solutionStatus: CapeSolutionStatus)
+ FlowsheetValidationStatusChanged(in validationStatus: CapeValidationStatus)
+ NextTimeStep()
```



```
CapeValidationStatus
CAPE_NOT_VALIDATED
CAPE_INVALID
CAPE_VALID
```

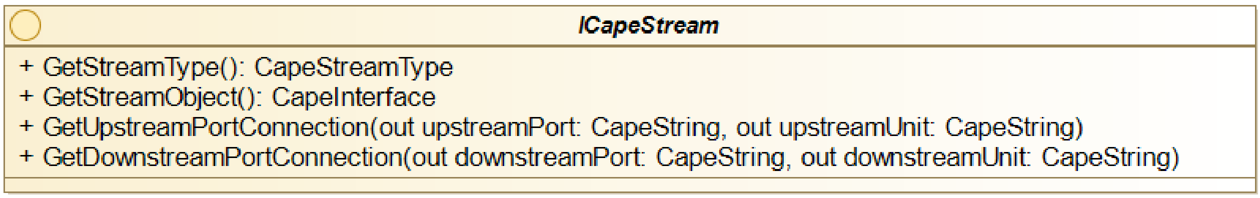


```
CapeSolutionStatus
CAPE_SOLVED
CAPE_NOT_SOLVED
```

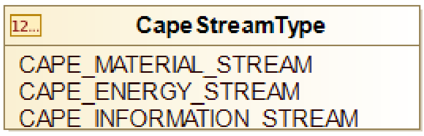


```
CapeStreamType
CAPE_MATERIAL_STREAM
CAPE_ENERGY_STREAM
CAPE_INFORMATION_STREAM
```

3.5.4 Interface ICapeStream



```
ICapeStream
+ GetStreamType(): CapeStreamType
+ GetStreamObject(): CapeInterface
+ GetUpstreamPortConnection(out upstreamPort: CapeString, out upstreamUnit: CapeString)
+ GetDownstreamPortConnection(out downstreamPort: CapeString, out downstreamUnit: CapeString)
```



```
CapeStreamType
CAPE_MATERIAL_STREAM
CAPE_ENERGY_STREAM
CAPE_INFORMATION_STREAM
```

3.6 Interface descriptions

The *ICapeFlowsheetMonitoring* interface is implemented by the PME and obtained through the Simulation Context so that the Flowsheet Monitoring Component can gain access to the Stream Collections and to the Unit Operation Collection in the Flowsheet. The *ICapeFlowsheetMonitoringComponent* interface is implemented by the Flowsheet Monitoring Component and provides methods for invocation and validation. The *ICapeFlowsheetMonitoringEventSink* interface implements event handlers for event driven operation of Flowsheet Monitoring Components.

3.6.1 ICapeFlowsheetMonitoring

Interface Name	ICapeFlowsheetMonitoring
Method Name	GetStreamCollection
Returns	CapeCollection

Description

Returns an *ICapeCollection* that enumerates available Streams of the requested type.

CapeStreamType		
Named Value	**Integer Value**	**Description**
CAPE_ANY_STREAMS	0	Returns a Collection containing all Streams.
CAPE_MATERIAL_STREAM	1	Returns a Collection containing all material Streams.
CAPE_ENERGY_STREAM	2	Returns a Collection containing all energy Streams.
CAPE_INFORMATION_STREAM	3	Returns a Collection containing all information Streams.

Arguments

Name	Type	Description
[in] type	CapeStreamType	Specification of which streams are to be included in the Collection: CAPE_MATERIAL_STREAM for a Collection that contains all Material Streams, CAPE_ENERGY_STREAM for a Collection that contains all Energy Streams, CAPE_INFORMATION_STREAM for a Collection that contains all Information Streams, or CAPE_ANY_STREAM for a Collection containing all Streams.

Errors

ECapeUnknown - No specific errors and no specific meaning of defined errors.

Notes

In case of an empty Stream Collection, the PME returns a collection object containing zero items. A NULL pointer is not a valid return value.

Interface Name	ICapeFlowsheetMonitoring
Method Name	GetUnitOperationCollection
Returns	CapeCollection

Description

Returns an *ICapeCollection* that enumerates all available Unit Operations.

Arguments

None

Errors

ECapeUnknown - No specific errors and no specific meaning of defined errors.

Notes

The objects in the Unit Operation Collection implement at least *ICapeIdentification* and *ICapeUnit*.

In case of an empty Unit Collection, the PME returns a Collection object containing zero items. A NULL pointer is not a valid return value.

Interface Name	ICapeFlowsheetMonitoring
Method Name	get_SolutionStatus
Returns	CapeSolutionStatus

Description

Returns the current solution status of the Flowsheet. Valid values for the solution status are:

CapeSolutionStatus		
Named Value	**Integer Value**	**Description**
CAPE_SOLVED	0	The Flowsheet is solved
CAPE_NOT_SOLVED	1	The Flowsheet is not solved

Arguments

None

Errors

ECapeUnknown - No specific errors and no specific meaning of defined errors.

Notes

Any status that is not CAPE_SOLVED should be considered as CAPE_NOT_SOLVED. For possible future extensions, solution status is kept as an enumeration rather than a Boolean.

Interface Name	ICapeFlowsheetMonitoring
Method Name	get_ValStatus
Returns	CapeValidationStatus

Description

Returns the validation status of the entire Flowsheet.

Arguments

None

Errors

ECapeUnknown - No specific errors and no specific meaning of defined errors.

Notes

Returns CAPE_VALID if the Flowsheet is ready for solving from the PME's point of view, CAPE_NOT_VALID in case the Flowsheet is not ready for solving or CAPE_NOT_VALIDATED in case a validation has not yet been performed.

Interface Name	ICapeFlowsheetMonitoring
Method Name	RegisterForEvents
Returns	--

Description

Subscribes to a specified set of events. The specified events should be among the supported events returned by the *GetSupportedEvents* method.

Arguments

Name	Type	Description
[in] component	CapeInterface	The FMC requesting the registration (the FMC that handles the event).
[in] events	CapeArrayEnumeration	Set of events for which the FMC is registering for. Must be one of the events listed in the Notes. EMPTY list means unregistering.

Errors

ECapeLimitedImpl - should be returned by the PME in case any of the events is not supported.

ECapeBadInvOrder - should be returned by the PME in case the method is called while an event is being handled by the Flowsheet Monitoring Component.

Notes

This method may be called at any time except when any event is being handled by the Flowsheet Monitoring Component. However, it is advised that Flowsheet Monitoring Components register for events during the *ICapeUtilities::Initialize* method so that in case of events that are required by the Flowsheet Monitoring Component but not supported by the PME, the Flowsheet Monitoring Component may raise an initialization failure exception.

The Flowsheet Monitoring Component should only call *RegisterForEvents* if the Flowsheet Monitoring Component implements *ICapeFlowsheetMonitoringEventSink*.

The set of events should be the complete set of events necessary to the Flowsheet Monitoring Component. Calling *RegisterForEvents* cancels previously registered events not in the argument list.

List of possible events

CapeMonitoringEvent	Integer value	ICapeFlowsheetMonitoringEventSink
CAPE_UNIT_OPERATION_ADDED	0	UnitOperationAdded
CAPE_UNIT_OPERATION_RENAMED	1	UnitOperationRenamed
CAPE_UNIT_OPERATION_REMOVED	2	UnitOperationRemoved
CAPE_STREAM_ADDED	3	StreamAdded
CAPE_STREAM_RENAMED	4	StreamRenamed

CAPE_STREAM_REMOVED	5	StreamRemoved
CAPE_CONNECTION_CHANGED	6	ConnectionChanged
CAPE_FLOWSHEET_SOLUTION_STATUS_CHANGED	7	FlowsheetSolutionStatusChanged
CAPE_NEXT_TIME_STEP	8	NextTimeStep

All related events must be explicitly fired, for example, deleting a connected Unit Operation leads to the following sequence of actions:

1. Flowsheet User requests PME to delete a Unit Operation
2. For each connected stream
    - PME disconnects stream from Unit Operation
    - PME fires CAPE_CONNECTION_CHANGED event
3. PME removes the Unit Operation from the Unit Operation Collection
4. PME fires CAPE_UNIT_OPERATION_REMOVED event
    - FMCs release references to the Unit Operation
5. PME invokes *ICapeUtilities::Terminate* on the Unit Operation
    - Unit Operation releases all external references
6. PME deletes Unit Operation
    - PME releases references to the Unit Operation

CAPE_FLOWSHEET_SOLUTION_STATUS_CHANGED event may also be fired at any point of this sequence.

Interface Name	ICapeFlowsheetMonitoring
Method Name	GetSupportedEvents
Returns	CapeArrayEnumeration

Description

Returns the list of events supported by the PME.

Arguments

None

Errors

ECapeUnknown - No specific errors and no specific meaning of defined errors.

Notes

The PME should use the list of events mentioned in the description of the *RegisterForEvents* method.

If the PME does not support any events, the method returns an empty array (VT_EMPTY or an array with zero elements).

3.6.2 ICapeFlowsheetMonitoringComponent

Interface Name	ICapeFlowsheetMonitoringComponent
Method Name	Monitor
Returns	

Description

Monitors the Flowsheet.

Arguments

None

Errors

ECapeUnknown - No specific errors and no specific meaning of defined errors.

ECapeInvalidOperation

Notes

Execution of the Flowsheet Monitoring Component may update its internal status, modify its output parameters but should not lead to a GUI popping up. See 2.2 (Flowsheet Monitoring) for different ways to execute Flowsheet Monitoring Component, one of which being this one.

Interface Name	ICapeFlowsheetMonitoringComponent
Method Name	Validate
Returns	CapeBoolean

Description

Checks whether the Flowsheet Monitoring Component is ready to monitor the Flowsheet.

Arguments

Name	Type	Description
[ACTUALLYout] message	CapeString	A message describing the result of the validation. This is a description of the reason that the validation status was set to CAPE_INVALID.

Notes

The Flowsheet Monitoring Component validation criteria is established by the developer of the Flowsheet Monitoring Component and can include having all Flowsheet Monitoring Component Parameters with CAPE_VALID status. The Flowsheet Monitoring Component developer will provide a description of the validation of the component, and potential causes for the result of the validation process being CAPE_INVALID.

Validation is optional prior to calling the *ICapeFlowsheetMonitoringComponent::Monitor* method. The purpose of validation is to guide the Flowsheet User into proper configuration of the Flowsheet Monitoring Component prior to solving the flowsheet, if solving the Flowsheet Monitoring Component is part of the flowsheet solution.

Following a successful validation of the Flowsheet Monitoring Component, the validation status of the component will either be CAPE_VALID or CAPE_INVALID.

Return value is *true* if the resulting validation status is CAPE_VALID, and *false* if the resulting validation status is CAPE_INVALID.

Errors

ECapeUnknown - No specific errors and no specific meaning of defined errors.

ECapeInvalidOperation – Indicates that the component validation process was not successfully completed. The validation status of the Flowsheet Monitoring Component will be set to CAPE_NOT_VALIDATED.

Interface Name	ICapeFlowsheetMonitoringComponent
Method Name	GetValidationStatus
Returns	CapeValidationStatus

Description

Returns the CapeValidationStatus value of the Flowsheet Monitoring Component.

Arguments

None

Notes

The Flowsheet Monitoring Component validation status may be CAPE_NOT_VALIDATED if:

- Validate has not yet been called
- An input Parameter has changed since the last call to Validate
- *ICapeUtilities::Edit* has been invoked since the last call to Validate.

Errors

ECapeUnknown - No specific errors and no specific meaning of defined errors.

3.6.3 ICapeFlowsheetMonitoringEventSink

Interface Name	ICapeFlowsheetMonitoringEventSink
Method Name	UnitOperationAdded
Returns	-

Description

Raises the event that a Unit Operation has been added into the Flowsheet.

Arguments

Name	Type	Description
[in] unit	CapeInterface	The Unit Operation added to the Flowsheet.

Errors

ECapeUnknown - No specific errors and no specific meaning of defined errors.

Interface Name	ICapeFlowsheetMonitoringEventSink
Method Name	UnitOperationRemoved
Returns	-

Description

Raises the event that a Unit Operation has been removed.

Arguments

Name	Type	Description
[in] unit	CapeInterface	The Unit Operation that has been removed from the Flowsheet. The Unit Operation Collection no longer contains this Unit Operation.

Notes

This event occurs after removal of a Unit Operation from the Flowsheet (after disconnecting streams from the Unit Operation’s ports, but before termination of the Unit Operation).

Errors

ECapeUnknown - No specific errors and no specific meaning of defined errors.

Interface Name	ICapeFlowsheetMonitoringEventSink
Method Name	UnitOperationRenamed
Returns	-

Description

Raises the event that a Unit Operation has been renamed.

Arguments

Name	Type	Description
[in] unit	CapeInterface	The Unit Operation that has been renamed. The Unit Operation's *ICapeIdentification* will provide its new name.
[in] oldName	CapeString	Name of the Unit Operation before it was renamed

Errors

ECapeUnknown - No specific errors and no specific meaning of defined errors.

Interface Name	ICapeFlowsheetMonitoringEventSink
Method Name	StreamAdded
Returns	-

Description

Raises the event that a Stream has been added to the Flowsheet.

Arguments

Name	Type	Description
[in] stream	CapeInterface	The Stream that has been added to the Flowsheet.
[in] type	CapeStreamType	Type of stream: CAPE_MATERIAL, CAPE_ENERGY or CAPE_INFORMATION

Notes

This event fires before connecting the Stream.

Errors

ECapeUnknown - No specific errors and no specific meaning of defined errors.

Interface Name	ICapeFlowsheetMonitoringEventSink
Method Name	StreamRemoved
Returns	-

Description

Raises the event that a Stream has been removed from the Flowsheet.

Arguments

Name	Type	Description
[in] stream	CapeInterface	The Stream that has been removed from the Flowsheet. The Stream Collection no longer contains this Stream.
[in] type	CapeStreamType	Type of stream: CAPE_MATERIAL, CAPE_ENERGY or CAPE_INFORMATION

Notes

This event occurs after removal of a Stream from the Flowsheet (after disconnecting the Stream from the ports of a Unit Operation).

Errors

ECapeUnknown - No specific errors and no specific meaning of defined errors.

Interface Name	ICapeFlowsheetMonitoringEventSink
Method Name	StreamRenamed
Returns	-

Description

Raises the event that a Stream has been renamed.

Arguments

Name	Type	Description
[in] stream	CapeInterface	The Stream that has been renamed. Its *ICapeIdentification* will provide the new name.
[in] oldName	CapeString	Name of the Stream before it was renamed
[in] type	CapeStreamType	Type of stream: CAPE_MATERIAL, CAPE_ENERGY or CAPE_INFORMATION

Errors

ECapeUnknown - No specific errors and no specific meaning of defined errors.

Interface Name	ICapeFlowsheetMonitoringEventSink
Method Name	ConnectionChanged
Returns	-

Description

Raises the event that the Flowsheet has been modified through connection or disconnection of a Stream from a Port.

Arguments

Name	Type	Description
[in] stream	CapeInterface	The Stream that was connected or disconnected.
[in] type	CapeStreamType	Type of Stream: CAPE_MATERIAL, CAPE_ENERGY or CAPE_INFORMATION
[in] port	CapeInterface	Port to which stream was connected or from which Stream was disconnected. *ICapeUnitPort::get_ConnectedObject* can be used to determine whether the event pertains to connecting or disconnecting the Stream.
[in] unit	CapeInterface	Unit Operation to which the Port belongs.

Errors

ECapeUnknown - No specific errors and no specific meaning of defined errors.

Interface Name	ICapeFlowsheetMonitoringEventSink
Method Name	FlowsheetSolutionStatusChanged
Returns	-

Description

Raises the event that the Flowsheet has gone from a solved status to an unsolved status as a result from a change to the Flowsheet, or the Flowsheet has gone from an unsolved status to a solved status.

Arguments

Name	Type	Description
[in] solutionStatus	CapeSolutionStatus	New solution status of the Flowsheet. CAPE_SOLVED in case a solution was reached. See *ICapeFlowsheetMonitoring::get_SolutionStatus* for details.

Errors

ECapeUnknown - No specific errors and no specific meaning of defined errors.

Interface Name	ICapeFlowsheetMonitoringEventSink
Method Name	FlowsheetValidationStateChanged
Returns	-

Description

Raises the event that the validation status of the Flowsheet has changed.

Arguments

Name	Type	Description
[in] validationStatus	CapeValidationStatus	Current validation status of the Flowsheet. CAPE_VALID if the Flowsheet is ready for solving. CAPE_INVALID if the Flowsheet is not ready for solving. CAPE_NOT_VALIDATED if the validation has not taken place since the last configuration change.

Errors

ECapeUnknown - No specific errors and no specific meaning of defined errors.

Interface Name	ICapeFlowsheetMonitoringEventSink
Method Name	NextTimeStep
Returns	-

Description

Raises the event that the time integration of the Flowsheet has progressed to the next time step.

Arguments

None

Notes

This event occurs upon completion of each time step in a dynamic simulation.

Errors

ECapeUnknown - No specific errors and no specific meaning of defined errors.

3.6.4 ICapeStream

Interface Name	ICapeStream
Method Name	GetStreamType
Returns	CapeStreamType

Description

Provides the Stream type among the following types: CAPE_MATERIAL_STREAM, CAPE_ENERGY_STREAM or CAPE_INFORMATION_STREAM.

Arguments

None

Errors

ECapeUnknown - No specific errors and no specific meaning of defined errors.

Notes

Interface Name	ICapeStream
Method Name	GetStreamObject
Returns	CapeInterface

Description

Returns the interface to the underlying Material, Energy or Information Stream object.

Arguments

None

Errors

ECapeUnknown - No specific errors and no specific meaning of defined errors.

Interface Name	ICapeStream
Method Name	GetUpstreamPortConnection
Returns	

Description

Returns names of the upstream Port and Unit.

Arguments

Name	Type	Description
[ActuallyOut] upstreamPort	CapeString	The name of the outlet Port to which the Stream is connected upstream.
[ActuallyOut] upstreamUnit	CapeString	The name of the Unit Operation to which the outlet Port belongs.

Errors

ECapeUnknown - No specific errors and no specific meaning of defined errors.

Notes

Should the Stream not be connected to any upstream outlet Port, the method returns UNDEFINED (i.e. NULL) for both arguments.

Instead of the Port and Unit Operations objects themselves, their names are returned. To get access to the Unit Operation object, use the Unit Operation Collection. The Port object can be obtained from the Port Collection of the Unit Operation.

Interface Name	ICapeStream
Method Name	GetDownstreamPortConnection
Returns	

Description

Returns names of the downstream Port and Unit.

Arguments

Name	Type	Description
[ActuallyOut] downstreamPort	CapeString	The name of the inlet Port to which the Stream is connected downstream.
[ActuallyOut] downstreamUnit	CapeString	The name of the Unit Operation to which the inlet Port belongs.

Errors

ECapeUnknown - No specific errors and no specific meaning of defined errors.

Notes

Should the Stream not be connected to any downstream inlet Port, the method returns UNDEFINED (i.e. NULL) for both arguments.

Instead of the Port and Unit Operations objects themselves, their names are returned. To get access to the Unit Operation objects, use the Unit Operations Collection. The Port object can be obtained from the Port Collection of the Unit Operation.

3.7 Scenarios

3.7.1 Thermodynamic calculations on a specific Stream

One typical scenario for post-processing application is making a series of thermodynamic calculations on a specific Stream. The Flowsheet Monitoring Component accesses the list of Material Objects within the Flowsheet, i.e. the Collection of Material Objects. The Flowsheet User selects a Stream from the Collection of Material Objects. The Flowsheet Monitoring Component retrieves the current conditions on the selected Material Object. The Flowsheet User defines the set of thermodynamic calculations to be run: variation of temperature, pressure, composition, type of calculations. The Flowsheet Monitoring Component duplicates the selected Material Object. The Flowsheet Monitoring Component requests the selected calculations on the Duplicated Material Object and presents the results to the Flowsheet User.

3.7.2 Calculation of physical and chemical exergy of a material stream

The conditions for the Reference Environment are provided as default values within the Flowsheet Monitoring Component or configured by the end-user in the Flowsheet Monitoring Component.

The Flowsheet Monitoring Component calculates the physical exergy and the chemical exergy of any material stream. For the physical exergy, its calculation relies on enthalpy and entropy, two thermodynamic properties that the Flowsheet Monitoring Component may obtain the calculation from by duplicating the Material Object and requesting such calculations. This scenario calls for configuration of the Flowsheet Monitoring Component regarding the reference environment, which is defined through temperature, pressure but also composition. The composition of the reference environment is typically one of air surrounding the process.

3.7.3 Archival of multiple runs of a Flowsheet

An end-user conducts a series of simulation runs on the same Flowsheet, changing each time one or several values of parameters or feed conditions. The FMC decides on a database scheme and data storage methodology. At the end of each run, when the end-user considers it proper for archiving, the end-user asks the Flowsheet Monitoring Component to save the results obtained. The Flowsheet Monitoring Component collects the information from each Stream in the Flowsheet so that the state of each stream is completely known. The Flowsheet Monitoring Component collects the information on each Unit Operation by retrieving values of each of its Public Parameter. The FMC populates its database with the information collected at each run. The Flowsheet Monitoring Component is managing a database of the runs performed and an analysis of these runs can be exercised by the Flowsheet Monitoring Component at the end-user request.

3.7.4 Connectivity analysis

One typical scenario for post-processing applications is the requirement to analyze the flowsheet connectivity (which streams connect to which unit operations). Connectivity can be analyzed by using the *GetUpstreamPortConnection* and *GetDownstreamPortConnection* methods on the *ICapeStream* interface in combination with the connected objects and the directions of Unit Operation Ports.

Streams that do not originate from a Unit Operation should be considered as feed Streams to the entire Flowsheet. Streams that do not go to a Unit Operation should be considered as product Streams of the entire flowsheet. Some simulation environments use feed and product Unit Operations. This can be accounted for in the above analysis when iterating over Unit Operations. If a Unit Operation has a single Material Port, it must be a feed or product Unit Operation. If it is an inlet Port, the Unit Operation is a product Unit Operation and the Stream connected to it must be considered as a product of the entire Flowsheet. If it is an outlet Port, the Unit Operation is a feed Unit Operation and the Stream connected to it must be considered as a feed to the entire Flowsheet.

4. Notes on the interface specifications

4.1 Versioning

Experiences with interoperability testing revealed a shortcoming of the current mechanism used to communicate which version of the CAPE-OPEN interfaces (IDL/type library) an object supports. For example, there is no way for a process modeling environment (PME) to know which versions of the thermodynamic specification that a process modeling component (PMC) such as a Flowsheet Monitoring Component supports.

Currently, the version of CAPE-OPEN supported by a PMC is indicated through the CapeVersion registry key in Microsoft COM implementations. While this key provides general information about the overall CAPE-OPEN support of an object, it does not provide the granularity needed to indicate support for multiple versions of CAPE-OPEN, as in the case presented by the release of Thermodynamics version 1.1. As a result, a new versioning scheme needs to be developed to provide information regarding which versions of individual standards an object supports.

The 'Implemented Categories' registry key is a mechanism to provide more granular information about version support to the PME and other objects. Currently, 'Implemented Categories' is used primarily to distinguish between types of PMCs; for instance, a Flowsheet Monitoring Component will include Category FMC GUID (CATID) of {7BA1AF89-B2E4-493d-BD80-2970BF4CBE99}. This is sufficient to inform the PME that the FMC supports CAPE-OPEN version 1.0 Flowsheet Monitoring specifications. However, this does not provide the PME, or other objects, with any information regarding whether this Flowsheet Monitoring Component is capable of supporting Thermodynamics version 1.1 or a later version. Using additional CATID values, as described below, will enable the object to provide more granular version information. For example, exposing the proper CATID combinations will enable a Flowsheet Monitoring Component to clearly indicate that it only supports version 1.1 of the CAPE-OPEN Thermodynamics standards.

It is useful for the PME to know whether a Flowsheet Monitoring Component needs access to the thermodynamic subsystem. In order to provide this information, the Flowsheet Monitoring Component should be registered with the following CATID:

Consumes_Thermo_CATID:	{4150C28A-EE06-403f-A871-87AFEC38A249}

Presence of Consumes_Thermo_CATID informs the PME that the Flowsheet Monitoring Component will require thermodynamic interfaces specified further by the CATIDs listed below. In the event that the Flowsheet Monitoring Component does not indicate that it consumes thermodynamics, the Flowsheet Monitoring Component either will not require access to the thermodynamic subsystem. Flowsheet Monitoring Components that consume thermodynamics will use the following CATID to indicate that version 1.1 of the thermodynamic interface is supported:

SupportsThermodynamics11_CATID:	{4667023A-5A8E-4CCA-AB6D-9D78C5112FED}

For future versions of the thermodynamic interfaces, CATIDs will be defined as these become available.

Based upon which registry key is present, the PME can choose which version of the thermodynamic interfaces is most appropriate to interact with the Flowsheet Monitoring Component.

Following the same logic, it is advantageous for the PME to know which version of the UNIT interface specification is used by the Flowsheet Monitoring Component, if any. In case the Unit Operation is accessed by the Flowsheet Monitoring Component, the Flowsheet Monitoring Component should be registered with the following CATID:

Monitors_UNIT_CATID:	{C049C4FC-FB57-4865-A263-C68815D99079}

Flowsheet Monitoring Components that monitor Unit Operations will use the following CATID to indicate which version of the UNIT interfaces is supported:

SupportsUNIT10_CATID:	{42A06BA3-03EA-42CD-8609-30A7D2186445}

This scheme will allow support for future versions of CAPE-OPEN Thermodynamic and UNIT interfaces.

5. Prototypes implementation

The information in this section will be accurate once the interface specification is considered final.

For testing implementations of Flowsheet Monitoring Components, a prototype socket is implemented in COCO Simulator[14]; this implementation does not (at point of this writing) include support for event driven activation of Flowsheet Monitoring Components. It has been tested to work with below mentioned implementations of plugs.

For testing implementations of Flowsheet Monitoring Support in simulation environments, two plugs are available:

- US-EPA’s WAR implementation[15]
- TERNYP (ternary phase envelope and property plotting utility), included in the COCO[14] installation

Neither of these implementations require or use event driven operation.

6. Specific glossary terms

Event: an action or occurrence originating from an action or change in the flowsheet. The firing of the event triggers handling of the event by software components listening to the event.

Flowsheet: a model of a process made within the PME using Unit Operations connected by Streams. A Flowsheet contains thermodynamic property models shared between Streams and Unit Operations.

Flowsheet Monitoring: The process of performing (post-processing) calculations by using information from one or more flowsheet elements (such as Streams and Unit Operations).

Parameter: in- or output information to a PMC, containing of a name, description, data type, a value and meta information about the value (such as default value, data limits, dimensionality and list of possible options)

PMC: Process Modeling Client. CAPE-OPEN plug implementation, e.g. Unit Operation, Thermodynamic Property Package, Flowsheet Monitoring Component, …

PME: Process Modeling Environment. CAPE-OPEN enabled simulation environment, e.g. Flowsheeting application.

Port: Connection point defined by Unit Operation for connecting a Stream

Simulation Context: object exposed by the PME and passed to PMCs, representing the environment (or application) that the PMC is currently embedded in.

Stream: container for information that flows between Ports of Unit Operations. Material streams contain matter, Energy Streams describe energy transfer, Information Streams can contain any data in the form of parameters. Material Streams implement a CAPE-OPEN Material Object, Information and Energy Streams implement a CAPE-OPEN Collection of CAPE-OPEN Parameters.

Unit Operation: PMC model representation of simulated physical equipment in a process.

7. Bibliography

[1] Young, D., Scharp, R. and Cabezas, H. (2000), “The waste reduction (WAR) algorithm: environmental impacts, energy consumption, and engineering economics”, Waste Management, Vol 20 no 8 pp 605-615.

[2] Ahmad K. Hilaly & Subhas K. Sikdar (1994) Pollution Balance: A New Methodology for Minimizing Waste Production in Manufacturing Processes, Air & Waste, 44:11, 1303-1308, DOI: 10.1080/10473289.1994.10467325.

[3] Linnhoff B. *et al* (1982), "User Guide on Process Integration for the Efficient Use of Energy", IChemE, Rugby, U.K.

[4] Unit Operations Interfaces, available from http://www.colan.org/index-33.html

[5] Utilities Common Interface, available from http://www.colan.org/index-35.html

[6] Identification Common Interface, available from http://www.colan.org/index-35.html

[7] Persistence Common Interface, available from http://www.colan.org/specifications/persistence-common-interface-specification-2/

[8] Parameter Common Interface, available from http://www.colan.org/index-35.html

[9] Simulation Context Interfaces, available from http://www.colan.org/index-34.html

[10] Thermodynamics and Physical Properties Interfaces, version 1.0, available from http://www.colan.org/index-33.html

[11] Collection Common Interfaces, available from http://www.colan.org/index-35.html

[12] Thermodynamics and Physical Properties Interfaces, version 1.1, available from http://www.colan.org/index-37.html

[13] Methods & Tools Integrated Guidelines, available from http://www.colan.org/index-32.html

[14] COCO, the CAPE-OPEN to CAPE-OPEN Simulator. URL: http://www.cocosimulator.org/

[15] US-EPA URL: http://www.epa.gov/nrmrl/std/mtb/p2tools/cape.htm

[16] Fermeglia, M., Longo, G., Toma, L., *COWAR: A CAPE OPEN Software Module for the Evaluation of Process Sustainability*. Environmental Progress, Vol.27, No.3, p 373 (2008).

[17] Barrett, W.M., van Baten, J.M. and Martin, T., *Implementation of the waste reduction (WAR) algorithm utilizing flowsheet monitoring*, Computers and Chemical Engineering 35 (2011) 2680-2686.