

CAPE-OPEN

Expanding Process Modelling
Capability through Software
Interoperability Standards

Cape Open Interface Specification: Manager



www.colan.org

ARCHIVAL INFORMATION

Title	Manager
Owner	M&T SIG
Location	.CommonSource_SpecificationExample.xml
Document Unique Identifier	608E43E4-81E4-4B81-AAD1-1EC38F17C6EF
Distribution	Public
Status	Approved
Document Version Number	1
Created Date	2024-11-06
Revision Date	2024-12-12

Version History

Document Version Number	RFC Date	Release Date	Comments
0.05	2024-11-06	-	Created by SIG
0.08	2024-11-12	-	Added a section 2.4
0.1	2024-12-01	-	Added a section 3
1	2024-12-12	-	Resolved Comments

IMPORTANT NOTICES

COPYRIGHT NOTICE

Copyright 2024 CAPE-OPEN Laboratories Network (CO-LaN).

PERMISSION NOTICE

Subject to all of the terms and conditions below, the owner of the copyright hereby grants you a fully-paid up, non-exclusive, non-transferable, perpetual, worldwide license (without the right to sublicense), to use this document to create and distribute software and to use, copy, and distribute this document provided that: (1) both the copyright notice identified above and this permission notice appear on any copies of this document; (2) the document will not be otherwise resold or transferred for commercial purposes; and (3) no modifications are made to this document. This limited permission automatically terminates without notice if you breach any of these terms or conditions.

THIS DOCUMENT IS PROVIDED "AS IS," AND CO-LAN MAKES NO REPRESENTATIONS OR WARRANTIES, EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, NON-INFRINGEMENT, OR TITLE; THAT THE CONTENTS OF THE DOCUMENT ARE SUITABLE FOR ANY PURPOSE; NOR THAT THE IMPLEMENTATION OF SUCH CONTENTS WILL NOT INFRINGE ANY THIRD-PARTY PATENTS, COPYRIGHTS, TRADEMARKS OR OTHER RIGHTS.

CO-LAN WILL NOT BE LIABLE FOR ANY DIRECT, INDIRECT, SPECIAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF ANY USE OF THE DOCUMENT OR THE PERFORMANCE OR IMPLEMENTATION OF THE CONTENTS THEREOF.

The name and trademarks of CO-LaN may NOT be used in advertising or publicity pertaining to this document or its contents without specific, written prior permission obtained from CO-LaN. Title to copyright in this document will always remain with CO-LaN.

Trademark Usage

Many of the designations used by manufacturers and seller to distinguish their products are claimed as trademarks. Where those designations appear in CO-LaN publications, and the authors are aware of a trademark claim, the designations have been printed in caps or initial caps.

SUMMARY

This document contains the textual CAPE-OPEN specification of the CAPE-OPEN Primary Process Modelling Component (PMC) called Manager that creates and manages a set of Primary PMC objects belonging all to the same category of Primary PMCs. Examples of Managers are Property Package Managers (i) and Chemical Reaction Package Managers (ii). The concept is extendable to Managers of other Primary PMC objects such as Unit Operations (iii).

CONTENTS

1. DOCUMENT ROADMAP.....	6
2. AUDIENCE	7
3. INTRODUCTION	8
4. DESIGN.....	9
5. REQUIREMENTS	11
6. USE CASES	14
7. CONCEPTUAL OBJECT MODEL	19
7.1 MANAGER.....	19
8. ANALYSIS AND DESIGN	20
8.1 INTERFACE DESCRIPTIONS.....	20

1. Document Roadmap

This document is a textual CAPE-OPEN Interface Specification Document of the CAPE-OPEN Primary Process Modelling Component (PMC) called Manager that creates and manages a set of Primary PMC Objects (vii) belonging all to the same category of Primary PMCs. A Manager allows creation of a Primary PMC Object by reloading a previously saved state, by selecting from a list of pre-configured Templates, or from scratch as a new unconfigured Primary PMC Object.

The document belongs to the documentation set of CAPE-OPEN interface specifications in its version 1.1. This imposes that for software components implementing the interface defined in this document, the CapeV-version registration value must be set to “1.1”.

2. Audience

This document is intended primarily for software developers who want to create CAPE-OPEN Managers.

For any reader, an understanding of UML diagrams is assumed.

This document is not intended for Process Engineers who are end-users of CAPE-OPEN process simulation components or process environments.

3. Introduction

The interface specification of a Manager is a generalization of the specification of a Property Package Manager (i). However, the Property Package Manager specification has some shortcomings, in particular for the configuration of new Property Packages and the restoration of Property Packages from persistence.

This document introduces the concept of a Manager that is not specific to Property Packages, while addressing the above shortcomings.

This document introduces the *ICapeManager* interface.

4. Design

Any Primary PMC Object is characterized by a component type. The characterization is made through a pre-defined Category ID for stand-alone Primary PMC Objects or through the Category ID of the Manager that creates the Primary PMC Object. For example, there is a pre-defined Category ID of a stand-alone Property Package, and there is a pre-defined Category ID for a Property Package Manager.

Some PMCs are custom-made for one scenario and have only one configuration. In that case these PMCs will be made stand-alone PMCs by their developer/vendor. In other cases, a more generic software component allows for customization of PMC objects by the end-user. For example, a Property Package Manager allows for configuration of a Property Package (e.g., selection of compounds, of thermodynamic models, of phases supported). Instead of requiring each of the configurations to be registered as a stand-alone software component, a Manager is the single software component to be registered, and visible to the end-user as such.

Each Manager software component type has a stand-alone counterpart. For example, a Manager for Chemical Reaction Packages (ii), a.k.a. a Chemical Reaction Package Manager, finds its counterpart in a stand-alone Chemical Reaction Package, which is not managed by a Manager. This document does not describe how any stand-alone PMC Object behaves.

A Manager is responsible for instantiating Primary PMC Objects. A Manager maintains a set of Templates, each containing the configuration of the instance of a Primary PMC Object. There are three different scenarios leading to the instantiation of a Primary PMC Object. A PME can request a Manager to create Primary PMC Objects as follows: 1. an un-configured Primary PMC Object for subsequent de-persistence by the PME, 2. from a list of pre-configured Templates, 3. a new PMC Object to be configured on-the-fly by the Flowsheet Builder. In all circumstances, the remaining lifespan of a Primary PMC Object created by a Manager is like for any Primary PMC Object, starting with *ICapeUtilities::Initialize* and ending with *ICapeUtilities::Terminate*.

A Manager is a Primary PMC Object. Primary PMC Objects may implement persistence (vi). However, a Manager should not implement persistence and a PME must not persist a Manager. The configuration of a Manager consists of a list of Templates and the configuration of each Template. Indeed, the configuration of the Manager is typically defined at the machine or current user level, not at the flowsheet level.

The visibility of a Template is not restricted to a particular session of the Manager or to a particular PME. Once a Template is created, it is visible in all PMEs through the Manager until the Manager deletes the Template. The way Templates are stored by the Manager is internal to the Manager.

To ensure portability over time and between machines, there is a need for a straightforward workflow that requires persistence to be implemented on Primary PMC Objects created through a Manager. Therefore, Primary PMC Objects created by the Manager must implement persistence.

The workflow for restoring a Primary PMC is distinctly different from the workflow for creating a new Primary PMC that is not based on a Template. In contrast to functionality offered by *ICapeThermoPropertyPackageManager* (i), for restoring a Primary PMC, the PME does not need to pick a Template name from a list and specify it to the Manager. Instead of the PME saving the Template name, the PME calls a method on the Manager that creates and returns the Primary PMC object for the purpose of de-persistence. Subsequently the PME de-persists the returned Primary PMC Object.

A Manager advertises whether it supports creating a new instance from scratch.

As a Primary PMC, a Manager and the objects it creates implement *ICapeUtilities* and therefore may support editing functionality through *ICapeUtilities::Edit*. If a component created by a Manager is being edited, the expectation is to change the configuration of the component. However, editing a Manager translates into managing its list of Templates (e.g., add, remove, rename, edit a Template). A Manager may provide functionality to save, as a Template, the configuration of a particular instance of the Primary PMC Object created by the Manager, for example via a button in the GUI of the Primary PMC Object. Such functionality would allow the end-user to save a Template in the form of an instance in the flowsheet document to transfer the Template to another machine. The Manager may also provide more direct functionality to import/export Templates, for example through the private GUI of the Manager. All the above operations that pertain to the configuration of the Manager are private to the Manager and fall outside of the scope of this interface specification.

In the scope of the Manager, a Template is identified through its name only. In the current version of the specification, no other generic descriptor of a Template is provided.

Through the Manager there is no access to internal information of the Template or the Primary PMCs corresponding to the Template. If more information is needed to select a Template, an instance of a Primary PMC Object must be created from the Template by the Manager so that this additional information may be retrieved from that instance itself. The Primary PMC Object should not be instantiated unnecessarily because it can be costly and prone to errors.

5. Requirements

This chapter defines the overall architectural design, the textual requirements on PME and Managers, and Use Cases.

Through textual requirements, this section explains how a Manager will interact with other software such as a PME or a Primary Process Modelling Component. These functional requirements are mostly written in natural language with technical terms defined in the Glossary when used repeatedly. Requirements are not organized in order of importance, urgency, and convenience.

REQ-MGR-1: A Manager is a primary Process Modelling Component.

Rationale

A Manager is created by a PME so it is a Primary Process Modelling Component.

REQ-MGR-2: A Manager is categorized as a CAPE-OPEN component.

Rationale

The Category ID for a CAPE-OPEN component appears in the registry for each of the CAPE-OPEN Primary PMCs. In future CAPE-OPEN versions, the "CAPE-OPEN Component" Category ID will be used to indicate that CAPE-OPEN version 1.1 is supported by the component.

REQ-MGR-3: Objects instantiated by a Manager are Primary Process Modelling Components.

Rationale

After instantiation by a Manager, from the PME point of view, the Primary PMC Object is indistinguishable from a stand-alone Primary PMC Object of the same type.

REQ-MGR-4: A Manager is registered with CAPE-OPEN Category ID(s) indicating which type(s) of software components the Manager creates.

Rationale

A Manager is a Primary Process Modelling Component. Each Primary Process Modelling Component belongs to a category that allows a PME to create a list of the Primary Process Modelling Components belonging to this category without the need to instantiate each of those. Without the Manager indicating the type(s) of Manager it is, the PME would list Property Package Managers and Chemical Reaction Package Managers without distinguishing between them. There can be no Manager without knowledge of which sort of Templates it manages. Only stand-alone Primary PMC Objects are registered with their own Category IDs.

REQ-MGR-5: Each instance created by a Manager is of the software component type(s) consistent with the Manager type(s).

Rationale

For example, if the Manager is a Property Package Manager, all software components created by the Manager must be Property Packages and must comply with the requirements on a Property Package as outlined in (i). If the Manager is a Chemical Reaction Package Manager, all software components created by the Manager must be Chemical Reaction Packages and must comply with the requirements on a Chemical Package as outlined

in (ii). If the Manager is both a Property Package Manager and a Chemical Reaction Package Manager, all software components created by the Manager must be Property Packages supporting Chemical Reactions and must comply with the requirements on a Property Package supporting Chemical Reactions as outlined in (ii).

REQ-MGR-6: A Manager provides the service to deliver the list of names of its Templates.

Rationale

The name of the Template is needed to request the creation of a Primary PMC based on the Template. For ease of interaction with the Process Engineer, the PME must have access to the list of Templates supported by the Manager. Note that it is not required from the Manager to be able to create Primary PMC Objects from all the Templates listed. Creation may fail for a number of reasons (e.g., ill configured Template, license is not present for a particular feature).

REQ-MGR-7: A Manager creates an instance from a Template with the same name as the Template.

Rationale

Matter of consistency which can be helpful to the Process Engineer. If names need to be unique (names used in a Collection for instance), it is the responsibility of the PME to rename the instance as needed using operations from ICapeIdentification (x).

REQ-MGR-8: The name of a Template follows the rules of the name property in ICapeIdentification.

Rationale

Since the instance created from a Template is identified by the name of the Template, the name of the Template must follow the rules applicable to any name identifying a CAPE-OPEN component through (x).

REQ-MGR-9: A Template is identified by its name in a case insensitive manner.

Rationale

Case insensitiveness is in accordance with choices made in CAPE-OPEN.

REQ-MGR-10: All Template names must be unique within a Manager.

Rationale

Names are used by the Manager and the PME to uniquely identify Templates.

REQ-MGR-11: If a PME persists the state of a Primary PMC Object created via a Manager, the PME persists also sufficient information to create the Manager.

Rationale

The PME cannot directly instantiate a Primary PMC Object originally created via a Manager. Therefore, the PME must be able to find which Manager needs to be instantiated in order for the proper Manager to load back the Primary PMC Object. The PME needs to keep sufficient information (e.g., ProgID, version independent ProgID, or CLSID) that defines uniquely the Manager responsible for the instantiation of the Primary PMC Object.

REQ-MGR-12: A Primary PMC Object created via a Manager implements persistence.

Rationale

As per the design decision, persistence is mandatory for all Primary PMC Objects created via a Manager.

REQ-MGR-13: A Manager advertises whether it supports creating a new instance from scratch.

Rationale

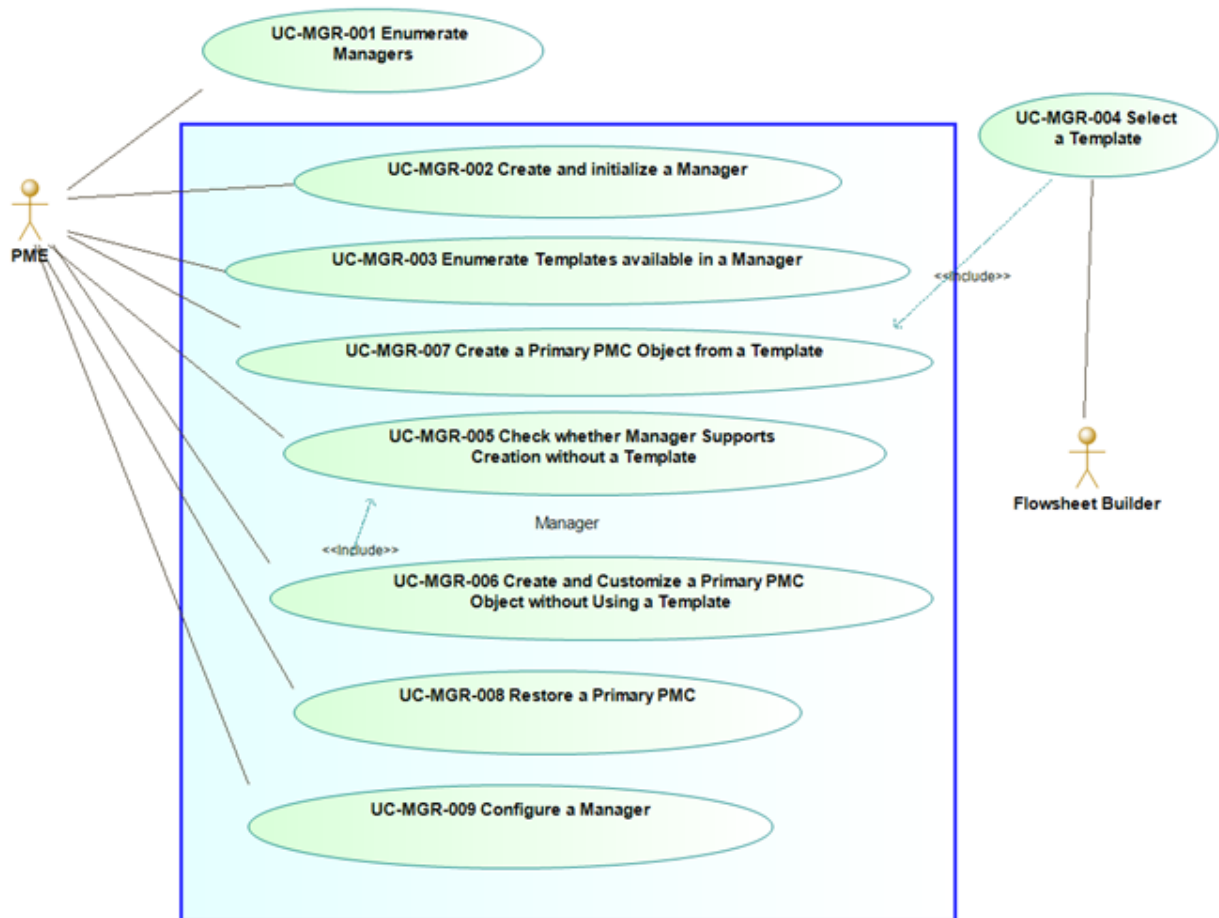
A Manager may or may not support creation of a new instance without a Template. The PME can properly arrange its user interface (for example enabling/disabling "new instance") if it knows whether the Manager supports this functionality.

REQ-MGR-14: PME must initialize the Primary PMC object after creation.

Rationale

To conform with the Utilities interface specification (iv), the PME must initialize the Primary PMC Object, irrespective of the way this object was created by the Manager.

6. Use Cases



Use Cases defined here are generic Use Cases for Package Managers. Use Cases specific to a particular category of Package Managers are found in the Interface Specification Document where the particular category is defined.

UC-MGR-001: ENUMERATE MANAGERS

Typically, a PME displays to the Process Engineer a list of available stand-alone Primary PMCs as well as a list of Managers providing Primary PMCs of the same type.

This Use Case fulfills the second part of this process.

Actor: <PME>

Priority: <High>

Status: This Use Case is fulfilled by middleware functionalities.

Pre-conditions: None

Flow of events: From the middleware component registry, PME enumerates all available Managers fitting

a given purpose (Managers are registered using the Category ID associated with the Manager type).

Errors: None

Post-conditions: The available Managers have been enumerated.

Uses: <><>

UC-MGR-02: CREATE AND INITIALIZE A MANAGER

A PME creates and initializes a Manager to prepare it for use.

This Use Case applies generic instantiation procedures combined with initialization via ***ICapeUtilities::Initialize***.

Actor: <PME>

Priority: <High>

Status: Use Case is fulfilled by middleware functionalities and ***ICapeUtilities::Initialize***.

Pre-conditions: None

Flow of events: 1. PME instantiates the Manager using middleware functionalities. 2. PME sets the Simulation Context on the Manager. 3. PME initializes the Manager using ***ICapeUtilities::Initialize***. (Note: At the end of the Manager's lifespan, it must be terminated by the PME using ***ICapeUtilities::Terminate***.)

Errors: The Manager cannot be created.

Post-conditions: The Manager is ready for use.

Uses: <><>

UC-MGR-03: ENUMERATE TEMPLATES AVAILABLE IN A MANAGER

The PME retrieves the list of Template names available from a Manager.

This Use Case is designed to enable the PME to present available Templates to the Process Engineer.

Actor: <PME>

Priority: <High>

Status: Use Case is fulfilled by the ***ICapeManager::GetTemplateList*** method.

Pre-conditions: The Manager has been created using UC-MGR-02.

Flow of events: 1. The PME calls ***GetTemplateList***. 2. The Manager returns a list of Template names (which may be empty).

Errors: The Manager fails to provide a list of Template names.

Post-conditions: The available Templates have been enumerated and can be presented to the Process Engineer.

Uses: <><>

UC-MGR-04: SELECT A TEMPLATE

A Template is selected from the list provided by the Manager.

The PME presents the list of Template names to the Flowsheet Builder, who then selects one. Additional details may be obtained by instantiating a Primary PMC Object (see UC-MGR-07).

Actor: <**Flowsheet Builder**>

Priority: <**High**>

Status: Fulfilled by functionality internal to the PME.

Pre-conditions: UC-MGR-03 has been executed successfully and the list of Template names is not empty.

Flow of events: 1. The PME presents a list of Template names. 2. The Flowsheet Builder selects a Template.

Errors: The Flowsheet Builder cancels the selection.

Post-conditions: A Template has been selected.

Uses: <**UC-MGR-07**>

UC-MGR-05: CHECK WHETHER MANAGER SUPPORTS CREATION WITHOUT A TEMPLATE

The PME verifies if the Manager allows creation of a new instance without using a Template.

This Use Case is used to enable or disable the "new instance" functionality in the PME's user interface.

Actor: <**PME**>

Priority: <**High**>

Status: Fulfilled through the property *SupportsCreateNew* of *ICapeManager*.

Pre-conditions: UC-MGR-02 has been executed successfully.

Flow of events: The PME obtains the value of the *SupportsCreateNew* property from *ICapeManager*.

Errors: None

Post-conditions: The PME knows whether the Manager supports creation without a Template and adjusts the user interface accordingly.

Uses: <><>

UC-MGR-06: CREATE AND CUSTOMIZE A PRIMARY PMC OBJECT WITHOUT USING A TEMPLATE

A new Primary PMC Object is created from scratch without using a Template.

The PME optionally checks if the Manager supports creation without a Template (UC-MGR-05) and then calls *ICapeManager::CreateNew*. The Manager may provide customization options for the new PMC Object.

Actor: <PME>

Priority: <Medium>

Status: Fulfilled by *ICapeManager::CreateNew* and the *LastCreateNewWasCanceledByUser* property.

Pre-conditions: UC-MGR-02 has been executed successfully.

Flow of events: 1. The PME optionally exercises UC-MGR-05. 2. The PME calls *ICapeManager::CreateNew*. 3. The Manager creates a new Primary PMC Object (default or empty) and may allow customization. 4. If the Flowsheet Builder cancels the customization, an exception is raised.

Errors: The creation may be cancelled by the Flowsheet Builder; the Manager might not support creation without a Template; or the PMC Object cannot be created (e.g., due to out-of-memory or license issues).

Post-conditions: A Primary PMC Object is available to the PME.

Uses: <UC-MGR-05>

UC-MGR-07: CREATE A PRIMARY PMC OBJECT FROM A TEMPLATE

A Primary PMC Object is created based on a selected Template.

The PME calls *ICapeManager::CreateFromTemplate* with the name of the Template. If recognized, the Manager creates and delivers an instance of the Primary PMC.

Actor: <PME>

Priority: <High>

Status: Fulfilled by *ICapeManager::CreateFromTemplate* and middleware functionalities.

Pre-conditions: UC-MGR-02 has been executed; and the PME knows which Template to use (e.g., via UC-MGR-04).

Flow of events: The PME executes *CreateFromTemplate* with the Template name, and the Manager returns an instance of the Primary PMC.

Errors: The Template may not be recognized; or the PMC Object might not be created (e.g., due to memory or licensing issues).

Post-conditions: An object representing the Primary PMC is instantiated.

Uses: <><>

UC-MGR-08: RESTORE A PRIMARY PMC

A persisted Primary PMC Object is restored.

The PME calls *CreateForLoad* on the Manager, then sets the simulation context, loads persisted data, and initializes the object.

Actor: <PME>

Priority: <High>

Status: This Use Case is fulfilled by the **CreateForLoad** method of **ICapeManager**.

Pre-conditions: UC-MGR-02 has been executed and persisted data for the Primary PMC Object is available.

Flow of events: The PME calls **CreateForLoad**, obtains an un-configured Primary PMC Object, optionally sets the Simulation Context, invokes Load, and then initializes the object via **ICapeUtilities::Initialize**.

Errors: The PMC Object may not be created (e.g., due to out-of-memory or licensing issues).

Post-conditions: The Primary PMC Object is available, initialized, and its persisted data restored.

Uses: <><>

UC-MGR-09: CONFIGURE A MANAGER

A Manager is configured via a dedicated editing function.

The PME executes **ICapeUtilities::Edit** on the Manager. The Manager displays a configuration dialog to reconfigure its Templates. Upon completion, the PME assumes the list of Templates has changed.

Actor: <PME>

Priority: <Medium>

Status: Fulfilled by **ICapeUtilities::Edit**.

Pre-conditions: UC-MGR-02 has been executed successfully.

Flow of events: The PME executes Edit on the Manager. The Manager may present a modal window for reconfiguration. Once finished, the PME retrieves the updated list of Templates.

Errors: The Manager may not implement Edit (raising a not-implemented error).

Post-conditions: A new list of Templates is available to the PME and other PMEs.

Uses: <><>

7. Conceptual Object Model

7.1 Manager

Interface	Required	Notes
ICapeManager	Required	

8. Analysis and Design

8.1 Interface Descriptions

ICAPEMANAGER

Interface Name	<i>ICapeManager</i>
Method Name	<i>SupportsCreateNew</i>
Returns	CapeBoolean

Description

Returns True if the Manager supports creation of new Primary PMC Objects through CreateNew().

Arguments

None

Notes on errors

ECapeNotImplemented – Raised if the function is not implemented.

General notes

If *CreateNew* function is not implemented, the property value is False. The PME can use this property to properly arrange its user interface, for example enable/disable a button.

Interface Name	<i>ICapeManager</i>
Method Name	<i>LastCreateNewWasCanceledByUser</i>
Returns	CapeBoolean

Description

Returns True if within last execution of CreateNew in the current thread, operation was cancelled by Process Engineer.

Arguments

None

Notes on errors

ECapeNotImplemented – Raised if **CreateNew** was not previously called in the current thread.

General notes

If **CreateNew** was not previously called in the current thread, the property may raise an error or have an arbitrary value.

Interface Name	<i>ICapeManager</i>
Method Name	<i>CreateForLoad</i>
Returns	–

Description

Returns an instance of the Primary Process Modelling Component object that the PME will de-persist.

Arguments

Name	Type	Description
[Out] CreateForLoad	CapeInterface	An instance of the Primary Process Modelling Component object that the PME will de-persist

Notes on errors

ECapeNotImplemented – No specific error.

General notes

As per the requirements in the Persistence Common Interface Specification (vi) and the Utilities Common Interface Specification (iv), before using the created Object, the PME must invoke Load on the persistence interface of the created Object, followed by initialization (*ICapeUtilities::Initialize*). Simulation Context may be set at any time, preferably before calling Load.

Interface Name	<i>ICapeManager</i>
Method Name	<i>CreateNew</i>
Returns	–

Description

Returns a Primary PMC Object that is not based on a Template and is not de-persisted.

Arguments

Name	Type	Description
[Out] CreateNew	CapeInterface	

Notes on errors

ECapeNotImplemented – Raised if the Manager does not allow for the creation of a new PMC Object through the interface.

General notes

The returned PMC Object should have a name. The PME is allowed to change the name using *ICapeIdentification* interface.

Interface Name	<i>ICapeManager</i>
Method Name	<i>GetTemplateList</i>
Returns	CapeArrayString

Description

Returns the list of the names of all Templates available from the Manager.

Arguments

None

Notes on errors

ECapeInvalidArgument – No specific error.

General notes

The returned list can be empty.

Interface Name	<i>ICapeManager</i>
Method Name	<i>CreateFromTemplate</i>
Returns	CapeString

Description

Returns a Primary PMC Object based on the Template specified by the client of the Manager.

Arguments

None

Notes on errors

ECapeInvalidArgument – Raised if templateName argument does not identify a Template known to the Manager.

General notes

The list of names of available Templates can be obtained by executing *GetTemplateList*.