# CAPE-OPEN

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

# Open Interface Specification:

# Identification Common Interface



www.colan.org

# ARCHIVAL INFORMATION

| | |
|---|---|
| Filename | Identification Common Interface.doc |
| Authors | CO-LaN consortium |
| Status | Public |
| Date | August 2003 |
| Version | version 3 |
| Number of pages | 24 |
| Versioning | version 3, reviewed by Jean-Pierre Belaud, August 2003 |
| | version 2, Methods & Tools group, September 2000 |
| | |
| | |
| Additional material | |
| Web location | www.colan.org |
| Implementation specifications version | CAPE-OPENv1-0-0.idl (CORBA) |
| | CAPE-OPENv1-0-0.zip and CAPE-OPENv1-0-0.tlb (COM) |
| Comments | |

# IMPORTANT NOTICES

# SUMMARY

This document describes a Common Interface proposed by the Methods & Tools group: the *Identification Common Interface*. The Common Interfaces are interfaces and implementation models for handling concepts that may be required by any CO interface specification.

The Identification interface was already used by the Unit Operation and SMST interface specifications and implementation of the UNIT and SMST prototypes, and proved to be useful.

# CONTENTS

# LIST OF FIGURES

# 1. Introduction

This document sets out a proposal for a cross work package concept: Identification. This interface will be used by those CAPE-OPEN components that wish to expose its name and description. This information refers to an instance of the component, not to the software class.

As illustration of requirements, the CAPE-OPEN Unit Interface Specification document states that within the scope of Unit component: ***What follows, as a corollary, is that every registered CAPE-OPEN standard component, used to deliver the CAPE-OPEN interfaces has to support the ICapeIdentification interface***.

Please refer to Methods & Tools Integrated Guidelines in order to have an introduction to the Common Interfaces within the CO standard.

# 2. Requirements

## 2.1 Textual requirements

### 2.1.1 General requirements

A particular Use Case in a system may contain several CAPE-OPEN components of the same class. The user should be able to assign **different names and descriptions** to each instance in order to refer to them unambiguously and in a user-friendly way. Since not always the software components that are able to set these identifications and the software components that require this information have been developed by the same vendor, a CAPE-OPEN standard for setting and getting this information is required.

So, the component will not usually set its own name and description: the user of the component will do it.

### 2.1.2 Requirements from the existing CO interface specifications

As illustration, we remind requirements coming from the existing interface specification and being connected with the Identification concept:

The Unit Operations Interfaces have the following requirements:

❑ If a flowsheet contains two instances of a Unit Operation of a particular class, the COSE needs to provide the user a textual identifier to distinguish each of the instances. For instance, when the COSE requires to report about an error occurred in one of the Unit Operations.

❑ When the COSE shows the user its GUI to connect the COSE's streams to the Unit Operation ports, the COSE needs to request the Unit for its list of available ports. For the user to identify the ports, the user needs some distinctive textual information for each of them.

❑ When the COSE exposes to the user its interfaces to browse or set the value of an internal parameter of a Unit Operation, the COSE needs to request the Unit for its list of available parameters. No matter if this COSE's interface is a GUI or a programming interface, each parameter must be identified by a textual string.

The ICapeThermoMaterialObject (used by both Unit and Thermo interfaces):

❑ If a Unit Operation has encountered an error accessing a stream (ICapeThermoMaterialObject), the Unit might decide to report it to the user. It would be desirable the stream to have a textual identifier for the user to be able to quickly know which stream failed.

The Thermodynamic Interfaces have the following requirements:

❑ The ICapeThermoSystem and the ICapeThermoPropertyPackage interfaces don't require an identification interface, since both of them have been designed as singletons (a single instance of each component class is required). That means that there is no need to identify this instance: its class description would be enough. However, the user might decide anyway to assign a name or a description to the CAPE-OPEN property systems or property packages used in her/his flowsheet. Furthermore, if these interfaces evolve, the singleton approach could be removed. In this case, identifying each instance will be a must.

The Solvers Interfaces have the following requirements:

❑ Many objects should provide the functionality coming from the Identification Common Interface.

The SMST Interfaces have the following requirements:

❑ The CO SMST component package depends on the Identification Interface package. The interface ICapeSMSTFactory must provide the Identification capabilities.

## 2.2 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. Mandatory information is the definition of actors participating in the use cases and the use cases themselves. Optional items are priorities for use cases, further categorization of use cases.

Most of the use cases will make no reference to any other CO interfaces, since the Identification Interface is of general purpose. However, a few use cases have been added as concrete examples of how this interface has been used by the Unit Interfaces.

*Use Case Categories*

❑ **General Purpose Use Cases.** Use Cases that express in a general way the functionality of the interface.

❑ **Specific Use Cases.** Use Cases that show concrete examples of how the CO Identification Interface has been used by other CO Interfaces.

*Use Cases Priorities*

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

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

### 2.2.1 Actors

- **Client.** Any person or software component that decides to browse or modify the name or description of a CAPE-OPEN component. As concrete examples under the CO scope, the client could be:

  o **Physical Properties Developer.** The human being who is notionally a physical properties expert and will set up the physical property options for use by the normal simulator end user. In principle, the Physical Properties developer is responsible for physical properties quality assurance in the organization using simulation.

  o **Unit Operation Developer.** The human being who is notionally a Unit Operation expert. This person will develop the code of the simulation of a Unit Operation, exposing the required CAPE-OPEN interfaces.

  o **Flowsheet Builder.** The person who sets up the flowsheet, the structure of the flowsheet, chooses thermo models and the 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.

o **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.

### 2.2.2 List of Use Cases

❑    UC-001 : Get Component Name

❑    UC-002 : Set Component Name

❑    UC-003 : Get Component Description

❑    UC-004 : Set Component Description

❑    UC-005 : Set Identification for Unit Ports

❑    UC-006 : Browse Identified Ports

### 2.2.3 Use Cases Map

As can be seen in the figure, different clients of a CO component can request or set its name or description. It might seem that having a single type of actor (client) is too simplistic. However, trying to define more concrete types of actors would induce thinking on the Identification Interface in a less flexible way. On the contrary: any clients can access any Use Case, in any order and for any purpose.
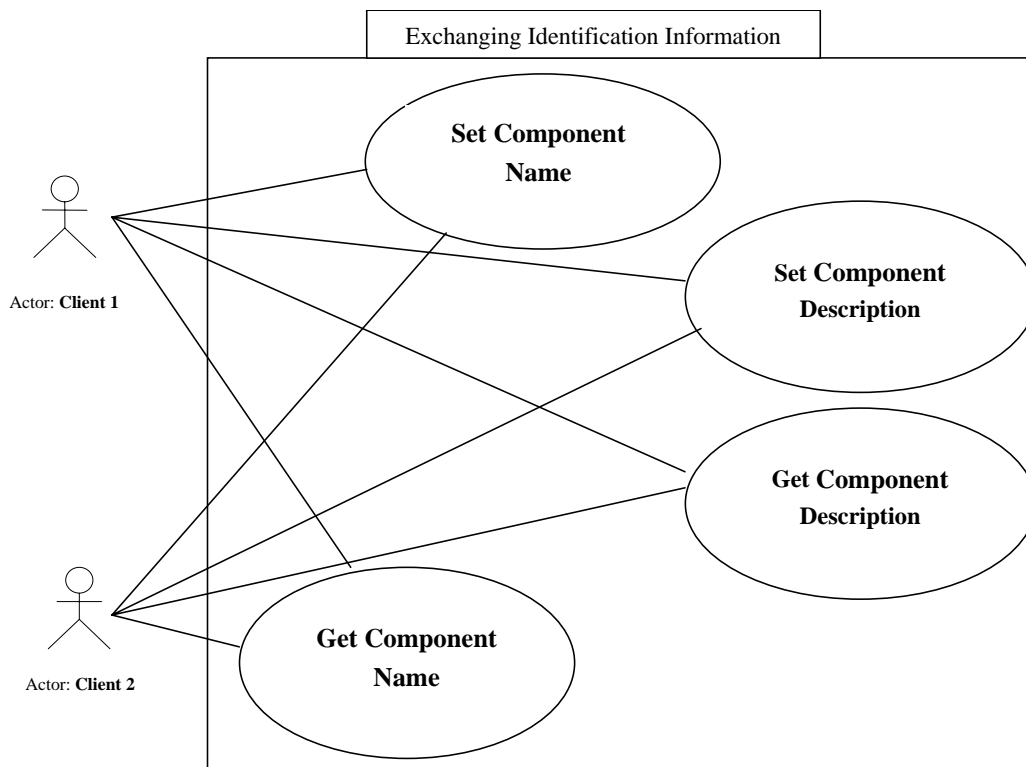


**Figure 1 General Purpose**

### 2.2.4 Use Cases

This subsection lists all the Use Cases.

UC-001 : GET COMPONENT NAME

<u>Actors</u>: Client

<u>Priority</u>: <High>

<u>Classification</u>: <General Purpose Use Cases >

<u>Context</u>: The component name is so general that it can be used under any context.

<u>Pre-conditions</u>: The component must assign an initial value for its component name.

<u>Flow of events</u>: The component retrieves the value of its name, as it was set by the last <Set Component Name> use-case. This value is returned.

<u>Post-conditions</u>: The state of the component does not change.

<u>Errors</u>:

<u>Uses</u>:

<u>Extends</u>:

UC-002 : SET COMPONENT NAME

<u>Actors</u>: Client

<u>Priority</u>: <High>

<u>Classification</u>: <General Purpose Use Cases >

<u>Context</u>: The component name is so general that it can be used under any context.

<u>Pre-conditions</u>:

<u>Flow of events</u>: The component stores the passed name.

<u>Post-conditions</u>: Subsequent calls to the <Get Component Name> use-case will return the passed name..

<u>Errors</u>:

<u>Uses</u>:

<u>Extends</u>:

UC-003 : GET COMPONENT DESCRIPTION

<u>Actors</u>: Client

<u>Priority</u>: <Low>

<u>Classification</u>: <General Purpose Use Cases >

<u>Context</u>: The component description is so general that it can be used under any context.

<u>Pre-conditions</u>: The Component must assign an initial value for its component description.

Flow of events: The component retrieves the value of its description, as it was set by the last <Set Component Description> use-case. This value is returned.

Post-conditions: The state of the component does not change.

Errors:

Uses:

Extends:

## UC-004 : SET COMPONENT DESCRIPTION

Actors: Client

Priority: <Low>

Classification: <General Purpose Use Cases >

Context: The component description property is so general that it can be used under any context.

Pre-conditions:

Flow of events: The component stores the passed description..

Post-conditions: Subsequent calls to the <Get Component Description> use-case will return the passed argument.

Errors:

Uses:

Extends:

## UC-005 : SET IDENTIFICATION FOR UNIT PORTS

Actors: Unit Operation Developer

Priority: <Low>

Classification: <Specific Use Cases>

Context: The Unit Operation Developer is creating the ports of a Unit Operation.

Pre-conditions: <The Unit Operation Developer has created at least one port of a Unit Operation.>

Flow of events: After creating a port, the UO developer will set the values of its name and description.

Post-conditions: Subsequent calls to <Get Component Description> use-case and <Get Component Name> use-case will return the values set by the Unit Operation Developer.

Errors:

Uses: <Set Component Description>, <Set Component Name>

Extends:

## UC-006 : BROWSE IDENTIFIED PORTS

Actors: Flowsheet Builder

Priority: <Low>

Classification: <Specific Use Cases>

Context: The Flowsheet Builder is browsing or editing the ports configuration of a Unit Operation.

Pre-conditions: The Flowsheet Builder has inserted a Unit Operation in a flowsheet.

Flow of events: The COSE accesses the ports collection of the selected Unit Operation. For each of the elements of the collection, the COSE obtains its Identification object. Now the COSE can show the name and/or description of each of the Unit Operation's ports.

Post-conditions: The state of the component does not change.

Errors:

Uses: <Get Component Description>, <Get Component Name>

Extends:

## 2.3 Sequence diagrams

### 2.3.1 Exchanging Identification information between two clients (ref. SQ 001)
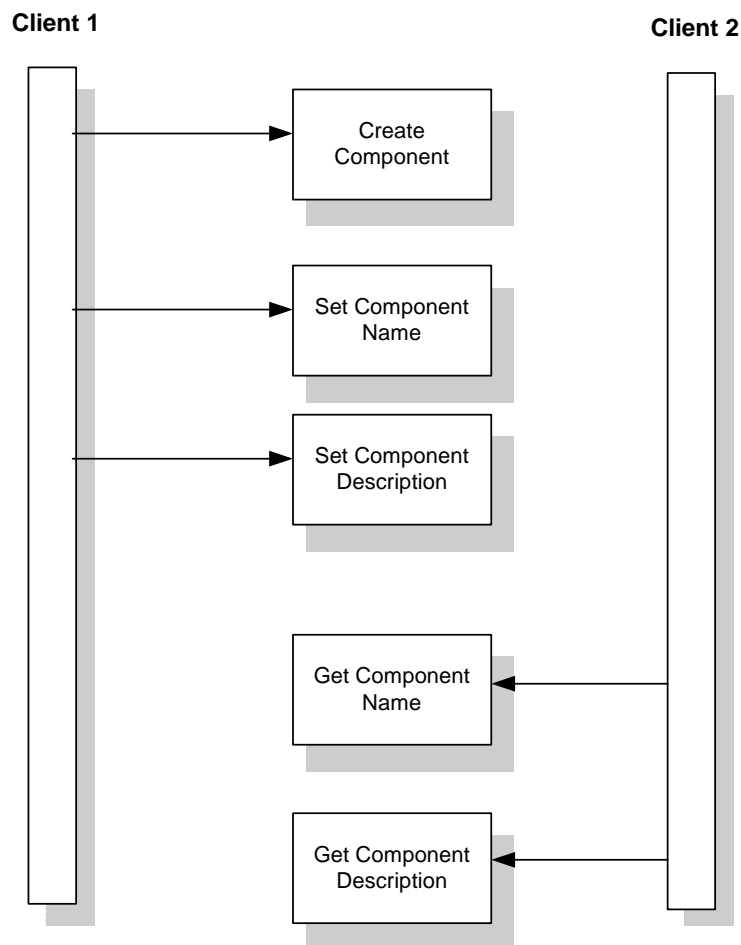


**Figure 2 Exchanging Identification information between two clients**

# 3.    Analysis and design

## 3.1    Overview

This chapter introduces the analysis models developed for the Identification Interface. These models are described by a combination of text and UML diagrams, to show the solutions derived for the requirements expressed in the Use Cases. The UML diagrams presented are the interface, state and component diagrams, as well as some explanatory sequence diagrams.

## 3.2    Sequence diagrams

This section lists the sequence diagrams.

SQ-002: SETTING THE IDENTIFICATION OF A COMPONENT



**Figure 3 Setting the identification of a Component**
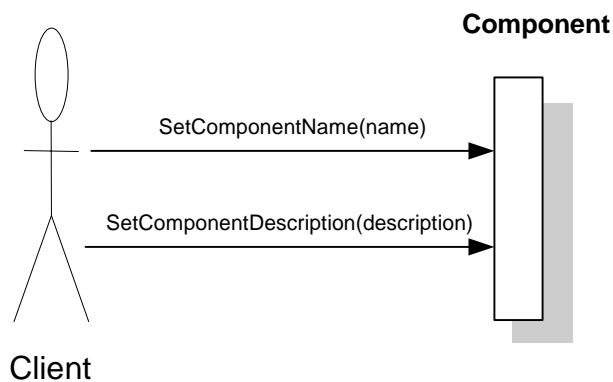
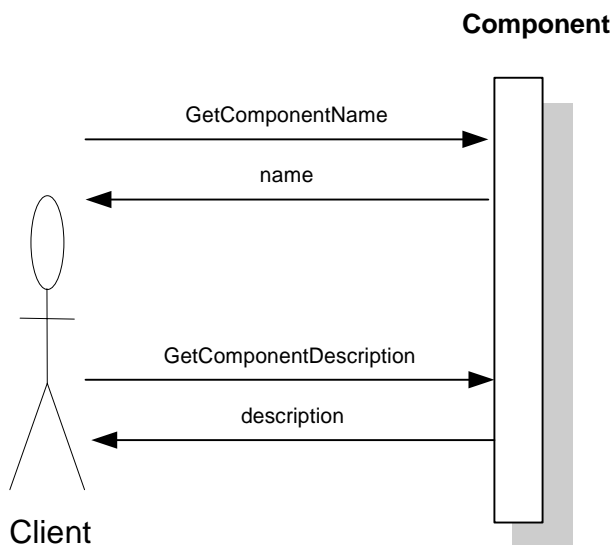SQ-003: GETTING THE IDENTIFICATION FROM A COMPONENT



**Figure 4 Getting the identification from a Component**

### 3.3 Interface diagrams

This section presents the interface diagram. It contains one interface.
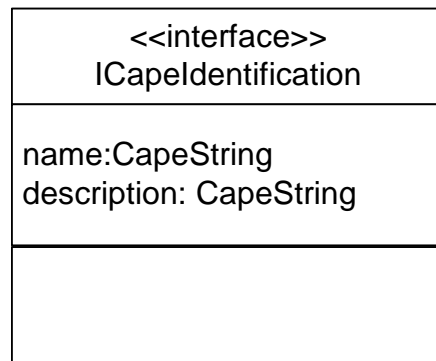
**Figure 5 ICapeIdentification Interface Diagram**

### 3.4 State diagrams

This section presents the State Diagram.
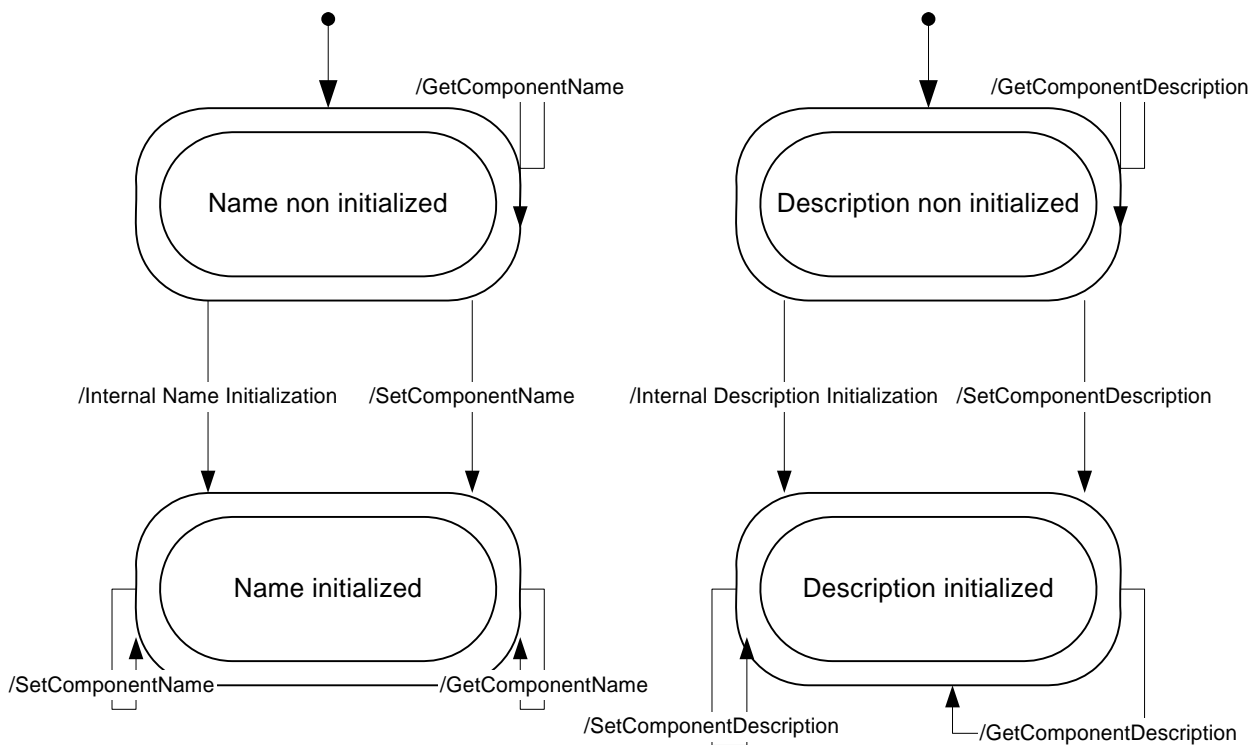
**Figure 6 ICapeIdentification State Diagram**

It has not been possible to design a single state diagram for the identification interface. Since the name and description attributes are completely independent, it does not make any sense to try to cross-link their states.

The only difference between the 'Name non initialized' and the 'Name initialized' states is that the first one returns an undetermined value to the GetComponentName action.

## 3.5    Interfaces descriptions

### 3.5.1  ICapeIdentification

| Interface Name | ICapeIdentification |
|---|---|
| Method Name | GetComponentName |
| Returns | CapeString |

**Description**

Gets the name of the component.

**Arguments**

None

**Errors**

ECapeUnknown

| Interface Name | ICapeIdentification |
| --- | --- |
| Method Name | GetComponentDescription |
| Returns | CapeString |

## Description

Gets the description of the component.

## Arguments

None

## Errors

ECapeUnknown

| Interface Name | ICapeIdentification |
| --- | --- |
| Method Name | SetComponentName |
| Returns | -- |

## Description

Sets the name of the component.

## Arguments

| Name | Type | Description |
| --- | --- | --- |
| [in] name | CapeString | The name of the component |

## Errors

ECapeUnknown, ECapeInvalidArgument

| Interface Name | ICapeIdentification |
|---|---|
| Method Name | SetComponentDescription |
| Returns | -- |

## Description

Sets the description of the component.

## Arguments

| Name | Type | Description |
|---|---|---|
| [in] desc | CapeString | The description of the component |

## Errors

ECapeUnknown, ECapeInvalidArgument

## 3.6 Scenarios

This section describes the scenario of actions that was used to validate the set of Identification Common Interface. This scenario exercises all the methods of the ICapeIdentification interface. The scenario is focused in a requirement of the Unit Operation interfaces.

### 3.6.1 Browse Unit Ports (ref. SC-001)

One of the steps that a COSE must follow to integrate a CO Unit Operation into a flow-sheet, is to connect the COSE's streams to the Unit Operation ports. For doing that, the COSE needs to request the Unit for its list of available ports. For the user to identify the ports, the user needs some distinctive textual information for each of them.

*Set identification of the Unit Ports.*

❑ The Unit Operation creates at least one port and adds it to its Ports collection.

❑ For each of the created ports, the Unit Operation set the values of its name and description setting the component name and component description..

*Browse Identified ports*

❑ The Flowsheet Builder has inserted a Unit Operation in a flowsheet.

❑ The Flowsheet Builder is browsing or editing the ports configuration of the Unit Operation.

❑ The COSE accesses the Unit Operation's ports collection attribute. For each of the elements of the collection, the COSE obtains its ICapeIdentification interface. Now the COSE can show the name and/or description of each of the Unit Operation's ports.

# 4.    Interface specifications

## 4.1    COM IDL

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

## 4.2    CORBA IDL

```
// You can get these intructions in CAPE-OPENv1-0-0.idl within the
CAPEOPEN100::Common::Identification module
```

# 5. Notes on the interface specifications

## 5.1 COM Specifications

## 5.2 CORBA Specifications

# 6.    Prototypes implementation

```
'this implementation can be used in the implementation of the following 'classes...

Private m_portID As String
Private m_description As String

Implements ICapeIdentification

'*****************************************************************
'*                                                               *
'*            ICapeIdentification Interface Implementation        *
'*                                                               *
'*****************************************************************

Private Property Get ICapeIdentification_ComponentDescription() As String
    ICapeIdentification_ComponentDescription = m_description
End Property

Private Property Get ICapeIdentification_ComponentName() As String
    ICapeIdentification_ComponentName = m_portId
End Property
```

## 7.    Specific glossary terms

# 8.    Bibliography

(i)     Methods and Tools Integrated Guidelines

(ii)    Error Handling Strategy: Error Common Interface

# 9.    Appendices