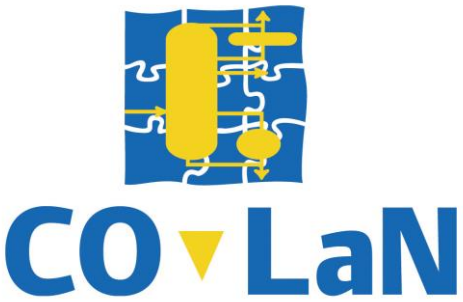
CAPE-OPEN

Expanding Process Modelling Capability
through Software Interoperability Standards

Errata and Clarifications

Identification Common Interface v1.0



www.colan.org

ARCHIVAL INFORMATION

Filename	Identification_Errata_and_Correction_1.0_1.012.docx
Authors	CO-LaN consortium
Status	Public
Date	December 2014
Version	version 1.0.014
Number of pages	11
Versioning	Version 1.0.000 edited by Bill Barrett (USEPA)
	Version 1.0.012 approved by M&T SIG June 6, 2014
	Version 1.01.014 approved by M&T SIG Nov 5, 2014
Additional material	
Web location	
Implementation specifications version	1.0
Comments	

IMPORTANT NOTICES

Disclaimer of Warranty

CO-LaN documents and publications include software in the form of *sample code.* Any such software described or provided by CO-LaN --- in whatever form --- is provided "as-is" without warranty of any kind. CO-LaN and its partners and suppliers disclaim any warranties including without limitation an implied warrant or fitness for a particular purpose. The entire risk arising out of the use or performance of any sample code --- or any other software described by the CAPE-OPEN Laboratories Network --- remains with you.

CO-LaN is a not for profit organization established under French law of 1901.

Trademark Usage

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

Microsoft, and the Component Object Model (COM) are registered trademarks of Microsoft Corporation.

SUMMARY

This document describes clarification to, and new requirements for the implementation of the CAPE-OPEN Identification Common Interface specification version 1.0. In particular, this document addresses the requirement of all CAPE-OPEN objects to implement the *ICapeIdentification* interface, places the burden of *ComponentName* uniqueness for Collection members on the Collection owner, and provides guidance on the minimum length of the *ComponentName,* as well as character sets and use of white space in both the *ComponentName* and *ComponentDescription*.

ACKNOWLEDGEMENTS

Many individuals and their organizations have contributed to this document.

CONTENTS

1. Introduction

This document lists errata and clarifications that have been raised and added after the formal approval process for the CAPE-OPEN Identification Common Interface specification version 1.0.

The intent is to clarify the interface specification where needed and to pinpoint what is missing in the specification document.

2. Audience

This document is intended primarily for software developers who want to build CAPE-OPEN PMEs. It is also intended for the developers of other software components, such as Unit Operations and Reaction Packages, which make use of these interfaces.

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

3. Clarifications

Context 1: The *ICapeIdentification* interface provides a means to identify each CAPE-OPEN object. Each CAPE-OPEN object is required to implement *ICapeIdentification* interface, and provide a value for the *ICapeIdentification.ComponentName* property for the CAPE-OPEN object.

Issue 1: The *ICapeIdentification.ComponentName* property is used by the Collection Common Interface to identify and return objects (such as Parameters or Ports) from these Collections. There is no mechanism within either the Collection Common Interface specification or the Identification Common Interface specification to ensure that each object within a Collection has a unique *ICapeIdentification.ComponentName* property.

Requirement 1: The Collection owner will ensure that the *ICapeIdentification.ComponentName* property of each object within the Collection is unique. This includes Unit Operation and Material Object Collection owned by the Process Modeling Environment (PME), as well as the Port Collection of a unit operation and the Parameter Collection of primary PMCs access through the *ICapeUtilities* interface. The requirement for the Collection owner to ensure uniqueness of item names is also conveyed in an Errata and Clarifications document for the Collection Common Interface Specification.

Context 2: There is a need to have an identification of various objects that implement CAPE-OPEN interfaces. For example, material and simulation context objects are not explicitly required to implement *ICapeIdentification*, although many PMEs do provide implementations. Future use cases, such as the proposed Flowsheet Monitoring interface, will require these objects to implement *ICapeIdentification*.

Issue 2: The *ICapeIdentification* interface is not available on all objects that need to be identified by name.

Discussion 2: The specifications provide inconsistent guidance on which objects must support the *ICapeIdentification* interface. The level of support has been discussed elsewhere, which indicated that both *ComponentName* and *ComponentDescription* are

supported, and minimum requirements for a name or description. In particular, Section 9.2.2 of the M&T Guidelines indicates that all PMC objects implement *ICapeIdentification*. There was no requirement placed on PME objects, including Material Objects to implement *ICapeIdentification*. The ability to obtain the name of a PME object was deemed useful, especially for flowsheet monitoring.

Requirement 2: All objects that support CAPE-OPEN interfaces must expose the *ICapeIdentification* interface. This includes secondary objects, such as Port Collection and Parameter Collection, as well as PME-owned objects such as individual Material Objects and the Simulation Context.

Issue 3: There are no guidelines for the characters that can be used in the *ICapeIdentification.ComponentName* and *ICapeIdentification. ComponentDescription* properties.

Discussion 3: The *ICapeIdentification.ComponentName* and *ICapeIdentification. ComponentDescription* properties are *CapeString*-valued. In general, the character set and encoding will follow the rules established for the middle-ware platform used for the specific CAPE-OPEN implementation.

In the current Microsoft Component Object Model (COM)-based CAPE-OPEN implementation, the *CapeString* data type used for *ComponentName* and *ComponentDescription* properties is defined as a BASIC String, or BSTR. The BSTR is a composite structure that consists of a four-byte integer length prefix, the UTF-16 encoded Unicode string data, and a two-byte, wide character null terminator (http://msdn.microsoft.com/en-us/library/windows/desktop/ms221069(v=vs.85).aspx). Internally, Microsoft Windows applications use the UTF-16 implementation of Unicode (http://msdn.microsoft.com/en-us/library/windows/desktop/dd317743 (v=vs.85).aspx). Any character that can be entered by a user and encoded by the application into the UTF-16 encoded version of Unicode would be a valid character for use as part of the value of the *ICapeIdentification. ComponentName* and *ICapeIdentification.ComponentDescription* properties in COM-based CAPE-OPEN.

A BSTR can contain null characters, either an 8-bit null char/byte or a 16-bit UTF-16 encoded Unicode zero-valued wide character. However, in the context of CAPE-OPEN, embedding null characters into the string is disallowed; the BSTR data is not to be interpreted as a generic information container (which can even have an odd byte count) as in COM, but rather as a null-terminated character string.

Issue 4: No minimum or maximum length is specified for the *ICapeIdentification.ComponentName* property.

Discussion 4: Any limit on the maximum length of the *ICapeIdentification.ComponentName* property string imposed by CO-Lan would be arbitrary. Middleware platforms may have limits imposed on the length of a string that can be transported in a data structure, which may place an upper bound on the length of the string. In the case of COM, the specification of the *CapeString* data type as a BSTR data type contains a string length expressed as a four-byte (32 bit) integer, which provides the upper bound for the length of the string as greater than 4 billion letters.

PMEs may limit the length of a *ComponentName* that can be assigned to a PMC for practical reasons associated with display of the name. This limitation will likely also restrict the length of a *ComponentName* that can be set through the PME as well.

As the *ICapeIdentification.ComponentName* property is to be unique within the collection containing the PMC, the string must contain at least one character.

Requirement 4: The minimum length of the string returned by the *ICapeIdentification.ComponentName* property will be one (1) character.

Issue 5: There are no guidelines for the use of white space (blank characters) in the *ICapeIdentification.ComponentName* and *ICapeIdentification.ComponentDescription* properties.

Discussion 5: Use of whitespace in component names or descriptions aid readability, and should not be unnecessarily restricted.

There are various types of whitespace characters that can be used, including tabs, line feed, carriage returns, new lines, and form feeds. Different character set encodings and middleware platforms may display or treat whitespace differently.

As component names are used by the *ICapeCollection.Item* method to access a Collection member by name, limitations of middleware specific string comparison methods may limit the use of whitespace in the component name. Any such middleware specific rule should be noted in the CAPE-OPEN specification for use of the middleware, and enforced in the CAPE-OPEN implementation of the specific middleware. In general, the *ComponentName* property should only use the blank space (ASCII 0x20, or equivalent Unicode whitespace) character for word-dividing whitespace.

Since the *ComponentDescription* property is used to provide information about a PMC to the Flowsheet User, greater flexibility in the use of whitespace should be allowed. In particular, horizontal tabs, end-of-line, new line, and carriage return characters should be allowed. PMEs should provide as much flexibility as possible in the entry and display of *ComponentDescription* property values, including the use of scrolling, word-wrapped, multi-line edit and display controls.

There is no minimum length to the *CapeDescription* property, and an empty string is a valid *CapeDescription* value. Visual Basic and COM utilizes a NULL-valued BSTR pointer (*vbNullString*) as an empty string. Alternatively, Microsoft .NET provides a *String.Empty* constant, that is a zero-length string (“”). Both the *vbNullString* and *String.Empty* are valid *ComponentDescription* values. PMEs should recognize that a NULL string (*vbNullString*) is a possible value and take actions to avoid potential memory access errors associated with accessing NULL values. Note that according to BSTR rules, NULL and L”” are semantically equal, and in case of CAPE-OPEN, are an UNDEFINED string.

Requirement 5: Use of whitespace in component names or descriptions aid readability, and should not be unnecessarily restricted. Both the first and last character of a component name or description must not be a blank character (white space). A component name must contain at least one non-whitespace character. The component description can be an empty (zero-length) string or a null string, depending upon middleware specifications. Control characters (carriage return, line feed, tab, escape, delete, form feed, etc…) must not be used in *ComponentName*.

Issue 6: Default component names and descriptions.

Discussion 6: The owner of the PMC is responsible for ensuring that the name of the PMC is unique. During initialization, a CAPE-OPEN component shall name itself with a default component name and description. Possible initial names include either the

PMC's COM registry *CapeDescription* key 'Name' named-value or the PMC's ProgID. This string can then be concatenated with an incremented static integer value to form a unique name. Alternatively, the PMC developer may use defined string can be used as a basis for the name, however, uniqueness is not assured. After this, the PMC owner is responsible for ensuring that the *ComponentName* remains unique within the context.

The component shall also provide a generic description for itself. This generic description can match the value in the *CapeDescription* registry key 'Description' named-value pair. As described above, a NULL or zero-length string is an acceptable description. There are no restrictions on modifying the *ComponentDescription* value.

Requirement 6: During initialization, a CAPE-OPEN component shall name itself with a default component name and description. The owner of the PMC is responsibility for name uniqueness, but the PMC should attempt to create a unique name for itself during instantiation.

Issue 7: Changes to the *ComponentName* during a call to *ICapeUtilities.Edit*.

Discussion 7: PMC's implementing the *ICapeUtilities.Edit* method typically allow changes to the *ComponentName* property of the PMC object. Changes to the *ComponentName* during edit may result in the *ComponentName* not being unique. Collection owners need to be aware of this possibility and verify that *ComponentName* changes during Edit do not result in loss of *ComponentName* uniqueness.

Requirement 7: The Collection owner managing a Collection of CAPE-OPEN objects will verify that the *ComponentName* of a Collection member is unique following a call to *ICapeUtilities.Edit*.

Issue 8: Ability to make calls on *ICapeIdentification* prior to initialization by calling *ICapeUtilities.Initialize*.

Discussion 8: All initialization steps that can fail are placed in the *ICapeUtilities.Initialize* step, which could include allocation of the string storage for the *ComponentName* and *ComponentDescription* fields. Prior to initialization of a PMC, there is no guarantee that the storage for the *ComponentName* or *ComponentDescription* has been allocated. For this reason, calls to *ICapeIdentification* are not supported until after the call to *ICapeUtilities.Initialize*.

Requirement 8: The *ComponentName* or *ComponentDescription* properties are not accessible on a PMC prior to a call to the *ICapeUtilities.Initialize* method on PMC Primary Object. The call to the *ICapeUtilities.Initialize* method will initialize the *ComponentName* and *ComponentDescription* properties for all PMC objects owned by the PMC Primary Object.

4. Bibliography

1. "Whitespace Character", Wikipedia, retrieved May 12, 2014. http://en.wikipedia.org/wiki/Whitespace_(computer_science)

2. "Identification Common Interface Specification" v3.0, CO-LaN, 2003.

3. Windows DevCenter – Desktop, BSTR, retrieved August 14, 2013. http://msdn.microsoft.com/en-us/library/windows/desktop/ms221069(v=vs.85).aspx
4. Microsoft Developers Network, "Character Sets," retrieved May 13, 2014. http://msdn.microsoft.com/en-us/library/windows/desktop/dd317743(v=vs.85).aspx
5. Lippert, Eric (2003). "Eric's Complete Guide To BSTR Semantics", retrieved May 12, 2014. http://blogs.msdn.com/b/ericlippert/archive/2003/09/12/52976.aspx?PageIndex=2#comments
6. Microsoft Developer Network, "BSTR Data Type," retrieved May 13, 2014. http://msdn.microsoft.com/en-us/library/ms221069.aspx