Methods and Tools
Special Interest Group Report
October 2021 – September 2022

Michael Hlavinka

Bryan Research & Engineering

21 September 2022

SIG Membership

Bill Barrett

US EPA

Michael Hlavinka

Bryan Research & Engineering, LLC

Jasper van Baten

AmsterCHEM

Mark Stijnman

Shell Global Solutions International B.V.

If you are interested in joining, please contact either SIG Leader or CTO

- **Bill Barrett – barrett.williamm@epa.gov**
- **Michel Pons - technologyofficer@colan.org**

M&T SIG Meetings

- Typically, four conference calls per month. Calls are held on Tuesdays at 1600 CET, 1000 US Eastern Time, 0900 US Central Time.

- Joint Conference call with Interop SIG, typically 3rd Wednesday of the month at 1730 CET, 1630 London Time, 1130 US Eastern Time, 1030 US Central Time.

M&T SIG Charter

- ❑ Improve integration and expand utilization of Computer-Aided Process Engineering (CAPE) applications within the enterprise through identification and resolution of existing cross-cutting issues with the CAPE-OPEN platform, develop mechanisms for use of CAPE within other application domains, and incorporate advances in information technology into the CAPE-OPEN platform.

- ❑ Key responsibilities
  - Resolve issues with the common interface specifications.
  - Develop and maintain standards and protocols for CAPE-OPEN implementations.
  - Incorporate advances in information technology into the CAPE-OPEN protocols.
  - Identify novel uses of CAPE and provide standards for utilizing CAPE within these applications.

No change to vision and responsibilities.

M&T SIG 2021/2022 Summary of Activities

- ❑ COBIA Maintenance, Development, and Testing
  - One maintenance release of COBIA Phase II
  - COBIA Phase III Work Package I Development
    - Generic TCP/IP Marshaler (completed)
    - Optimizing transport for specific marshaling scenarios (initiated)
    - On-the-fly proxy generation (initiated)
- ❑ Threading in CAPE-OPEN (more detail next presentation)
- ❑ Interaction with Interop SIG
  - Support for Certification Testing
  - Versioning
- ❑ Developing Test Specifications for Common Interface Specifications
  - Parameter
  - Utilities
  - Identification
  - Collection

COBIA Project Roadmap

- ❑ Phase I – Proof of Concept **Completed**
  - Core technical components
  - Demonstrate COM/COBIA interoperability with Thermo 1.1 interface set
- ❑ **Phase II Full Windows Native – Completed/In Use**
  - Expanding COBIA to all interfaces of business value
  - Support for C/C++ development.
  - Allow development of fully functional COBIA-based PMEs and PMCs
  - Maintenance – Ongoing
- ❑ **Phase III – Cross-Platform Interoperability**
  - WP I: Marshaling In-Progress
- ❑ **Phase IV – Documentation Future**

COBIA Timeline to Date

- ♦ October 2016 - Phase I (Proof of concept) completed
- ♦ October 2017 – Phase II (C/C++ Application Framework) status presented and demonstrated
- ♦ October 2018 – Early adopter's version of COBIA
  - COBIA Training
  - Testing, bug fixes and third-party use of COBIA
- ♦ April 2019 – Management Board support for KBC and HTRI development
- ♦ September 2020 – Phase II Public Release of COBIA 1.2.0 (SDK and runtime)
  - Suitable for used in commercial packages
- ♦ Spring/Summer 2021– The first commercial packages using COBIA were released by vendors.
- ♦ February 2021 – Start of Phase III.
- ♦ September 2021 –Generic marshaler source code made available to CO-LaN membership
  - No public release.
  - Will be released as part of a future SDK/runtime update – pending completing Thread Architecture.

❑ **September 2022: Proposal of COBIA threading architecture**

COBIA Phase II Maintenance

- ❑ CO-LaN actively supports COBIA by providing maintenance releases of COBIA of runtime and development kit.
  - 9 maintenance releases to date.
  - Resolved issues raised during development and use of commercial applications.
- ❑ COBIA actively used in released commercial applications.
  - As of today, more than 12 months of COBIA use in commercial applications.
  - Vendor feedback about COBIA has been positive.
    - Easier development of CAPE-OPEN software.
    - Very little maintenance of COBIA-based CAPE-OPEN implementations.
    - No end user issues related to COBIA.
  - Product Maturity
    - Only 1 Maintenance release has been required since October 2021.
    - March 2022: Version 1.2.0.9, included updating Symbol Server.
  - Latest version of SDK downloaded 93 times

Overview of COBIA Phase III

❑ Divided into four Work Packages:
- WP I: Marshaling
- WP II: Language Bindings
- WP III: Remote Computing
- WP IV: Logging

Work Package I (Marshaling) Update

- ❑ COBIA Marshaler for CAPE-OPEN Interfaces
  - Marshals between threads/processes/machines.
  - Uses TCP/IP transport mechanism
    - Most general case, used in-process, out-of-process, remotely.
    - Further remote machine developments (*e.g.*, encryption) under WP III.
  - Available for testing by interested CO-LaN members.
- ❑ Currently optimizing transport for specific marshaling scenarios
  - Shared Memory Transport Mechanism
    - Improves performance on local machine compared to TCP/IP.
    - Designed for out-of-process marshaling.
    - Uses synchronized access to shared memory block.
  - C++ Heap Memory Transport
    - Optimized synchronization of in-process applications between threads.
- ❑ On-the-fly proxy generation (successfully tested using libffi package)
  - Allows automated creation of proxy classes.
  - Enables custom (non-CAPE-OPEN) interfaces, COM interop, and language bindings.

COBIA Threading Architecture

❑ Threading Architecture required to finalize WP I

❑ Considerations:

- Modern/flexible approach to multi-threaded applications.
- Establish responsibility for synchronization.
- Minimize need to marshal calls.
- Interoperable with COM

❑ Threading Options

- Default Threading
- Restricted Threading

❑ More details in next presentation.

COBIA Phase III Language Bindings (WP II)

❑ CO-LaN members were surveyed in 2020 regarding prioritization of language bindings.

❑ Candidate programming languages (Votes from survey)

- C Application Binary Interface (ABI) (9)
- Python (15)
- FORTRAN (10)
- Microsoft .NET (5)

❑ Language binding development is contingent upon software vendors stating a desire to Management Board to develop tools using that language. Please provide feedback!

COBIA Phase III Remote Computing and Logging Work Packages

❑ WP III: Remote Computing Work Package

- Extends the WP I generic TCP/IP marshaler to access objects on remote computers either within an enterprise network or cloud computing environments.
- Feature development will be guided by expressed needs.

❑ WP IV: Logging Work Package

- Allows logging calls over the COBIA pipeline.
- Enables CO-LaN to develop a logging application that can trace calls between client and server.

❑ WP III and WP IV can be initiated once WP I is completed.

Test Specification Developments

❑ M&T SIG developing Test Cases for the Common Interfaces.

- M&T SIG owns the Common Interfaces and is responsible for developing test cases for them.
- These test cases serve as examples for other SIGs to develop Test Cases for their interfaces.

❑ Test Cases

- Test case format is based upon interface use case format.
- Test cases are based on interface specification requirements, use cases, and method descriptions.
- Test checks for correct return value types, where applicable.
- Test cases include ensuring that the object supports required interfaces.

Identification Test Cases

- ❑ IDENTIFICATION 1::ICapeIdentification_Existence
- ❑ IDENTIFICATION 2::ComponentNameRetrieval
- ❑ IDENTIFICATION 3::ProvidesDefaultComponentName
- ❑ IDENTIFICATION 4::UseOfWhitespaceInDefaultComponentName
- ❑ IDENTIFICATION 5::UseOfControCharactersInDefaultComponentName
- ❑ IDENTIFICATION 6::PutComponentName
- ❑ IDENTIFICATION 7::ComponentDescriptionRetrieval
- ❑ IDENTIFICATION 8::UseOfWhiteSpaceInDefaultComponentDescription
- ❑ IDENTIFICATION 9:: UseOfControCharactersInDefaultComponentDescription
- ❑ IDENTIFICATION 10::PutComponentDescription

♦ Issue: Put_ComponentName should not be implemented on secondary objects such as collection elements.

Collection Tests

- COLLECTION 1::ICapeCollection_Existence
- COLLECTION 2::CountRetrieval
- COLLECTION 3::CollectionOneIndexed
- COLLECTION 4::ItemRetrievalByInteger-ValuedIndex
- COLLECTION 5::ItemNameUniqueness
- COLLECTION 6::ItemRetrievalByString-ValuedIndex

Example of Test Case Execution Identification 4::UseOfWhitespaceInDefaultComponentName

1. The Test obtains the Component-under-Test from the Test Context.

2. The Test queries the Component-under-Test for the ICapeIdentification interface. If no ICapeIdentification interface pointer to the Component-under-Test is obtained by the Test, the Test logs an ERROR with the following Error Message and execution stops here.

   Message: “[Component-under-Test] does not expose ICapeIdentification.”

3. The Test obtains the ComponentName from the Component-under-Test by exercising ICapeIdentification::getComponentName, with error handling. If no ComponentName value is obtained, the Test logs an ERROR with the following Error Message and execution skips to Step 5:

   Error Message: “Obtaining ICapeIdentification::ComponentName from [Component-under-Test] raised [EXCEPTION].”

4. If the ComponentName value begins or ends with white space values (space, non-breaking space), the Test reports a TEST FAILURE with the following Expected Result, Actual Result, and Additional Information:

   Expected Result: “ComponentName value does not begin or end with white space.”

   Actual Result: “ComponentName value begins or ends with white space characters.”

   Additional Information: “Obtaining ICapeIdentification::ComponentName from [Component-under-Test] results in a value that begins or ends with white space”

5. The Test releases the interface pointer to the ICapeIdentification interface implemented on the Component-under-Test.

Anticipated 2022/2023 Activities

- ❑ Maintenance of COBIA Phase II
- ❑ Complete Documentation of COBIA Threading Architecture (RFC)
- ❑ COBIA Phase III Work Packages
  - Release of COBIA version 1.2.1 with Marshaling and Multi-Threading Support
- ❑ Develop Test Case Specifications for Common Interfaces
- ❑ Interface Specifications
  - CAPE-OPEN Native Persistence Interface Specifications (CAPE-OPEN Version 1.2+)
  - Rewrite Parameters Interface Specification document (CAPE-OPEN Version 1.2+)
  - Incorporate Errata and Clarifications into Identification Interface Specification (CAPE-OPEN Version 1.1+)
  - Finalize Reporting and Managers (CAPE-OPEN Version 1.1+)
- ❑ Work with Interop SIG
  - Advise on Certification Tools
  - Versioning documentation

Thank you For Your Attention

- Any Questions?