



***Development of CAPE-OPEN
Compliant Process Modeling
Components in Microsoft .NET***

William M. Barrett, Jr PhD, PE
Engineer

CAPE-OPEN Laboratories Network Interoperability Showcase
23 February 2006

Current CAPE-OPEN MiddleWare Specifications

- **Component Object Model (COM)**
 - Primarily Windows-Based
 - An outgrowth of Object Linking and Embedding (OLE)
 - Active X Controls
- **Common Object Request Broker Architecture (CORBA)**
 - More Common on UNIX Systems
 - Different Object Brokers May Be Incompatible

What is .NET

- .NET is an “update” of Microsoft’s COM
- Open Standards:
 - Ecma-334*: C# Specification
 - Ecma-335*: CLI Specification – Virtual Machine and XML-based class library
- Unix-based Open Source Implementation - Mono

How Does .NET Differ from COM?

COM

- DLL Hell
- Reliance on System Registry
- HRESULTS

.NET

- Versioning
- Side-by-Side Execution
- XCopy Deployment
- Security
- Platform Neutrality
- Strongly Typed (CTS)
- XML/Web Services
- Garbage Collection
- Structured Exceptions

COM/.NET Interoperation

- “It Just Works” (In General, Mostly, But In Practice, CAPE-OPEN Does IJW)
- Creating A Primary Interop Assembly – Strongly Named Types In The PIA
- Using PIAs - Creates A “Native” .NET Version Of The CAPE-OPEN Interfaces That May Not Be Quite Exactly The Way To Do It.
- Export COM Type Library From .NET
- Comparison of The Two Interface Definitions

Comparison Of Data Types

CAPE-OPEN Data Type	Description of Data Type	COM Data Type	.NET Data Type
CapeLong	long	long	long
CapeShort	short	short	short
CapeDouble	double	double	double
CapeFloat	float	float	float
CapeBoolean	boolean	VARIANT_BOOL	bool
CapeChar	char	BYTE	char
CapeString	String	BSTR	String
CapeDate	string date	VT_DATE	DateTime
CapeURL	URL string	BSTR	String
CapeVariant	Container of any other type	VARIANT	Object
CapeInterface	CO Interface	LPDISPATCH	Object
CapeArray(TYPE)	Array of Type	VARIANT containing Safearray(Type)	Array<Type> downcast to an Object

CAPE-OPEN Types

- CAPE-OPEN defines a number of enumeration types such as the CapeValidationStatus enum
- Interfaces Are Also Types
- Creating a PIA results in creating a namespace (e.g. CAPEOPEN093 from CAPE-OPENv0-9-3.tlb)
- CAPEOPEN093.CapeValidationStatus *IS NOT THE SAME AS* CAPEOPEN100.CapeValidationStatus
- CAPEOPEN093.ICapeParameter *IS NOT THE SAME AS* CAPEOPEN100.ICapeParameter
Even Though the GUIDs Are The Same (Same COM Type)

Comparison of Interfaces

C++/CLI Interface Class

```
//This interface provides the basic functionality for a Unit
//Operation component
// CAPE-OPEN v1.0
[
    ComVisibleAttribute(true),
    Guid("678c0998-0100-11d2-a67d-00105a42887f"),//ICapeUnit_IID,
    System::ComponentModel::DescriptionAttribute("ICapeUnit Interface")
]
public interface class ICapeUnit
{
    // Get the collection of unit operation ports
    //
    // CAPE-OPEN exceptions:
    // ECapeUnknown, ECapeFailedInitialisation, ECapeBadInvOrder
    [DispIdAttribute(1), System::ComponentModel::DescriptionAttribute("Gets the whole list of ports")]
    property Object^ ports
    {
        [returnvalue: MarshalAs(UnmanagedType::IDispatch)]
        Object^ get();
    };

    // Gets the flag to indicate unit's validation status
    // notValidated(0),invalid(1) or valid(2)
    //
    // CAPE-OPEN exceptions
    // ECapeUnknown, ECapeInvalidArgument
    [DispIdAttribute(2), System::ComponentModel::DescriptionAttribute("Get the unit's validation status")]
    property CapeValidationStatus ValStatus
    {
        CapeValidationStatus get();
    };

    // Executes the necessary calculations involved in the unit
    // operation model
    //
    // CAPE-OPEN exceptions raised:
    // ECapeUnknown, ECapeBadInvOrder, ECapeOutOfResources, ECapeTimeOut,
    // ECapeSolvingError, ECapeLicenceError
    [DispIdAttribute(3), System::ComponentModel::DescriptionAttribute("Performs unit calculations")]
    void Calculate();
    // Validate that the parameters and ports are all valid
    //
    // CAPE-OPEN exceptions:
    // ECapeUnknown, ECapeBadCOPParameter, ECapeBadInvOrder
    [DispIdAttribute(4), System::ComponentModel::DescriptionAttribute("Validate the Unit"), returnvalue: MarshalAs(UnmanagedType::VariantBool)]
    bool Validate(String^ %message); // C# bool Validate (ref string message)
};
```

COM IDL

```
[
    object,
    uuid(ICapeUnit_IID),
    dual,
    helpstring("ICapeUnit Interface"),
    pointer_default(unique)
]
interface ICapeUnit : IDispatch
{
    // Get the collection of unit operation ports
    //
    // CAPE-OPEN exceptions:
    // ECapeUnknown, ECapeFailedInitialisation, ECapeBadInvOrder
    [propget, id(1), helpstring("Gets the whole list of ports")]
    HRESULT ports([out, retval] CapeInterface* ports);

    // Gets the flag to indicate unit's validation status
    // notValidated(0),invalid(1) or valid(2)
    //
    // CAPE-OPEN exceptions
    // ECapeUnknown, ECapeInvalidArgument
    [propget, id(2), helpstring("Get the unit's validation status")]
    HRESULT ValStatus([out, retval] CapeValidationStatus *valStatus);

    // Executes the necessary calculations involved in the unit
    // operation model
    //
    // CAPE-OPEN exceptions raised:
    // ECapeUnknown, ECapeBadInvOrder, ECapeOutOfResources, ECapeTimeOut,
    // ECapeSolvingError, ECapeLicenceError
    [id(3), helpstring("Performs unit calculations")]
    HRESULT Calculate();

    // Validate that the parameters and ports are all valid
    //
    // CAPE-OPEN exceptions:
    // ECapeUnknown, ECapeBadCOPParameter, ECapeBadInvOrder
    [id(4), helpstring("Validate the Unit")]
    HRESULT Validate([ACTUALLYout] CapeString* message, [out, retval] CapeBoolean* isValid);
};
```

Excerpt of Unit Operation Base Class Implementation

```
[Serializable]
[ComVisibleAttribute(true)]
public ref class CCapeUnit abstract :
    public CCapeObject,
    public ICapeUnit,
{
    CPortCollection^ m_ports;
    CapeValidationStatus m_valStatus;
public:
    CCapeUnit();
    {
        this->m_ports = gnew CapeOpen::CPortCollection();
        m_valStatus = CapeValidationStatus::CAPE_NOT_VALIDATED;
    };

~CCapeUnit()
{
    delete m_ports;
}

[ComRegisterFunctionAttribute]
static void RegisterFunction(Type^ t)
{
    String^ keyname = String::Concat(L"CLSID\\{", t->GUID.ToString(), L"}\\Implemented Categories");
    Microsoft::Win32::RegistryKey^ key = Microsoft::Win32::Registry::ClassesRoot->OpenSubKey(keyname, true);
    key->CreateSubKey("{678c09a5-7d66-11d2-a67d-00105a42887f}"); // Cape Unit Operations
    key->CreateSubKey("{678C09A1-7D66-11D2-A67D-00105A42887F}"); // Cape Object
    key->Close();
}

[ComUnregisterFunctionAttribute]
static void UnregisterFunction(Type^ t)
{
    String^ keyname = String::Concat(L"CLSID\\{", t->GUID.ToString(), L"}\\Implemented Categories\\{678c09a5-7d66-11d2-a67d-00105a42887f}");
    Microsoft::Win32::Registry::ClassesRoot->DeleteSubKey(keyname);
    keyname = String::Concat(L"CLSID\\{", t->GUID.ToString(), L"}\\Implemented Categories\\{678C09A1-7D66-11D2-A67D-00105A42887F}");
    Microsoft::Win32::Registry::ClassesRoot->DeleteSubKey(keyname);
}

// ICapeUnit Methods
virtual void Calculate(void) = 0;

[returnvalue: MarshalAs(UnmanagedType::VariantBool)] virtual bool Validate(String^ (%message));

virtual property Object^ ports
{
    [returnvalue: MarshalAsAttribute(UnmanagedType::IDispatch)]
    virtual Object^ get (void)
    {
        return this->m_ports;
    }
}

property CapeValidationStatus ValStatus
{
    virtual CapeValidationStatus get (void)
    {
        return m_valStatus;
    }
}
};
```

DEMONSTRATION

Create Unit Operation in C#
Run CAPE-OPEN Tester on it