Generic Interfaces

This paper describes a technique currently named “Generic Interfaces”. It allows calling COM VTable Objects without having to define types in VisualWorks. The advantages are clear: No Interface- and InterfacePointer classes need be implemented or no CTypes defined in ExternalInterfaces.

Nevertheless, Generic Interfaces is still in development and may change until it is released.

Loading

In order to activate the support for Generic Interfaces, please load the parcel “Com-GenericInterfaces”. The parcel will be found in the equally-named subdirectory of VisualWorks’ preview folder.

HowTo Use

Usage is quite simple. First the developer has to declare which typelibrary will be used.

```
COMTypeLibrary using: <aGUID>.
```

Here ***aGUID*** is the type library id. The easiest way to retrieve this library is to lookup the library in the Automation Browser and copy the guid from the tools pane. After that, VisualWorks knows all types which are declared inside this type library.

The class COMClient

Now a new object may be created in the following syntax equal to the one used by COMDispatchDriver:

```
app := COMClient createObject: ‘Application.Class’.
```

Of course, in addition to the registered class name the class id maybe used to create the object.

After that, all methods of the application may be called. Thereby the selector is built from the name of the method and a number of arbitrary keywords. The number of keywords depends on the number of method arguments.

Assume the following method is called:

```
INT app.MyTestMethod(INT index, BSTR*  name, IUnknown* parent);
```

In Smalltalk, valid calls to the method could look like this:

```
app MyTestMethod: index with: name with: parent.
app MyTestMethod: index name: name parent: parent.
app MyTestmethod: index _: name _: parent
```

Although all listed sends will call the same COM method, for the Smalltalk environment the selectors are still different. So, although the implementation does not force a specific selector scheme, it is generally recommendable to follow only one in order to be able to find calls to the same COM method again.

Please also note that the case of the COM function definition is kept for the method name. Therefore the first letter of the selector may be uppercase even though this is uncommon in Smalltalk.

When you are finished using the object, be sure to release it by sending the #release message .

The IAnonymous Interface

Usually it is not possible to determine the class of a method return value. Therefore COM methods will usually return interfaces instead of ***COMClients***. As an aid to the developer the ***IAnonymous*** interface was extended to act as a generic proxy. This allows the developer freedom to not distinguish between working with classes or interfaces. Both act as proxy and delegate messages to the COM Server Object, both supporting the same syntax, and both need to be released when they are no longer used.

Extended Datatype support

GenericInterfaces utilizes and extends ***Userdefined Datatype Support*** such that all COM datatypes including enumerations and structures are supported.

Structures may - as determined in ***Userdefined Datatype Support -*** be passed as Dictionaries or classes that are bound to types. See the ***Userdefined Datatype Support*** documentation for more information about binding.

Enumeration values may be provided as numbers or symbols. In addition, a combination of enumeration values may be passed as array of symbols. In this case, the values may be combined using ***bitOr:*** :

```
anObject callMethodWith: #EnumerationValueOne.
anObject callMethodWith: #(#EnumerationFlagOne #EnumerationFlagEight
#EnumerationFlag512).
```

As in Automation, reference Values (Pointers) need to be passed as ValueReferences in order to be able to update them. Usually it is enough to send ***#asValue*** to achieve this.

Hints

Currently Tools support has not yet been extended to show methods supported by non-automation COM objects. Nevertheless it is possible to get information about the features of a class or an interface:

Each ***COMClient*** and ***IAnonymous*** instance holds a ***SpecificationTable*** which describes all of its features in detail.