Userdefined Datatype Support for COM Connect

Introduction

Using Com- UDT Support Extension it is now possible to call COM functions which use structure parameters. This document describes how to use this new functionality.

Loading the extension

The extension is provided in parcel “Com- UDT Support.pcl” which can be found in the UDT Support subdirectory of the VisualWorks preview folder.

Using structures with Automation calls

The COMDispatchDriver class facilitates using structure parameters in Automation calls. In its simplest case, a ValueReference on a Dictionary can be passed describing your structure. The structure elements to which values are assigned are identified by dictionary keys. Custom classes instead of dictionaries are also supported. To transfer Smalltalk values into a COMRecord using #memberAt:put it is enough to implement the method #putValuesIntoCOMRecord:.

Currently COMRecords are released explicitly like all other COM Structures.
All BYREF parameters and structure return-values as used in COMRecords reference COM memory and need to be released when their use is completed. If you forget to release these structures a dialog will open warning that the COMRecord and an IRecordPointer were released by finalization.

Automation DataType support

The Automation Types Variant and SafeArray have been extended to support COMRecords.

For COMVariantStructure and subclasses, there is a new instance creation method named *typeDescription:value:* which allows passing type descriptions instead of only typecodes that do not provide enough information for creating and managing structure variants.

SafeArrays of Records may be created by using newly added methods *fromCollection:type:* and *new:type:*.

The *typeNamed:* method defined in *COMDispatchDriver* may be used to acquire a record data type used to create a variant or a SafeArray.

Accessing records in Variants and SafeArrays works like accessing any other value, except that it will provide an instance of *COMRecord*. This COMRecord must be released manually. COMRecords are discussed in the following section.

The COMRecord class

COMRecord is a class which manages COM structures. They manage allocated memory and provide operations for accessing members.

Instance creation

A COMRecord is created by sending a *new* message to a *DispStructureDescription* instance. In turn, a *DispStructureDescription* instance can be acquired by sending the *typenamed:* message to an COMDispatchDriver.

```
app := AdvancedDispatchDriver createObject: 'MyApp.Application'.
myType := myAppObject typeNamed: #myType.
app release.
```

Member accessing

For accessing members of a *COMRecord*, a set of methods similar to DLLCC Syntax is provided: *memberAt:*, *memberAt:put:* and *refMemberAt:*.

memberAt: returns a copy of the value at the given name. The object returned by *memberAt:* will not provide write access to elements of a Record. Acquiring references to record elements is possible by using *refMemberAt:*.

memberAt:put: - assigns a value into the member. This is also capable of setting nested structure values.

refMemberAt: - returns a variant which holds a reference to the element with the given name – modifying the variant will modify the structure. For accessing structures in such variants, acquire the value of such a variant and modify the structure elements.

Nested member access

The accessing methods *memberAt:, memberAt:put:* and *refMemberAt:* also provide direct access to nested elements using C like syntax:

```
aStructure memberAt: '<element1>.<element2>'
```

Here <element1> is the name of a top-level element of the structure and <element2> is the name of an element of the substructure named <element1>.

Accessing members of array structure elements works in the same manner, except of course that it is required to provide the index of the array element to be accessed.

```
aStructure memberAt: 'element1[10].element2'
```

Conversion support

COMRecords can be converted to common Smalltalk objects by sending the *asSTObject* message. The result of this operation depends on whether the specific record type has been bound to a Smalltalk Class before. If the type has been bound to a class, the result will be an instance of the specified class

Binding record datatypes

Binding Record data types to Smalltalk classes can be achieved by sending the *bindTypeTo:* message to a record type or by sending the *bindTo:* message to its type. A type can be acquired from the typelibrary by sending the *typeNamed:* message to an *COMDispatchDriver*.

It is possible to customize conversion behavior by implementing the class method *fromUDTObject:ofType:*. This will override the default implementation in *Object*.