Base Language - Feature #3262

implement COM/OLE Automation support

03/10/2017 09:43 AM - Greg Shah

Status:	Closed	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:		**% Done:**	100%
Category:		**Estimated time:**	0.00 hour
Target version:			
billable:	No	**version:**	
vendor_id:	GCD		

Description

Related issues:

Related to User Interface - Feature #1818: implement a timer service to repla...	**Closed**
Related to Base Language - Feature #3505: implement CONNECT TO support for CR...	**New**
Related to Base Language - Feature #3506: implement array support for COM aut...	**New**
Related to User Interface - Feature #3858: implement a widget that replaces t...	**Closed**

History

#1 - 03/10/2017 09:46 AM - Greg Shah

COM/OLE Automation is the remote access of properties and methods in COM objects, which reside in another process. The Windows Common Object Model (COM) and its APIs are used to handle this remote access. Since there is no integration with the visual UI (layout, z-order, drawing, events...), the scope of supporting this is much less than OCX support. We have already implemented a similar approach for native API call support (via NativeInvoker + libp2j + libffi).

The core of the ABL syntax is like this:

COM-HANDLE
CREATE OBJECT com-object-type com-handle
com-handle:PROPERTY = abl_type
abl_type = com-handle:PROPERTY
abl_type = com-handle:METHOD
com-handle:<property_or_method>:<chained_property_of_method>:<more_chaining>
RELEASE OBJECT com-handle

The abl_types have marshalling/unmarshalling to COM-specific native types, similar to the native API calls. I already understand the basic idea there.

Questions

1. What WIN32 API is used for each of the above items?

2. How tightly coupled is the definition of a property or method to the COM object type being accessed? For native API calls, we dynamically load a DLL and lookup a function pointer to match the name of the API provided in the 4GL. Then we translate the ABL types to the native types and call the function pointer through libffi.

#2 - 03/10/2017 09:47 AM - Greg Shah

From Constantin:

> I found an interesting article here: https://notendur.hi.is/snorri/SDK-docs/getstart/javac003.htm Basically, COM objects can be used from Java with the help of the javatlb tool (from Microsoft, https://msdn.microsoft.com/en-us/library/windows/desktop/ms691398(v=vs.85).aspx). The link above also presents type mappings between COM and Java. What I'm not sure is how much of the details are about Visual J++ and not Oracle Java.
>
> Another interesting article is this: https://docs.oracle.com/cd/E13222_01/wls/docs61/jcomreference/UsingCOMObjectsfromJava.html - looks like Oracle via the WebLogic Server and com2java tool has a similar way to use COM objects from Java. And I think a more complete type mapping is here: https://docs.oracle.com/cd/E13222_01/wls/docs61/jcomreference/SupportedActiveXdatatypes.html#1104695 At the end it has also type conversion details.

#3 - 03/10/2017 09:48 AM - Greg Shah

From Constantin:

> Some other notes related to conversion: I expect com-handle vars to be treated the same way as we treat handle vars - that means, the COM object reference can be passed from a var to another via assignment, so we can't know at runtime which COM object is referenced via a var.
>
> For conversion, I would suggest having APIs like:
>
> 1. com-handle:PROPERTY = abl_type converted to comHandle.set("PROPERTY", abl_type)
> 2. abl_type = com-handle:PROPERTY converted to abl_type.assign(comHandle.get("PROPERTY"))
> 3. abl_type = com-handle:METHOD converted to abl_type.assign(comHandle.invoke("METHOD", abl_type1, abl_type2, ...))

#4 - 03/10/2017 09:49 AM - Greg Shah

From Hynek:

> there is a really good and relatively short article on codeproject.com that explains the COM basics: https://www.codeproject.com/Articles/13601/COM-in-plain-C, I wouldn't explain it better myself. I am attaching a C code sample showing how one interacts with the COM objects (and what Win32 API functions are used), the sample is taken from the same article.
>
> Besides the COM basics (in-memory vtable layout of COM interfaces, type marshaling, calling conventions of the interface methods, reference counting, etc.) there are other areas that need to be addressed in order to be able to consume the "Automation" COM objects (AKA OLE or ActiveX). Especially remoting and the relevant automation interfaces.
>
> The automation interfaces allow to talk to the COM objects without knowing the type information of the COM interfaces beforehand by using the IDispatch COM interface. IDispatch provides the information of the methods and properties, of the COM objects which can be retrieved on runtime and also allows invocation for these. There are also so called type libraries which provide similar information as IDispatch but are used during compilation (for example to build the remoting proxies and stubs). I am not sure whether Progress uses type libraries on the fly when creating the COM objects or whether it just utilizes the IDispatch interface, I would guess the latter.
>
> There do exist ready to use Java libraries that could be used to consume COM objects. Some use JNDI for this, some do it natively in Java. j-Interop looks particularly interesting (http://j-interop.dimentrix.com/documentation.html).

#5 - 03/10/2017 09:50 AM - Greg Shah

From Eugenie:

> This is a great readings. The native code we can use is the almost one we do at this time. Plus additional COM specific actions - request component with specific already known ID, let it know we are using it and then release. I think we will not need additional Java based COM bridge.
>
> I think if we need to implement our own COM component - this will be one more DLL with known rules of interactions with P2J.DLL. And it must be registered in Windows registry.
>
> Also this will be Windows specific component, looks like Linux is missing with this functionality.

#6 - 03/10/2017 09:50 AM - Greg Shah

From Sergey:

> There is JACOB project https://sourceforge.net/p/jacob-project/wiki/Home/ that can help to work with Windows COM API.

#7 - 03/10/2017 11:57 AM - Greg Shah

My initial thought that the COM object was in a separate process was only partially true. It can be a separate process, if the COM object is designed as an EXE but if it is a DLL, then the usage is in-process.

The access is definitely dynamic:

- The ABL doesn't have a built-in native compiler. Its "compiler" really just tokenizes the ABL code and encodes it as "r-code". To implement using the native interfaces that bind statically to the COM methods/properties, it would have to do a native compile.
- There is no type checking at ABL compile time, only at runtime.
- Event the way the COM/OLE Automation object is specified is by using a string (name). If it was non-dynamic it would use a GUID.
- The docs even suggest it is dynamic.

The following is a starting point for how to call the IDispatch APIs to dynamically access COM methods or properties:

https://msdn.microsoft.com/en-us/library/windows/desktop/ms221694(v=vs.85).aspx

This is a good reference to create a COM object that allows this dynamic access:

https://www.codeproject.com/Articles/13862/COM-in-plain-C-Part

#8 - 03/10/2017 11:59 AM - Greg Shah

The full design will require careful study of two chapters in "OpenEdge Development: Programming Interfaces" book (the "Using COM Objects in ABL" and "ActiveX Automation Support"). In addition, read the language reference for the data types, CREATE OBJECT, RELEASE OBJECT and the method/property access syntax.

#9 - 06/09/2017 01:12 PM - Ovidiu Maxiniuc

- Related to Feature #1818: implement a timer service to replace the PSTimer OCX added

#10 - 09/01/2017 09:02 AM - Greg Shah

Eugenie: The following attempts to summarize our current status and give you guidance on the next steps.

When 1818a was merged to trunk as revision 11161, a first pass of the conversion infrastructure was put into place for COM Automation.

See some notes in #1818-4 and #3262-3. The final result was a little different than both approaches but it is close in concept. The key usage is in the com.goldencode.p2j.util.comhandle class. It stores the Java representation of the COM object and provides methods to use that object. Look at the following methods:

- call() - execute a method, possibly with parameters and return the result
- chain() - access a property of the COM object and return it as a comhandle (to which you can chain other access)
- chainCall() - execute a method, possibly with parameters and return the result as a comhandle (to which you can chain other access)
- getProperty() - return the value of the property of the given name
- setProperty() - change the value of the property of the given name

This is the external interface used by converted 4GL code.

There is a kind of internal interface to a com.goldencode.p2j.comauto.ComObject instance that is the Java base class for the representation of the actual COM object. It uses getComAccessor() to lookup getters/setters and methods in the ComObject using Java reflection. This won't work for the generic COM support. We need to allow COM automation usage without writing a dedicated class for each WIN32 COM object that will be used. We do have the com.goldencode.p2j.extension.PSTimer class that is a manually written replacement for the Progress PSTimer OCX. We need to make sure that any changes to the internal interface of COM automation will not break the PSTimer. But the exact approach for this manually written replacement can't be used for generic COM automation support.

You will write native code in p2j.dll to provide the low level access in WIN32 to COM automation. The com.goldencode.p2j.comauto.ComServer is used for CREATE OBJECT and RELEASE OBJECT. You will have to provide backing function that uses native code in the client to initialize/terminate the COM automation and instantiate/destroy COM objects. Then we need to provide some mechanism to dynamically access the COM object's methods and properties and map them back into ComObject.

You will have to carefully consider how we can access this generically (no dedicated Java class for each COM object).

I think there will also be some testing code needed to determine exactly how the WIN32 data types get converted to/from 4GL data types.

The converted 4GL code will run on the server but the real WIN32 COM object will be "remote" on the FWD Java client, accessed via JNI.

If you can't find a COM automation object that has suitable range of features (and data types) for testing, you can write one. I did this in #1634 (see testcases/uast/library_calls/) and it made it much easier to check all the different parameter/no parameter/return cases and all the data type processing. You should look at that code and check the 4GL docs to get ideas about the data type mappings. Then you must surely test each of the combinations to make sure the docs are right. As we usually find, the 4GL docs are sometimes incorrect and always incomplete.

Post your questions here and we will discuss it.

#11 - 09/06/2017 08:57 PM - Eugenie Lyzenko

Continue investigation the background theory. The questions at this time:

1. The COM object we need to integrate are COM Automation objects and ActiveX controls. Correct?
2. All real work for COM objects(init, get/set property, call method, release) will be performed with actual COM object in native code with getting the return values back to java code. Correct?
2.1. If we will execute native COM object do we need it to be run in separate dedicated address space(separate process).
2.2. What about COM objects that has GUI output? Do we need native Win32 GUI component to coexist with FWD simulations?
3. All COM object instantiating/getting handle will be performed using String based COM object name. Correct?
4. The goal can be considered as having a set of JNI to use java COM related code as entry points to native functionality? Or may be we need to implement more deep replacement for COM leaving the only state change to JNI?

#12 - 09/07/2017 06:45 AM - Greg Shah

> 1. The COM object we need to integrate are COM Automation objects

Yes.

> and ActiveX controls. Correct?

No. Active-X controls are visual. We are NOT going to support visual controls. If the Active-X control can also be used as a COM Automation object (non-visual and non-interactive), then we will support it.

> 2. All real work for COM objects(init, get/set property, call method, release) will be performed with actual COM object in native code with getting the return values back to java code. Correct?

Yes. And also we must pass Java parameters down to the setters and method calls as needed.

> 2.1. If we will execute native COM object do we need it to be run in separate dedicated address space(separate process).

No. It should be inside the FWD client JVM process.

> 2.2. What about COM objects that has GUI output? Do we need native Win32 GUI component to coexist with FWD simulations?

No.

> 3. All COM object instantiating/getting handle will be performed using String based COM object name. Correct?

Yes.

> 4. The goal can be considered as having a set of JNI to use java COM related code as entry points to native functionality?

Yes.

> Or may be we need to implement more deep replacement for COM leaving the only state change to JNI?

No, we are mapping the converted 4GL constructs into the real COM implementation on Windows.

#13 - 09/07/2017 11:12 PM - Eugenie Lyzenko

So far the generic plan to access COM Automation objects(without all required details for now):

1. Starting from server side(Java) by call ComServer.create("String COM Id"). The class will check if it already has mapped ComObject internally.
2. If the object is not yet mapped the ComServer will call to FWD client first to initialize object by given name and get back the pointer to IDispatch interface. The FWD client makes the JNI calls to perform init steps.
3. Every ComObject going to use can be either artificial(like PSTimer) or native generic, native generic objects has long type member to store the value of the long pointer to IDispatch interface.
4. Then new comhandle need to be associated with obtained interface to be able to work from Java using handle class.
5. Both artificial manual objects and generic native objects will extend the same ComObject to provide Java interface level compatibility for each type.
6. When it is required to work with any COM object the ComServer gets if from internal map(assumed the init stage is completed) and depending on if it is manually written or native performs respective action(either making pure Java work for manually written or asking the client to make the rest work passing the value of the IDispatch long pointer as the first mandatory parameter for JNI)
7. On completion all native COM object must be released making appropriate call to client side JNI.

The native code is planning to leverage the Win32 API CLSIDFromProgID(), CoCreateInstance(), QueryInterface() to obtain pointer to IDispatch from given String based object ID and Invoke() to call methods and gettesr/setters.

This is very base plan, more details will be added further after additional investigation(how to invoke particular methods in native COM, how to set/get properties, how to handle method parameters and return values). Continue working.

Also need to check/verify required 4GL data mapping for COM<->4GL.

#14 - 09/08/2017 04:22 AM - Ovidiu Maxiniuc

Sounds like a plan.

You need to query the OS or read from the registry the actual list of available "String COM Id"-s. The ComServer's internal map can only be populated with java written automations (like PSTimer). Probably you need to create two separate lists, one with top-level and one with lower level or create a structure containing this piece of data, too. Please see the "CREATE automation object statement" in ABL Reference. It is partially implemented in ComServer, but since there are no implementation of native calls I could not foresee the exact flow and connections.

#15 - 09/08/2017 08:08 AM - Eugenie Lyzenko

Ovidiu Maxiniuc wrote:

> ... Probably you need to create two separate lists, one with top-level and one with lower level or create a structure containing this piece of data, too.

I see. On the other hand the map in ComServer holds the Object so can take any type of objects. Some of them can be the manually written like PSTimer while other can have native nature. It will be up to ComServer logic to choose respective handling for different objects.

#16 - 09/08/2017 08:48 AM - Greg Shah

> It will be up to ComServer logic to choose respective handling for different objects.

If possible, it is best to hide the differences in behavior behind a common interface. Then the ComServer can just use the interface and doesn't need to know the differences.

#17 - 09/08/2017 01:34 PM - Eugenie Lyzenko

Greg Shah wrote:

> Eugenie: The following attempts to summarize our current status and give you guidance on the next steps.

Greg,

I'm investigating the data type matching processing details and need a clarification for:

> If you can't find a COM automation object that has suitable range of features (and data types) for testing, you can write one. I did this in #1634 (see testcases/uast/library_calls/) and it made it much easier to check all the different parameter/no parameter/return cases and all the data type processing. You should look at that code and check the 4GL docs to get ideas about the data type mappings. Then you must surely test each of the combinations to make sure the docs are right. As we usually find, the 4GL docs are sometimes incorrect and always incomplete.

Do you mean writing Win32 DLL that serves Java calls in #1634? As general consideration I'm looking for existed COM objects and inclining to conclusion the best way will be to write and register our own native COM object for Win32 to completely test/control data flow we need to check. This is an idea you mentioned?

Can you please point me the some example in testcases/uast/library_calls/ to start from? May be something in testcases/uast/library_calls/input_output or may be another?

#18 - 09/08/2017 01:49 PM - Greg Shah

> Do you mean writing Win32 DLL that serves Java calls in #1634?

No. I mean writing a COM object that allows us to write 4GL code to test how the COM support works. Then we would convert that 4GL code and use the support written in this task to use the same COM object from FWD.

> As general consideration I'm looking for existed COM objects and inclining to conclusion the best way will be to write and register our own native COM object for Win32 to completely test/control data flow we need to check. This is an idea you mentioned?

Yes.

> Can you please point me the some example in testcases/uast/library_calls/ to start from? May be something in testcases/uast/library_calls/input_output or may be another?

Start reading the 4GL code using library_calls/test_runner.p. The native code is in library_calls/testapi/. The details on how to build and run the tests are in library_calls/readme.txt.

#19 - 09/08/2017 02:21 PM - Eugenie Lyzenko

Greg Shah wrote:

> > Do you mean writing Win32 DLL that serves Java calls in #1634?
>
> No. I mean writing a COM object that allows us to write 4GL code to test how the COM support works. Then we would convert that 4GL code and use the support written in this task to use the same COM object from FWD.

Yes, I've made confused question. I meant like you have written native DLL component to test #1634 I need to write native COM object to work with 4GL testcases to check conversion/runtime new features. Looks like I have clarified this point.

Another question/consideration. Assume we a running the FWD client code on Linux/Solaris based OS. Because COM feature is working on Windows only - the COM related native code will not work even if the server side is running on the Windows based machine. In this config we will generate the error(GUI or ChUI) and ignore the COM automation converted code(like if we have only stub level implementation). Is it correct understanding?

#20 - 09/08/2017 02:30 PM - Greg Shah

> Assume we a running the FWD client code on Linux/Solaris based OS. Because COM feature is working on Windows only - the COM related native code will not work even if the server side is running on the Windows based machine. In this config we will generate the error(GUI or ChUI)

Yes. Test how the 4GL responds in both GUI and ChUI and match this.

> and ignore the COM automation converted code(like if we have only stub level implementation).

I'm not sure about this. It will depend on how the 4GL behaves. My guess is that any access to the COM features will raise an ERROR or STOP condition and FWD should do the same.

#21 - 09/28/2017 09:14 AM - Eugenie Lyzenko

To create COM OLE native sample it is required to have the tool that producing unique 16 byte GUIDs identifier. Although it is possible to use some on-line web calculators there is option to have Microsoft designed tool included in one of the Visual Studio package. As far as I know not all distribution have the tool GUIDGEN.EXE. MS decided to not include it in modern versions. But version MS Visual Studio 2010 Express has this one. To use it just install minimum components(this version is free to use), run MS Visual Studio command processor and execute guidgen command.

It is not clear how this environment interferes with already installed MinGW tools so it is better to set it up on a standalone system or virtual machine until this point will be finally clarified.

You can generate GUID with tool copy it and paste into any source code or text file.

#22 - 09/28/2017 10:04 AM - Hynek Cihlar

Eugenie Lyzenko wrote:

> To create COM OLE native sample it is required to have the tool that producing unique 16 byte GUIDs identifier. Although it is possible to use some on-line web calculators there is option to have Microsoft designed tool included in one of the Visual Studio package.

Eugenie, why not to use java.util.UUID.randomUUID() to generate GUIDs? Also see https://stackoverflow.com/a/6953207.

#23 - 09/28/2017 12:32 PM - Eugenie Lyzenko

Hynek Cihlar wrote:

> Eugenie, why not to use java.util.UUID.randomUUID() to generate GUIDs? Also see https://stackoverflow.com/a/6953207.

Because I need to implement native C code working on Windows. I need to have GUID on DLL compilation stage.

#24 - 09/28/2017 12:48 PM - Hynek Cihlar

Eugenie Lyzenko wrote:

> Hynek Cihlar wrote:
>
> > Eugenie, why not to use java.util.UUID.randomUUID() to generate GUIDs? Also see https://stackoverflow.com/a/6953207.
>
> Because I need to implement native C code working on Windows. I need to have GUID on DLL compilation stage.

In that case just use CoCreateGuid, the function is part of the core COM infrastructure available on any Windows since version 2000 I believe.

#25 - 09/28/2017 01:20 PM - Eugenie Lyzenko

Hynek Cihlar wrote:

> In that case just use CoCreateGuid, the function is part of the core COM infrastructure available on any Windows since version 2000 I believe.

Again, I do not need to generate GUID at application runtime. I need have and **know** the GUID at compilation stage of the COM DLL, when I will install COM object on the system and use it from 4GL code. This must be the same GUID the COM object associated with because it is the only way to separate one COM object from another.

#26 - 09/28/2017 01:29 PM - Hynek Cihlar

Eugenie Lyzenko wrote:

> Hynek Cihlar wrote:
>
> > In that case just use CoCreateGuid, the function is part of the core COM infrastructure available on any Windows since version 2000 I believe.
>
> Again, I do not need to generate GUID at application runtime.

I was replying to #3262#note-21. You suggest there to download Visual Studio (eventually in a virtual machine). This sounded like a lot of hassle to me when you could create a simple program in MinGW using CoCreateGuid to generate GUIDs.

#27 - 09/28/2017 07:28 PM - Eugenie Lyzenko

Hynek Cihlar wrote:

> I was replying to #3262#note-21. You suggest there to download Visual Studio (eventually in a virtual machine). This sounded like a lot of hassle to me when you could create a simple program in MinGW using CoCreateGuid to generate GUIDs.

There is even simpler approach if it is a problem to use VS distro - just to use on-line Web GUID generator.

#28 - 09/28/2017 07:33 PM - Eugenie Lyzenko

The bzr testcases repo has been updated to revision 1642.

This adds uast/com_ole_automation/win_com_dll/ directory with starting point code to create our own COM object to test. This can be build into DLL with MinGW make. Continue working.

#29 - 09/28/2017 09:06 PM - Eugenie Lyzenko

Before trying to use our COM object we have to register it in the system. To do this manually:

1. Open Windows registry editor
2. Locate HKEY_LOCAL_MACHINE\SOFTWARE\Classes\CLSID entry.
3. Inside this create a key names with GUID taken from guidgen in format {XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX}.
4. Change the key default to something descriptive to help identifying like Fwd COM sample object.
5. Under key created in 3 create new key named InprocServer32.
6. Change default value for key created in 5 to the full path name of the COM object DLL for system is able to find it while loading.

7. Add new REG_SZ value to the key created in 5 with name ThreadingModel and assign the value data to Both.

#30 - 09/28/2017 09:12 PM - Greg Shah

Can you create a simple command line tool to register the COM object? I appreciate the manual documentation (it really tells you all the good details). But it seems error prone to do it manually every time. I think the tool can have no args or very simple args.

#31 - 09/28/2017 10:29 PM - Eugenie Lyzenko

Greg Shah wrote:

> Can you create a simple command line tool to register the COM object? I appreciate the manual documentation (it really tells you all the good details). But it seems error prone to do it manually every time. I think the tool can have no args or very simple args.

OK. I just wanted to present all the ways possible for this pretty rare operation. Also from this instruction it is clear how to completely remove it.

#32 - 10/01/2017 09:29 AM - Greg Shah

From Eugenie:

> Our COM object and all other automation objects must be accessed from 4gl code by string name identifier. This will be performed in FWD native code by CLSIDFromProgID() MS API and call native object with CLSID obtained to get the IDispatch with object access to methods and properties. This will be our Java-COM bridge.

#33 - 10/02/2017 10:49 PM - Eugenie Lyzenko

The bzr testcases repo has been updated to revision 1644.

This change adds IDispatch functionality handling. So our COM object has all internals need to test usage with method name calls. The C program does not need more to use COM. But 4gl like other script based language needs the type library to be compiled and installed in addition to DLL code itself.

And here we have faced another MinGW compiler. It does not have MIDL.EXE tool to build custom type library. So for now we need to use MS Visual C SDK package at least to compile testing library. The good news is from the FWD implementation point of view there is no need to have MIDL.EXE. Going to use the system I have Visual Studio installed.

I think the further implementation need to change a plan a bit. I'm going to start implementing FWD part in parallel to test library. In addition we need to write some helpful command line tools like test DLL reg/unreg. There are a set of potentially dangerous registry changes need to be completely safe. In general I think now I have enough info to start working on FWD part. The plan is to go in all implementing directions in parallel to find out the unknown point as fast as possible.

The HKEY_LOCAL_MACHINE\Software\Classes\ProgID must have string name that is referencing to define COM object in 4gl. This key should be set up for our testing DLL. And for every COM object the 4gl/fwd is going to use.

Continue working.

#34 - 10/03/2017 05:53 AM - Greg Shah

The plan is good.

Make sure you have read the "COM objects: Automation objects and ActiveX controls", "Using COM Objects in ABL", "COM Object Data Type Mapping" and "ActiveX Automation Support" chapters of the "Programming Interfaces" 4GL book. Do not trust that the documentation is complete or even correct. But it should help ensure you are not missing any major features. Finding out which part of the documentation is missing and which parts are wrong is important.

We are not going to support Active-X Controls (or the ABL CONTROL-FRAME). We only need to support COM Automation (also known as Active-X Automation) which is created via CREATE automation-application-name.

Please make sure you are handling the following:

- Method signatures that test:
  - Case-insensitive method names.
  - All possible return BDT data types including memptr, unknown, com-handle, no-return-value and extent (if extent is possible).
  - no-return-value should be tested with code that requires it, code that doesn't require it and with code that actually returns a value.
  - All possible parameter BDT data types including memptr, unknown, com-handle and extent (if extent is possible).
  - Parameter configuration:
    - access modes: INPUT (which is the default, can it be specified directly?), INPUT-OUTPUT, OUTPUT parameter types including mixtures of these types
    - Make sure you check this note from the 4GL docs "Note that ABL does not use Type Library information to determine the parameter mode. This prevents the COM object from updating a variable that you do not expect or want to change. Thus, if the COM object ordinarily changes the value of a particular parameter, you can prevent any variable you pass from having its value changed by omitting any mode option on the parameter."
    - AS datatype
    - BY-POINTER
    - BY-VARIANT-POINTER
    - "null parameter" case where you just have whitespace in the comma-separated parameter location
  - Different numbers of parameters. Try 1, 2, 3 and something larger like 6 ot 7.
  - Call a method name that doesn't exist.
  - Pass parameters using an ABL data type that doesn't match or automatically convert.
  - Assign a return value to ABL data type that doesn't match or automatically convert.
- Properties that test:
  - Case-insensitive property names.
  - All possible parameter BDT data types including memptr, unknown, com-handle and extent (if extent is possible).
  - Indexed properties, including multiple index dimensions. By the way, we probably parse these cases as COM methods not COM properties, so I think this will be a conversion bug. Since the syntax is the same for the two cases AND there is no type information available at conversion/parse time, I don't see an obvious way to deal with this during parsing. If it is on the left side of an assignment operator, then we can fix that in the parser. But if this same form is used in a non-assignment expression or on the right side of an assignment operator, this will have to be handled by the runtime code.
  - Assignment of an indexed property.
- Chaining on the left side of the assignment operator to obtain the property to assign.
  - AS datatype

- Read a property name that doesn't exist.
- Write to a property name that doesn't exist.
- Write to a read-only property.
- Read a property into an ABL data type that doesn't match or automatically convert.
- Write to a property with an ABL data type that doesn't match or automatically convert.

- Events
  - ENABLE-EVENTS method
  - event invocation of the internal procedure named with the matching prefix.eventname
  - parameter passing (if this works) to the event procedure
- COM code that raises an error. The documentation says these are translated into ABL errors, what does this mean and how do we match it?
- All 4 CREATE forms:
  - named automation object
  - CONNECT with named automation object
  - CONNECT TO file with named automation object
  - CONNECT TO file with implied automation object
- RELEASE OBJECT and VALID-HANDLE, it is especially important to match exactly what is happening on release (this is not exactly clear from the docs but the docs suggest that a release may cause some state change in the native/COM code such that all com-handles that reference that same object will be invalid AND the results of valid-handle() will be consistent)

I expect that some BDT data types may not map to native equivalents. For example, can memptr be converted to com-handle? For this reason, you must very carefully test all of the default data type mappings as documented in "Conversions from COM to ABL data types" and "Conversions from ABL to COM data types" of the "Programming Interfaces" 4GL book. In addition to these, try to find any additional mappings that are not documented but which might conceivably work. Consider also the array processing, pointer conversion and alternate type names that are possible.

Data type sizes, underflows and overflows are important to check. I think this would happen with the AS datatype forms where small native types (unsigned byte, short...) interact with larger ABL types (integer, int64, decimal).

It is also important to test the creation of new COM objects and returning them (so that the 4GL maps it to a com-handle). This use case and the

chaining of calls to the returned com-handle is really important to test because it is a common use case. I think this means that you must create more than one COM object (so that we have a second COM object type to return and use).

You may find some limitations when you try extent versions of these types. If you find any 4GL compile failures, that test code will have to be commented out of the 4GL. Let such code remain in the program inside a comment that also has a copy of the compile error that occurs when the code is uncommented. If you find a runtime error, then use NO-ERROR (if possible) and put code in to test the ERROR-STATUS like this:

```
run input_uint8_cdecl (input d1) no-error.
if not error-status:error or not error-status:get-number(1) eq 3231 then run log-err (input "Unexpected result
 for type date (" + string(d1) + ").").
```

You may find some cases where the error handling cannot be suppressed with NO-ERROR. Something like this can be used to test such cases:

```
/* This generates a STOP condition and displays the "error" message: */
/* ** "missing-proc-def" was not found. (293) */
/* Since it is not an error, NO-ERROR does not suppress the message.  The message text is */
/* logged (get-message() returns it) in the error-status and the number is listed using */
/* get-number() but the error-status:error flag is not set to true */
flag = true.
do on stop undo, leave:
   run missing-proc-def (output i) no-error.
   flag = false.
end.
if not flag or not error-status:get-number(1) eq 293 then run log-err (input "Expected error 293 for calling a
 procedure name that was never defined. Got '" + error-status:get-message(1) + "'.").
```

I know this is a lot to check. This means many testcases need to be written and lots of properties and methods to create in the COM object. Attention to detail now will make it unnecessary to fix bugs later. Fixing those bugs will take much longer than just writing it correctly the first time. This means it will save time in the end by doing more work in advance and being really thorough. The library_calls testcases show a great example of this AND it handles so many of the same problems (return values, parameter modes, data type mapping and conversion, create/release of resources...). Using that as a model may save you some time and help you understand the long list of items above.

Let me know what questions you have.

#35 - 10/03/2017 06:37 PM - Eugenie Lyzenko

The bzr testcases repo has been updated to revision 1647.

This adds the simple tool to register/deregister our COM object without manual enter data into registry. In addition to actions described in previous manual scenario the tool adds ProgID key into HKEY_LOCAL_MACHINE\SOFTWARE\Classes\CLSID\{FWD GUID number} entry with string value "IFwdTest.object" and key IFwdTest.object for HKEY_LOCAL_MACHINE\SOFTWARE\Classes\ hive with the GUID value. This makes possibility to exactly match verbose name("IFwdTest.object") we will use in 4gl and numeric GUID.

The tool IFwdReg.c must be compiled in the same directory as other COM native tool. The usage is:
IFwdReg to install the FWD COM object,
IFwdReg u to remove FWD COM object from system.

Type Libraries are not yet supported.

#36 - 10/03/2017 07:52 PM - Eugenie Lyzenko

Greg Shah wrote:

> Let me know what questions you have.

For now the point is really unclear for me is event processing. I thought the COM event handling is the feature of the ActiveX control object within control frame. And we are not planning to implement support for this functionality. An I wrong here? Are there any COM specific event processing for automation object we are going to implement? Is there any 4gl documentation I can read to find out the internals?

#37 - 10/04/2017 10:47 AM - Greg Shah

Make sure you read the section of the 4GL docs (Programming Interfaces book) called "ActiveX Automation Support", especially the "Automation event support" sub-section. This is the same as COM Automation. COM Automation does support events.

#38 - 10/04/2017 07:57 PM - Eugenie Lyzenko

The bzr testcases repo has been updated to revision 1648.

Adding type library support for testing COM object. Changed internal string usage from char array to BSTR to match the COM MS methodology, changed the test application to check COM object. The building commands are also changed.

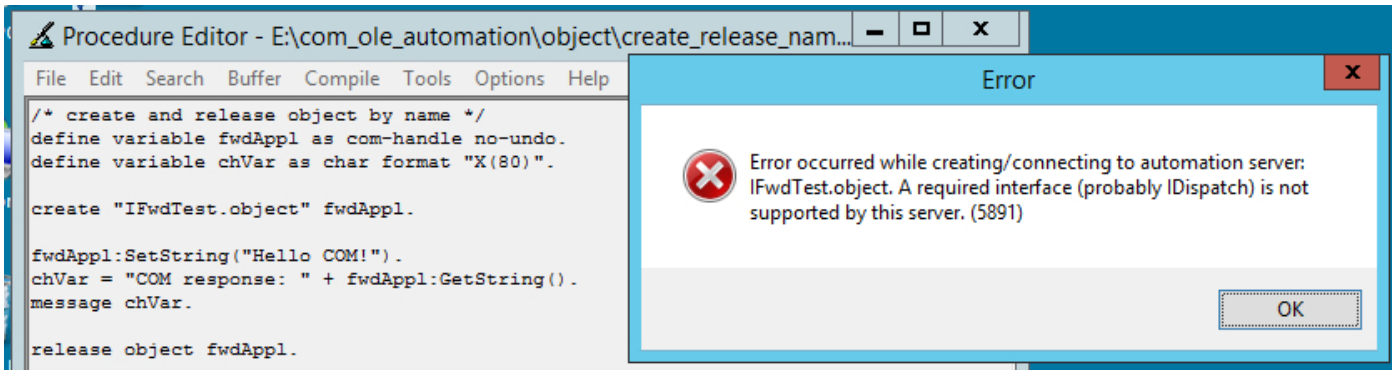
#39 - 10/05/2017 10:18 PM - Eugenie Lyzenko

- File create_object_error.jpg added

The bzr testcases repo has been updated to revision 1649.

Starting to create complete test set to investigate required features to implement. The first 4gl example with simple object creation and get/set operation produces the runtime error meaning something is wrong with our COM test object(or installation). However the C based code for using the

COM test works fine. Investigation is in progress to have working COM library for 4gl. The problematic picture is attached:



#40 - 10/06/2017 10:12 PM - Eugenie Lyzenko

The bzr testcases repo has been updated to revision 1653.

Starting with this update we have working COM test object co-existed with 4gl sample. I have tested it also with Visual Basic script(also works now) but it is not included here because far away from branch subject. The problem was in DLL code error, IDispatch interface support is the key point in QueryInterface call to work properly for script based languages. During debugging the EXE based com object was also created(included here with install/remove tool). Anyway now our 4gl tests are able to work with both DLL and EXE based com objects.

The working system is developer virtual machine image from <customer_name_redacted> - Windows Server 2012. I had to set up all required compilers, MinGW and Visual Studio Community 2017(to make type library compilation and GUID getting). They can co-exist. Continue adding more tests/testing features.

#41 - 10/09/2017 10:27 PM - Eugenie Lyzenko

The bzr testcases repo has been updated to revision 1654.

This update adds checking the case sensitivity for methods and properties. Also changes the DLL test library to properly coexist both calls.

The result is: There is no difference for letter case, 4gl ignores the name case for both methods and properties, no errors, the calls are handling fine no matter what the letter case is.

#42 - 10/10/2017 11:30 PM - Eugenie Lyzenko

The bzr testcases repo has been updated to revision 1657.

Added simple data type testcases. LOGICAL, INTEGER, DECIMAL, INT64 and CHARACTER.

The new finding is strange. Actually INT64 is not supported for COM automation objects in 4GL. Attempting to go beyond 4 byte integer causes the out of range message. The value inside COM object is not updating although the room is enough(long data in C code).

#43 - 10/11/2017 09:50 PM - Eugenie Lyzenko

The bzr testcases repo has been updated to revision 1660.

Added complex data type testcases for methods parameters and return values: LONGCHAR, DATE, DATETIME, DATETIME-TZ, COM-HANDLE, ARRAY(EXTENT based), VARIANT(arbitrary data are able to be mapped). The finding is MEMPTR is not supported. Also it is not clear how the array data is mapping with COM<->ABL. The simple schema with using SAFEARRAY does not work generating runtime exception for invalid parameters. Probably direct simple data type primitive array is used instead. On the other hand the array size must be passed to COM and back together with array itself. Continue investigating. The next steps will be adding data tests for COM object parameters set/get and COM OLE events.

#44 - 10/11/2017 10:20 PM - Greg Shah

I'm surprised that MEMPTR can't be used. In WIN32 library calls, the MEMPTR is used heavily. For example, any WIN32 API that calls for a pointer to a structure to be passed, must have the structure manually built in the buffer allocated as the MEMPTR.

It would surprise me if this same thing was not possible for COM objects. Can structures get passed as parameters?

#45 - 10/12/2017 09:04 AM - Eugenie Lyzenko

Greg Shah wrote:

> I'm surprised that MEMPTR can't be used.

This match the 4GL Programming Interfaces document book, page 624.

> In WIN32 library calls, the MEMPTR is used heavily. For example, any WIN32 API that calls for a pointer to a structure to be passed, must have the structure manually built in the buffer allocated as the MEMPTR.
>
> It would surprise me if this same thing was not possible for COM objects. Can structures get passed as parameters?

I think if the address can be converted to integer value(or longint) it can be passed to COM. The question is if it is possible to do in 4gl.

#46 - 10/12/2017 09:14 AM - Greg Shah

> I think if the address can be converted to integer value(or longint) it can be passed to COM. The question is if it is possible to do in 4gl.

See GET-POINTER-VALUE(memptr) in the 4GL docs.

On the other hand, if this is never used by real applications then it is less concerning. I think it makes sense for you to setup the customer application and run reports. Then you can look at the COM automation use cases and hopefully make sure that we are handling everything needed.

#47 - 10/12/2017 12:43 PM - Eugenie Lyzenko

Greg Shah wrote:

> I think if the address can be converted to integer value(or longint) it can be passed to COM. The question is if it is possible to do in 4gl.

> See GET-POINTER-VALUE(memptr) in the 4GL docs.

Yes, the result of the GET-POINTER-VALUE(memptr) can be stored in COM. With only restriction the INT64 as function result for GET-POINTER-VALUE will be truncated to 4-bytes INTEGER inside COM object.

> On the other hand, if this is never used by real applications then it is less concerning. I think it makes sense for you to setup the customer application and run reports. Then you can look at the COM automation use cases and hopefully make sure that we are handling everything needed.

OK.

#48 - 10/12/2017 09:20 PM - Eugenie Lyzenko

The bzr testcases repo has been updated to revision 1662.

Added properties access(set and get) tests for set and get data types covered in previous method set/get. No special consideration findings for properties data types mapping. Including MEMPTR handling. Also not clear how the EXTENT base ABL arrays mapped to COM. Continue investigations.

#49 - 10/13/2017 08:47 AM - Eugenie Lyzenko

Greg Shah wrote:

> On the other hand, if this is never used by real applications then it is less concerning. I think it makes sense for you to setup the customer application and run reports. Then you can look at the COM automation use cases and hopefully make sure that we are handling everything needed.

Do you mean the source <customer_information_redacted>, correct?

#50 - 10/13/2017 08:52 AM - Greg Shah

> Do you mean the source <customer_information_redacted>, correct?

Yes.

#51 - 10/13/2017 10:38 PM - Eugenie Lyzenko

The additional investigation performed to make array handling for ABL<->COM transformation. The only case the EXTENT based array is passing to the COM object from 4gl is defining the array option/return value as VARIANT array[]. However this case has at least two constraints:

1. We have to exactly know what is the array dimension inside COM object because this signature does not have room for array size parameter unlike SAFEARRAY.
2. The VARIANT based API functions to work with data need the pointer to VARIANT structure. Passing the VARIANT[] variable causes some issues accessing data buffer to be set up while copy data from COM to external world.

Investigation must be continued to find out all details.

Also another finding of this day. The correct COM data used to map 4gl INT64 is __int64 instead of long. long is 32 bit integer in term of MIDL compiler(the way the external languages will see the COM object data type). Using __in64 in our C based COM test module fix INT64 constraints noted previously so now we can say it works as expected for 64-bit integer.

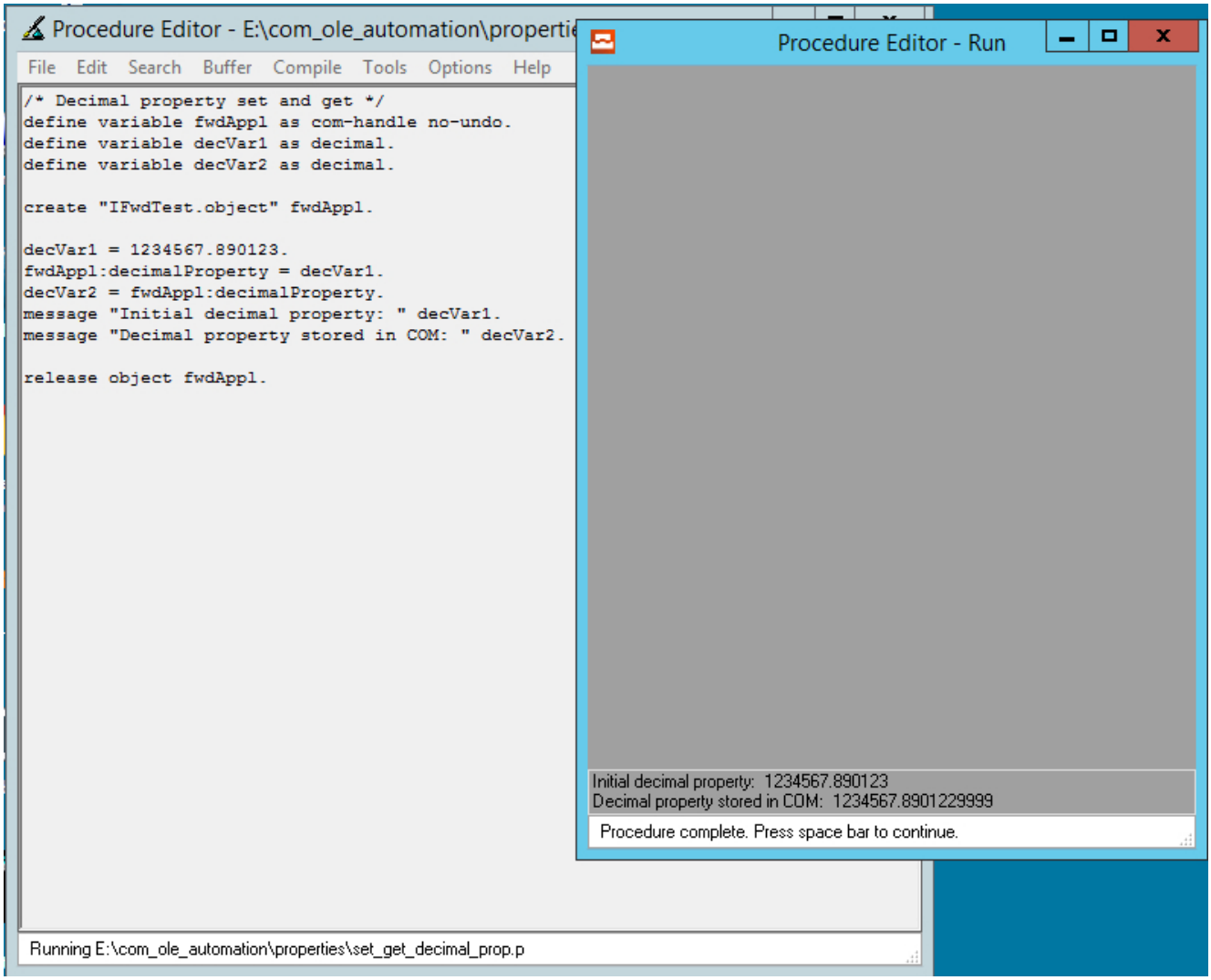
#52 - 10/14/2017 07:54 PM - Eugenie Lyzenko

- File decimal_precision_issue.jpg added

The bzr testcases repo has been updated to revision 1665, 1666.

Fixed INT64 data processing. Added EXTENT based array data type. The right way was to use VARIANT* variable as parameter or return type to move data between ABL and COM. The 4gl passes the pointer to array. COM is taking array encapsulated into VARIANT* type variable.

Another finding is related to decimal value storing into COM(both property and method). There is small precision loss when getting back the value. On the picture attached you can see the value came from COM is slightly different.



#53 - 10/15/2017 08:56 AM - Greg Shah

Is the decimal precision loss explained by the use of floating point on the COM object side? If a double is used to store the decimal value in native code, not all decimal numbers can be represented exactly. The imprecision could show up as this unintended data modification.

#54 - 10/15/2017 12:17 PM - Eugenie Lyzenko

Greg Shah wrote:

> Is the decimal precision loss explained by the use of floating point on the COM object side? If a double is used to store the decimal value in native code, not all decimal numbers can be represented exactly. The imprecision could show up as this unintended data modification.

Yes, I used double to store DECIMAL value.

#55 - 10/16/2017 11:49 PM - Eugenie Lyzenko

The investigation is performing to have a clear picture of COM automation objects event processing for 4gl application. Certainly the 4gl supports events handling coming from COM objects. The key point is automation object method. The approach is:
1. Write procNamePrefix.(event1 | event2 |... eventN) procedures.
2. Call Object:ENABLE-EVENTS(procNamePrefix) to turn on event processing.
3. COM object fire events(event1 | event2 |... eventN) causing the respective procedure to be called on the 4gl side.

Now investigating COM side details to implement/integrate event firing into our IFwdTest sample object. The IDL file is ready, need to add C code handling. Learning IConnectionPoint and IConnectionPointContainer object required implementations as the core functionality required to be written.

#56 - 10/17/2017 11:07 PM - Eugenie Lyzenko

The bzr testcases repo has been updated to revision 1667.

This is starting point for investigation different aspects of the COM Automation event processing. All facilities required for COM object to fire events are there. The other work is up to 4GL code. Will need to create new Progress event related tests.

Some details:
1. All event processing is based on callback technology. The client approach should provide entry points for COM object to fire events.
2. We will have to implement all internals that currently 4GL does to transform the 4GL procedure's pointers into callback sink subsystem.
3. The COM object approach of handling events firing consist of following steps on the COM server side:
- client calls COM:QueryInterface (ConnectionPointContainerID...) to get the pointer to the IConnectionPointContainer object from COM
- having IConnectionPointContainer the client then calls IConnectionPointContainer.FindConnectionPoint to get the IConnectionPoint object for given COM
- having IConnectionPoint the client then calls IConnectionPoint.Advise()/Unadvise() passing the pointers to callback set to be executed when events are happening.
- COM object prepares event optional parameters, pack it into callback buffer and call Invoke to pass data to the client

The approach from points is what we will have to implement to handle events in our FWD<->COM bridge/(sink part for event handling). Currently it is expected the same bridge in implemented in 4GL.

So the next step is to write COM events 4GL samples to uncover all 4gl event processing details in this area.

#57 - 10/17/2017 11:10 PM - Eugenie Lyzenko

Very good reading related to COM event processing:
https://www.codeproject.com/Articles/14183/COM-in-plain-C-Part

As soon as all articles of the same author regarding COM in pure C implementation. It is **very** rare to find something about COM development in C.

#58 - 10/18/2017 07:07 AM - Greg Shah

We will have to dispatch received COM events into calls to converted internal procedure. The internal procedure name(s) are mapped to the event name and at ENABLE-EVENTS() it would call the registration processing described in #3262-56. Upon dispatch the internal procedure signature would have to match the required COM event signature.

Eugenie: Make sure to test the error processing for internal procedures that do not match the right signature.

Constantin: Can we use a dispatching approach similar to named events (pub/sub)?

#59 - 10/18/2017 03:18 PM - Eugenie Lyzenko

I have faced with small surprise. The 4GL object handle built-in method ENABLE-EVENTS() does not work as expected. It requires the COM object to have respective ENABLE-EVENTS() method. And this is strange because the COM object does not have to know anything about Progress 4GL intentions to use it. Investigating.

#60 - 10/18/2017 11:09 PM - Eugenie Lyzenko

The bzr testcases repo has been updated to revision 1669.

Fixed the small bug in interface ID definition. However this resolves only part of the issue. Now is is possible to use event firing from our COM with VisualBasic test(IFwdEvent.vbs script). This means technically the COM object is able to generate events in certain condition. But 4GL code sample that included in this update generates error meaning wrong/missing interface.

Looks like 4GL interpreter uses more advanced approach to connect the COM object and we need to do matching capability. I guess to start with implementing EnumConnectionPoints (currently it is stubbed). Continue debugging.

#61 - 10/19/2017 09:23 PM - Eugenie Lyzenko

The bzr testcases repo has been updated to revision 1670.

Debugging performed to investigate the 4GL event handler installing issue.

1. First I have added native C based client application to verify current COM code to work properly. The result - everything is OK when client knows the ID of the event interface. This means the 4GL client uses different approach connecting COM object than implemented in native code and VisualBasic.
2. Tracking the QueryInterface method of the COM object it was discovered the 4GL client asks for the new Windows component interface, ID == {B196B283-BAB4-101A-B69C-00AA00341D07}, looking for registry found this is IProvideClassInfo interface not documented at this time anywhere. This class has method to get the type info for interfaces of the COM object. Adding implementation to this new interface(included in 1670) resolves the error message for invalid interface type. However this does not come us to working COM object event firing for Progress 4GL sample.
3. Additional investigation shows the 4GL approach for installing event callback uses IConnectionPointContainer.EnumConnectionPoints() method to be implemented in COM(unlike to VisualBasic code and native C application).

So we need to add implementation for EnumConnectionPoints() method to the COM object. The main task here is to add another interface: IEnumConnectionPoints to provide info about event related interfaces from COM to client. And this is the point where is no help good info on the Web to uncover MS secrets, so we will have to be the pioneers. We can not ignore this approach because we need to implement the respective replacement in our native bridge code. Continue investigating.

#62 - 10/20/2017 07:27 AM - Greg Shah

IEnumConnectionPoints MSDN Docs

#63 - 10/20/2017 11:24 PM - Eugenie Lyzenko

Greg Shah wrote:

> IEnumConnectionPoints MSDN Docs

The bzr testcases repo has been updated to revision 1673.

This update introduces the event test firing/processing for FWD COM object. To use the functionality in 4GL the COM object must have IConnectionPointContainer.EnumConnectionPoints() method implemented. In addition the interface IEnumConnectionPoints must also be implemented. I made this only for methods used during 4GL event handling negotiation(Clone is not yet implemented because not used to install callback).

Very important point. Although we have some MSDN description noted above for IEnumConnectionPoints interface, there is an issue using this doc. The interface's methods are mentioned in alphabetical order on MS Site. And this order is not correct for doing method table. To find out the correct order(and this is a key point for interface to work properly) it is required to manually search *.h files inside MinGW package and reproduce exact order from there. Otherwise COM object will trap. The same is true for every I* MS interface we will need to implement.

The other finding is the IDL file needs to be written considering recent MS updates for event callback description. Now it must have keyword dispinterface. Previously used interface ... : IDispatch clause now is not working.

Also when using COM object from 4GL code another sub object is used - IRunnableObject with ID {00000126-0000-0000-C000-000000000046}. For now the test samples works fine without implementing this so I postpone possible implementation for this interface to later time in case we will need it.

So now we can finally start testing the event behavior coming from COM object to 4GL. The first simple test shows the event handler name on the Progress side is case insensitive. Will continue with other event related testing next day.

The test COM object now has some debug points to show internal details using Windows message boxes. In final version this will be cleaned up.

And some new finding I'm concerning a bit. The event handling for COM object part uses so-called cookie variable to store client's callback procedure address to register/de-register. The interesting fact the variable is 32-bit. And having wrong value in this variable causes the COM based application to stop working properly. And now consider 64-bit pointers that are not able to be packed into 32-bit DWORD data type. So the main question: Is the COM OLE MS event subsystem is only 32-bit aware? Not very good if so. Especially when working with object pointers passed from one application to another. Not clear for now how it can affect 64-bit system working with COM objects.

#64 - 10/21/2017 10:01 AM - Eugenie Lyzenko

Created task branch 3262a from trunk revision 11180.

#65 - 10/21/2017 07:34 PM - Eugenie Lyzenko

The bzr testcases repo has been updated to revision 1674.

Adding functionality and tests to pass different parameter types from COM to 4GL within event via VARIANT.

#66 - 10/22/2017 06:09 AM - Constantin Asofiei

Eugenie, two issues:

1. am I right to assume on the native side there will be a registry mapping an ID to a COM pointer?
2. do you have a list of native APIs which would be required?

#67 - 10/22/2017 12:23 PM - Eugenie Lyzenko

Constantin Asofiei wrote:

> Eugenie, two issues:
> 1. am I right to assume on the native side there will be a registry mapping an ID to a COM pointer?

I'm planning to use CLSIDFromProgID() in native code to map application string based name into ID we will use to work with COM.

> 2. do you have a list of native APIs which would be required?

For now I do not have such exact list for native calls I will have to use inside C code because it may be expanded. From the Java side view there no such requirements to know native implementation internals. We will need to commit the JNI call names you will need to use working on the Java side and I will need to implement on the native C side. This is pretty freedom choice for JNI method names. The first thing we need to commit is the Java helper class that will handle JNI requests incoming from Java side. Any suggestions? I would suggest ../p2j/comauto/ComOleHelper.java

Basically the Java side will need to:
1. Create COM object instance
2. Set/Get COM object properties
3. Call COM object method
4. Set up Java based event handler(4GL ENABLE-EVENTS() built-in method)

Task branch 3262a has been rebased with current trunk 11181.

#68 - 10/22/2017 01:43 PM - Constantin Asofiei

Eugenie Lyzenko wrote:

> Constantin Asofiei wrote:
>
> > Eugenie, two issues:
> > 1. am I right to assume on the native side there will be a registry mapping an ID to a COM pointer?
>
> I'm planning to use CLSIDFromProgID() in native code to map application string based name into ID we will use to work with COM.

Is it possible to have multiple instances of the same COM application?

#69 - 10/22/2017 04:00 PM - Eugenie Lyzenko

Constantin Asofiei wrote:

> Is it possible to have multiple instances of the same COM application?

I guess it depends on particular COM object implementation. Sometimes it is possible, sometimes it is not. We have to consider both cases.

#70 - 10/22/2017 05:33 PM - Constantin Asofiei

Eugenie Lyzenko wrote:

> Constantin Asofiei wrote:
>
> > Is it possible to have multiple instances of the same COM application?
>
> I guess it depends on particular COM object implementation. Sometimes it is possible, sometimes it is not. We have to consider both cases.

OK, then we will need the native API call used to create a COM object to return an ID, which will be passed to all other calls related to this COM object (i.e. property access, method access, destroy, etc). And the native side will need to keep track of this "ID - to - COM" mapping to be able to identify what instance the API call will target.

#71 - 10/22/2017 07:14 PM - Eugenie Lyzenko

Constantin Asofiei wrote:

> I guess it depends on particular COM object implementation. Sometimes it is possible, sometimes it is not. We have to consider both cases.

OK, then we will need the native API call used to create a COM object to return an ID, which will be passed to all other calls related to this COM object (i.e. property access, method access, destroy, etc). And the native side will need to keep track of this "ID - to - COM" mapping to be able to identify what instance the API call will target.

This is the important point we need to get an agreement. In Windows the COM object is identified with 16-bytes unique ID. This is too much data to pass from Java to native code as function option, agreed?

In native code after object created/connected to the application receives the pointer to it(LPVOID data type). The size is different for different system architecture(32 or 64 bit) but in every case this pointer value can be packed into single base data type value for given system in use. I suggest to use this object pointer to identify COM entity for Java code. So no GUID values will be used in Java code. The COM object Java implementation will store the native pointer inside and pass it to native code every time the JNI call is required, agreed?

#72 - 10/23/2017 09:19 AM - Constantin Asofiei

Eugenie Lyzenko wrote:

> So no GUID values will be used in Java code.

This is good.

> The COM object Java implementation will store the native pointer inside and pass it to native code every time the JNI call is required, agreed?

Java side doesn't really cares what kind of ID is returned by the native API which creates/connects to the COM object - so whatever this API returns, it will associate it with the Java-side COM handle instance and pass it to all native API calls for this Java COM handle instance.

So, can I assume a long value for this internal, native-side, ID of the COM object?

#73 - 10/23/2017 11:16 AM - Eugenie Lyzenko

Constantin Asofiei wrote:

> Eugenie Lyzenko wrote:
>
> So, can I assume a long value for this internal, native-side, ID of the COM object?

Having and keeping yet another object<->long value mapping inside native code is extra not very useful step. What I have suggested is to use COM object pointer received from Windows OS, cast it to long value and pass to Java side. Meaning the ID of the objects on the Java side will not be ordered(0, 1, 2,...). But they will be integer(32 or 64 bit depending on OS or generally 64 bit for every case - if we decide to use 64 bit anytime) and unique.

#74 - 10/23/2017 11:18 AM - Constantin Asofiei

Eugenie Lyzenko wrote:

> Constantin Asofiei wrote:
>
> > Eugenie Lyzenko wrote:
> >
> > So, can I assume a long value for this internal, native-side, ID of the COM object?
>
> Having and keeping yet another object<->long value mapping inside native code is extra not very useful step. What I have suggested is to use COM object pointer received from Windows OS, cast it to long value and pass to Java side. Meaning the ID of the objects on the Java side will not be ordered(0, 1, 2,...). But they will be integer(32 or 64 bit depending on OS or generally 64 bit for every case - if we decide to use 64 bit anytime) and unique.

OK, I agree and understand.

#75 - 10/23/2017 04:28 PM - Constantin Asofiei

Eugenie, have you investigated the CONNECT option in CREATE com... CONNECT TO ... statement?

Ovidiu, maybe you could share some light on what ServerComObject should do in FWD?

#76 - 10/23/2017 04:55 PM - Constantin Asofiei

Eugenie, do you have a mapping of 4GL data types to C/native data types? I need this to marshall/unmarshall the values, so the native side will always receive i.e. a String instance instead of character.

#77 - 10/23/2017 05:02 PM - Constantin Asofiei

Two more issues:

1. on a COM method call, if the method returns a COM pointer, i.e. a comVar:comProperty:someOtherProperty chain - where comProperty is a COM reference, I need to differentiate this value - any idea how we can mark this as a 'COM pointer'?
2. if the property/method name is not found in the COM object, are you planning to throw exceptions?

#78 - 10/23/2017 06:37 PM - Eugenie Lyzenko

Constantin Asofiei wrote:

> Eugenie, do you have a mapping of 4GL data types to C/native data types? I need this to marshall/unmarshall the values, so the native side will always receive i.e. a String instance instead of character.

Already verified:

4gl <--> native C
---
character, longchar<-->BSTR
decimal<-->double
integer<-->int
int64<-->__int64
date, datetime, datetime-tz<-->DATE
logical<-->VARIANT_BOOL
array, anydata<-->VARIANT*
com-handle<-->IUnknown*

#79 - 10/23/2017 06:52 PM - Eugenie Lyzenko

Constantin Asofiei wrote:

> Two more issues:
> 1. on a COM method call, if the method returns a COM pointer, i.e. a comVar:comProperty:someOtherProperty chain - where comProperty is a COM reference, I need to differentiate this value - any idea how we can mark this as a 'COM pointer'?

I think in this case we need to construct the proper calls on the conversion level like:
getComVar().getProperty1().getProperty2()
because if the property1 has a property2 - this means the propery1 is a COM object. None other data types can have properties.

> 2. if the property/method name is not found in the COM object, are you planning to throw exceptions?

As far as I know 4GL shows the error with "Unable to get property" for given wrong name. We can return the error code to Java meaning the property/method not found and then on Java level to make further error processing.

#80 - 10/23/2017 06:53 PM - Constantin Asofiei

Eugenie Lyzenko wrote:

> Already verified:

See bellow for my current assumptions of what will be passed from Java side:

```
4gl <--> native C
------------------------------
character, longchar<-->BSTR          Java: String
decimal<-->double                    Java: Double
integer<-->int                       Java: Integer
int64<-->__int64                     Java: Long
date, datetime, datetime-tz<-->DATE  Java: Date
logical<-->VARIANT_BOOL              Java: boolean
array, anydata<-->VARIANT*           Java: array  - have you tested extent parameters?
com-handle<-->IUnknown*              Java: any idea?
```

Also, I'll assume the 4GL unknown value will be identified as Java null.

3262a rev 11182 contains a first pass at the Java-side refactoring.

#81 - 10/23/2017 06:57 PM - Constantin Asofiei

Eugenie Lyzenko wrote:

> Constantin Asofiei wrote:
>
> > Two more issues:
> > 1. on a COM method call, if the method returns a COM pointer, i.e. a comVar:comProperty:someOtherProperty chain - where comProperty is a COM reference, I need to differentiate this value - any idea how we can mark this as a 'COM pointer'?
>
> I think in this case we need to construct the proper calls on the conversion level like:
> getComVar().getProperty1().getProperty2()
> because if the property1 has a property2 - this means the propery1 is a COM object. None other data types can have properties.

See note #3262-10, this can't be done, as conversion-time doesn't really know the structure of this. The native side needs to send this value already marked - can you build from the native side a 'ComObject' instance (or any other new custom, FWD, type) which I know to interpret as a pointer?

> > 2. if the property/method name is not found in the COM object, are you planning to throw exceptions?
>
> As far as I know 4GL shows the error with "Unable to get property" for given wrong name. We can return the error code to Java meaning the property/method not found and then on Java level to make further error processing.

The idea here is how do you pass the error-codes to the Java side? Can I use some ComOleHelper.hadError API to check the state of the last ComOleHelper API call? Otherwise I can't distinguish between i.e. a read property value and an error code.

#82 - 10/23/2017 07:10 PM - Eugenie Lyzenko

Constantin Asofiei wrote:

> Eugenie Lyzenko wrote:

> Already verified:
>
> See bellow for my current assumptions of what will be passed from Java side:
> [...]
> array, anydata<-->VARIANT* Java: array - have you tested extent parameters?

Yes, array here means extent N based array variable.

> com-handle<-->IUnknown* Java: any idea?

Object may be?

> Also, I'll assume the 4GL unknown value will be identified as Java null.

OK.

#83 - 10/23/2017 07:10 PM - Constantin Asofiei

Greg, I have a conversion issue which a hint to solve it would help a lot:

```
/* Case insensitive names call tests */
define variable fwdAppl as com-handle no-undo.
define variable chVar1 as char format "X(80)".
chVar1 = fwdAppl:About(input chVar1).
release object fwdAppl.
```

the input chVar1 is see as a input chVar1 in frame f1, and resulted code generates a frame0.getChVar1(), even if there is no frame involved. The problem is somewhere in the parser, as it assumes INPUT is a function, not an option.

#84 - 10/23/2017 07:32 PM - Eugenie Lyzenko

Constantin Asofiei wrote:

> Eugenie, have you investigated the CONNECT option in CREATE com... CONNECT TO ... statement?

Yes, we need to support all 3 cases of the object creation, probably by setting some mode option within create object Java method.

#85 - 10/23/2017 09:20 PM - Eugenie Lyzenko

The bzr testcases repo has been updated to revision 1675.

This testcases update adds event processing investigation, including events that accepting parameters and provide return value to the COM object. There was one important issue related to the events with return values. The handling of such cases is doing not the way directly documented on MS pages or somewhere the peoples look for solutions.

Consider we have one variable as event parameter and one variable as return value. The native code must prepare the VARIANT based array with 2 input parameters, one of the array entry is reserved for output value. And the parameter's ordering is reversed, the last parameter(return value in our case) become the first element of the input array. The specially designed return value storage is wrong place to put the return value storage, this does not work and causes the COM object to crash.

So the remaining aspect of the investigation is error processing part. Something already done. If the method/property name is missing or wrong or event procedure has mismatched input parameters - we have 4GL based error message and if the error is not fatal for COM code - the 4GL application continues. If COM object is trapped(for example accessing wrong memory region with 0x0000000D), the 4GL application stops execution.

#86 - 10/23/2017 10:20 PM - Eugenie Lyzenko

I'm preparing the base structure to implement native code. And need to clarify one point.

Consider calling COM Automation functions from Linux. What is our reaction in this case? Need to provide some error message? Or just use exception for native call is missing?

I would suggest the following approach.
1. Implement comauto.c as starting point for all native calls(from both Windows and Linux). Forward processing to OS specific module.
2. Implement comauto_win.c and comauto_linux.c as OS specific modules. comauto_win.c will have all real work for COM object handling. comauto_linux.c will implement only initComBridge() method returning false meaning the subsystem is not supported. The other JNI methods in comauto_linux.c will do nothing.
3. The first step the Java side will do is to call initComBridge() and set up support internal flag. Then for any COM related call Java side will checks this flag and do works only if support is true. This way we avoid traps on *nix based OS at runtime.

Is it OK? Greg, what do you think?

#87 - 10/23/2017 10:28 PM - Eugenie Lyzenko

Constantin,

Please clarify your expectations for JNI:

```
public static native Object[] call(long comId, String methodName, String modes, Object[] aParams);
```

Object[] aParams is an array of input parameters, correct?
JNI returns the value from method in array, correct? Do you expect the amount of the return values more than 1?
What is param String modes for?

#88 - 10/23/2017 10:40 PM - Eugenie Lyzenko

Task branch 3262a has been updated to revision 11183.

Adding init JNI to helper class and instructions to generate header for native library.

#89 - 10/24/2017 04:53 AM - Constantin Asofiei

Eugenie Lyzenko wrote:

> Constantin,
>
> Please clarify your expectations for JNI:
>
> [...]
>
> Object[] aParams is an array of input parameters, correct?

Yes. These will be Java types, not native, but objects from java.lang or arrays.

> What is param String modes for?

This represent how the arguments are passed. It is a string with the same length as the number of arguments, each char being:

- I for INPUT
- O for OUTPUT
- U for INPUT-OUTPUT

> JNI returns the value from method in array, correct? Do you expect the amount of the return values more than 1?

For OUTPUT or INPUT-OUTPUT arguments, I need the argument's value passed back by the caller. The return array for call will be:

1. on index 0, the return value
2. on index 1 to n (where n is the number of arguments), it will be:

- null for INPUT or OUTPUT/INPUT-OUTPUT arguments which are unknown
- a value for OUTPUT or INPUT-OUTPUT arguments

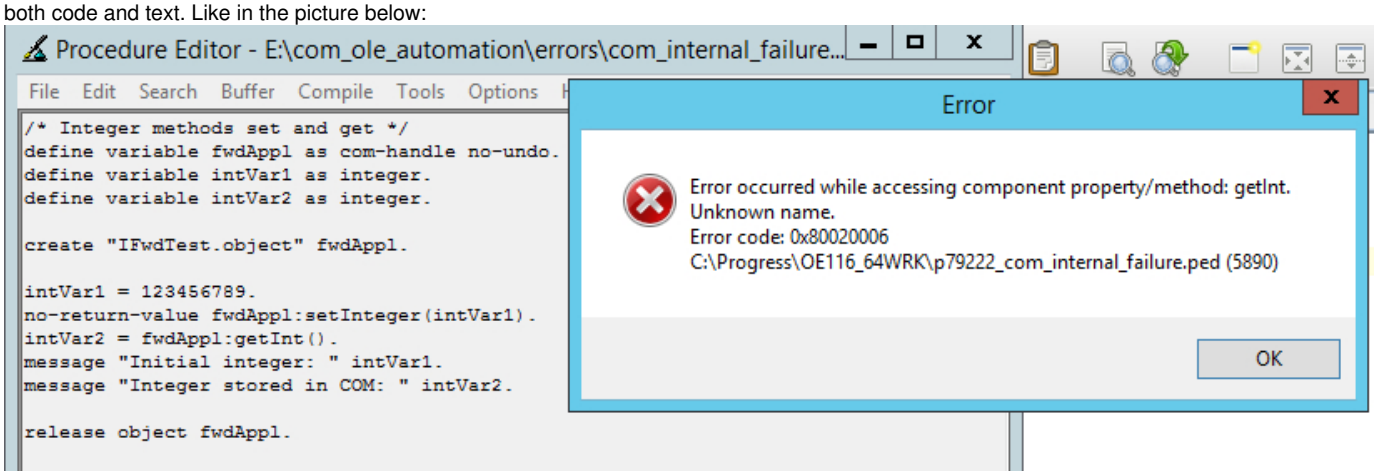
#90 - 10/24/2017 07:11 AM - Eugenie Lyzenko

- File com_no-error_option.jpg added

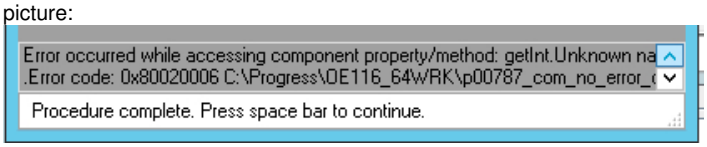
- File com_error_regular.jpg added

The bzr testcases repo has been updated to revision 1676.

This adds investigation results and testcases for COM related errors handling. There are two possible operation modes that control application behavior:
1. NO-ERROR option is not used. In this case the error code and message text is passing to 4GL side and 4GL displays the message box containing both code and text. Like in the picture below:



2. NO-ERROR option is used. In this case there is no message box displayed but the error code is storing in the 4GL ERROR-STATUS handle variable. The message details text and actual hexadecimal code can be obtained by ERROR-STATUS:GET-MESSAGE(errorNum). The sample picture:



So looks like in case 1 we need to initiate message box showing from native code(or throw exception from native code that FWD will intercept and display message box). For case 2 we need to set the ERROR-STATUS handle from native code with required info. This will require JNI->Java call methods I guess.

#91 - 10/24/2017 07:32 AM - Ovidiu Maxiniuc

Constantin Asofiei wrote:

> Eugenie, have you investigated the CONNECT option in CREATE com... CONNECT TO ... statement?
>
> Ovidiu, maybe you could share some light on what ServerComObject should do in FWD?

I guess this question is related to your question "Is it possible to have multiple instances of the same COM application?" from note 68. I don't know how things works exactly. In previous projects, I used ActiveX-es, but they were were a bit tamed by ActiveQt (see [http://doc.qt.io/qt-5/activeqt.html]). The set of possible ways to instantiate/attach to AX instance is well defined in CREATE automation object statement in ABL Reference. I used this section to write create methods in com.goldencode.p2j.comauto.ComServer. This is more than a stub, but they were not really tested. It is possible that they will need fixes, depending on the issues Eugenie is finding while investigating (like the above mentioned mapping between Java ComObject and the Windows AX id).

#92 - 10/24/2017 08:30 AM - Greg Shah

> Then for any COM related call Java side will checks this flag and do works only if support is true. This way we avoid traps on *nix based OS at runtime.

Yes, this is good. I assume Progress generates some ERROR or STOP condition in this case. Just implement the same approach.

#93 - 10/24/2017 08:33 AM - Greg Shah

> So looks like in case 1 we need to initiate message box showing from native code(or throw exception from native code that FWD will intercept and display message box). For case 2 we need to set the ERROR-STATUS handle from native code with required info. This will require JNI->Java call methods I guess.

No, I think you are making this too complicated. What you are describing is just the normal behavior when an ERROR condition is raised (e.g. ErrorManager.recordOrThrowError()). I think no special JNI or native behavior is needed.

#94 - 10/24/2017 08:52 AM - Greg Shah

> the input chVar1 is see as a input chVar1 in frame f1, and resulted code generates a frame0.getChVar1(), even if there is no frame involved. The problem is somewhere in the parser, as it assumes INPUT is a function, not an option.

Yes, this is a really ambiguous part of the 4GL (and thus also in our parser).

See 3262a revision 11184 for a fix.

#95 - 10/24/2017 03:29 PM - Constantin Asofiei

Eugenie, I need some details about how BY-POINTER and BY-VARIANT-POINTER work... I assume BY-POINTER would tell the native-side to interpret whatever value it receives as a pointer address? And BY-VARIANT-POINTER should be a pointer to an array?

#96 - 10/24/2017 04:35 PM - Eugenie Lyzenko

Constantin Asofiei wrote:

> Eugenie, I need some details about how BY-POINTER and BY-VARIANT-POINTER work... I assume BY-POINTER would tell the native-side to interpret whatever value it receives as a pointer address?

Exactly, this is the address of the variable that storing the data.

> And BY-VARIANT-POINTER should be a pointer to an array?

No. This is VARIANT type variable that contains the pointer to another VARIANT that stores the variable. VARIANT is flexible data type, it can hold the pointers, simple types, ... and arrays too.

#97 - 10/24/2017 11:26 PM - Eugenie Lyzenko

Task branch 3262a has been updated to revision 11185.

This introduces first base work for native COM support implementation. The generic source functions schema implemented. The creation JNI partially implemented, Release implemented, COM init implemented.

Not clear for now how to get the object string name from object pointer.

The next day work will be to complete properties get/set, methods calls and finish remaining native part.

Now planning to do rebase with recent trunk in 5 min if nobody objects.

#98 - 10/24/2017 11:35 PM - Eugenie Lyzenko

Task branch 3262a has been rebased with trunk 11182, new revision is 11186.

#99 - 10/25/2017 03:39 AM - Constantin Asofiei

Eugenie Lyzenko wrote:

> Constantin Asofiei wrote:
>
> > Eugenie, I need some details about how BY-POINTER and BY-VARIANT-POINTER work... I assume BY-POINTER would tell the native-side to interpret whatever value it receives as a pointer address?
>
> Exactly, this is the address of the variable that storing the data.
>
> > And BY-VARIANT-POINTER should be a pointer to an array?
>
> No. This is VARIANT type variable that contains the pointer to another VARIANT that stores the variable. VARIANT is flexible data type, it can hold the pointers, simple types, ... and arrays too.

OK, then I think I need to send, beside the parameter mode (INPUT, OUTPUT, INPUT-OUTPUT), the BY-POINTER/BY-VARIANT-POINTER and any AS data-type, too... otherwise, I don't think the native side will know how to interpret the received value.

So, how do you prefer to do this? Send a ComParameter instance? Each argument as an array with 5 elements, as (value, mode, by-pointer, by-variant-pointer, AS data-type)? Something else?

#100 - 10/25/2017 03:50 AM - Constantin Asofiei

Eugenie, something else: can you describe what happens when a COM object fires an EVENT?

1. is it a push model, where the COM event pushes the event details to the 4GL side, and this event is processed when the next event processing loop (i.e. wait-for) is reached?

2. what is invoked on 4GL side? What name must follow the 4GL procedure invoked? How are parameters resolved?

#101 - 10/25/2017 07:07 AM - Eugenie Lyzenko

Constantin Asofiei wrote:

> OK, then I think I need to send, beside the parameter mode (INPUT, OUTPUT, INPUT-OUTPUT), the BY-POINTER/BY-VARIANT-POINTER and any AS data-type, too... otherwise, I don't think the native side will know how to interpret the received value.
>
> So, how do you prefer to do this? Send a ComParameter instance? Each argument as an array with 5 elements, as (value, mode, by-pointer, by-variant-pointer, AS data-type)? Something else?

In ideal case the native side needs to know:

1. The data passing mode to use(by-value, by-pointer or by-variant-pointer). In any case the 4GL makes this assumption even omitting the modifiers in 4GL code. Default mode is used for example simple parameters passing by values, complex - by pointer, arrays or memory pointers or com-handle objects(com-handle is a pointer to objects instance).

2. The 4GL data type in use is a good info to know. Some types(INT64 or DECIMAL may require additional efforts to keep precision or data range support).

#102 - 10/25/2017 08:37 AM - Eugenie Lyzenko

Constantin Asofiei wrote:

> Eugenie, something else: can you describe what happens when a COM object fires an EVENT?
> 1. is it a push model, where the COM event pushes the event details to the 4GL side, and this event is processed when the next event processing loop (i.e. wait-for) is reached?
> 2. what is invoked on 4GL side? What name must follow the 4GL procedure invoked? How are parameters resolved?

I do not think it is "classic" push model. Instead MS uses "callback" approach by registering procedures to be executed when event is coming. For example if we need to handle "Event1" the procedure must be written in 4GL:

```
PROCEDURE eventHandler.Event1:
  DO SOMETHING...
END PROCEDURE.
```

Then comObject.enable-events("eventHandler") is calling to start tracking of the "Event1".

Having the Event1 arrived the native COM object directly calls the procedure in 4GL.

From the our COM native module perspective I think we need to know:
1. COM object ID/pointer
2. Array of the procedure names(like Event1, Event2, ...)
3. Respective array of the procedure address for Event1, Event2, ...

The question is: if it is possible to execute Java code from external native process by only address? Or we need to implement some intermediate level to transform COM style events to our regular event processing loop(like a key processing handler).

#103 - 10/25/2017 08:47 AM - Eugenie Lyzenko

Constantin,

One question for JNI to implement. The call ComOleHelper.getName(long comId) is used to get the string name of the given (comId) object used while the COM object was created, correct? What is the purpose of this call? Asking because the implementation of this mapping can be complicated. Do you really need this JNI? It can be simpler to keep this map on the Java level just after the COM object is creating. Looks like the running COM object does not keep neither GUID nor string name inside after creation so I will have to keep dynamic array of this mapping within native FWD module if this JNI is mandatory.

#104 - 10/25/2017 08:54 AM - Greg Shah

> Respective array of the procedure address for Event1, Event2, ...

The internal procedures that get executed will be run on the server and they are Java code. There is no procedure address or function pointer that can be provided.

The native code will have to implement the callback signature and then when called, it will have to gather the parameters, send an event notification up to Java where it will get processed and then any return value will be passed back to native code where it can be returned to COM.

> I do not think it is "classic" push model. Instead MS uses "callback" approach by registering procedures to be executed when event is coming.

Constantin's question about WAIT-FOR is important. Does there have to be a WAIT-FOR that is active for the 4GL event processing to be executed? If yes, what happens when an event fires and there is no WAIT-FOR?

#105 - 10/25/2017 09:14 AM - Eugenie Lyzenko

Greg Shah wrote:

> Constantin's question about WAIT-FOR is important. Does there have to be a WAIT-FOR that is active for the 4GL event processing to be executed?

No, 4GL does not need the active WAIT-FOR to receive events.

> If yes, what happens when an event fires and there is no WAIT-FOR?

This is regular case(no WAIT-FOR and COM events firing). The only thing is required for event EventName is having handlerPrefix.EventName procedure defined in 4GL code.

COM events are not the same as any other 4GL events.

#106 - 10/25/2017 09:38 AM - Ovidiu Maxiniuc

Regarding the events / callbacks functions.
I have already implemented the callback call for PSTimer. This is a particular case of event, with no parameters and no return value. I run a few testcases and this work. I think this is a starting point for implementing all other events: once the COM event notification reaches ComServer (or similar object) we marshal the parameter, resolve the handling converted procedure (set by ADD-EVENTS-PROCEDURE) and dispatch the event. Once this generic mechanism works, we can use it back in PSTimer to avoid duplicating code.

#107 - 10/25/2017 09:52 AM - Constantin Asofiei

Ovidiu Maxiniuc wrote:

> Regarding the events / callbacks functions.
> I have already implemented the callback call for PSTimer. This is a particular case of event, with no parameters and no return value. I run a few testcases and this work. I think this is a starting point for implementing all other events: once the COM event notification reaches ComServer (or similar object) we marshal the parameter, resolve the handling converted procedure (set by ADD-EVENTS-PROCEDURE) and dispatch the event. Once this generic mechanism works, we can use it back in PSTimer to avoid duplicating code.

The PSTimer implementation is a little inversed, as the event is raised from server-side, not client-side. See PSTimer.fire and ComServer.emit which use LogicalTerminal.postServerEvent. For native COM objects, we are on client-side and we must use a different approach.

#108 - 10/25/2017 09:53 AM - Constantin Asofiei

Eugenie Lyzenko wrote:

> Constantin Asofiei wrote:
>
> > OK, then I think I need to send, beside the parameter mode (INPUT, OUTPUT, INPUT-OUTPUT), the BY-POINTER/BY-VARIANT-POINTER and any AS data-type, too... otherwise, I don't think the native side will know how to interpret the received value.
> >
> > So, how do you prefer to do this? Send a ComParameter instance? Each argument as an array with 5 elements, as (value, mode, by-pointer, by-variant-pointer, AS data-type)? Something else?
>
> In ideal case the native side needs to know:
>
> 1. The data passing mode to use(by-value, by-pointer or by-variant-pointer). In any case the 4GL makes this assumption even omitting the modifiers in 4GL code. Default mode is used for example simple parameters passing by values, complex - by pointer, arrays or memory pointers or com-handle objects(com-handle is a pointer to objects instance).
>
> 2. The 4GL data type in use is a good info to know. Some types(INT64 or DECIMAL may require additional efforts to keep precision or data range support).

OK, then the arguments will be an array of ComParameter objects which will hold all the relevant information about this parameter received from FWD side.

#109 - 10/25/2017 09:56 AM - Constantin Asofiei

Eugenie Lyzenko wrote:

> Constantin,
>
> One question for JNI to implement. The call ComOleHelper.getName(long comId) is used to get the string name of the given (comId) object used while the COM object was created, correct? What is the purpose of this call? Asking because the implementation of this mapping can be complicated. Do you really need this JNI? It can be simpler to keep this map on the Java level just after the COM object is creating. Looks like the running COM object does not keep neither GUID nor string name inside after creation so I will have to keep dynamic array of this mapping within native FWD module if this JNI is mandatory.

The only place where is really needed is if a COM property or method call returns a COM object. In this case, we need to create a new NativeComObject instance, which requires a name - it should work if a (com-id, name) mapping is kept on native side, but what if the COM object returned was NOT created by FWD?

#110 - 10/25/2017 10:04 AM - Constantin Asofiei

Eugenie Lyzenko wrote:

> Greg Shah wrote:
>
> > Constantin's question about WAIT-FOR is important. Does there have to be a WAIT-FOR that is active for the 4GL event processing to be executed?
>
> No, 4GL does not need the active WAIT-FOR to receive events.

Are you sure about this? Because it will break the 'single-threaded' paradigm for a 4GL client... did your tests ever stop for user key press (like a pause or anything else)? Usually any case which waits for user key press will eventually process all events.

#111 - 10/25/2017 10:05 AM - Constantin Asofiei

Eugenie, one more thing: for property getter and method call return value, I think I will need some kind of information of the data type of the value returned. So Java side requires both the value returned and its data-type, so I can wrap it into a proper FWD type.

#112 - 10/25/2017 11:05 AM - Eugenie Lyzenko

Constantin Asofiei wrote:

> No, 4GL does not need the active WAIT-FOR to receive events.

Are you sure about this?

Pretty much. The example: uast/com_ole_automation/events/event_with_param_integer.p.

> Because it will break the 'single-threaded' paradigm for a 4GL client... did your tests ever stop for user key press (like a pause or anything else)? Usually any case which waits for user key press will eventually process all events.

There is another opportunity. The 4GL is still single threaded but re-enterable with recursive call to itself.

#113 - 10/25/2017 11:26 AM - Constantin Asofiei

Eugenie Lyzenko wrote:

> There is another opportunity. The 4GL is still single threaded but re-enterable with recursive call to itself.

Please simulate a long-running code in your test, something like a loop from 1 to 1 million which calls substring or creates a string (something heave-duty, don't use PAUSE), log a message after the loop is finished and see if the COM event is processed by 4GL while the client executes this long-running code.

The idea here is that an event should never interrupt currently executing logic...

Also, are you executing uast/com_ole_automation/events/event_with_param_integer.p from the command line? If not, please try it from command line. Also, remove any UI statements from it (replace MESSAGE with writing something to file), so that the default window is never shown.

#114 - 10/25/2017 11:47 AM - Eugenie Lyzenko

Constantin Asofiei wrote:

> Eugenie, one more thing: for property getter and method call return value, I think I will need some kind of information of the data type of the value returned. So Java side requires both the value returned and its data-type, so I can wrap it into a proper FWD type.

Why can't you use BaseDataType abstract class to accept return value and call getTypeName() to identify particular data type?

Otherwise we will have to implement the VARIANT data type in FWD Java types set.

#115 - 10/25/2017 12:51 PM - Constantin Asofiei

Eugenie Lyzenko wrote:

> Constantin Asofiei wrote:
>
> > Eugenie, one more thing: for property getter and method call return value, I think I will need some kind of information of the data type of the value returned. So Java side requires both the value returned and its data-type, so I can wrap it into a proper FWD type.
>
> Why can't you use BaseDataType abstract class to accept return value and call getTypeName() to identify particular data type?

I was thinking for the native side to have as little knowledge of the FWD internals as possible. But if you want to build the FWD compatible value for method return or property value getter, I'm OK with it.

> Otherwise we will have to implement the VARIANT data type in FWD Java types set.

If 4GL side receives a pointer or variant as a return value or from an OUTPUT argument, I don't think that can be used without specifying BY-POINTER or BY-POINTER-VALUE, when passing this pointer/variant as a value to another argument for a COM object method call. Do you know otherwise?

#116 - 10/25/2017 01:17 PM - Eugenie Lyzenko

Constantin Asofiei wrote:

> Otherwise we will have to implement the VARIANT data type in FWD Java types set.

> If 4GL side receives a pointer or variant as a return value or from an OUTPUT argument, I don't think that can be used without specifying BY-POINTER or BY-POINTER-VALUE, when passing this pointer/variant as a value to another argument for a COM object method call. Do you know otherwise?

I would prefer not to introduce extra artificial data types into FWD. The idea is: if some data type does not exist in 4GL -> there is no counterpart in FWD for it. There is no such data type as VARIANT in original 4GL. So we are leaving this data within native code, making appropriate transformations to 4GL native data(meaning FWD implementations of them) before returning parameter option or method return values. I have seen so-called "Anydata" in 4G documentation but I think this is just a generic to define the possibility to use any data type(but some particular, INTEGER, or INT64, or LOGICAL, or ... and so on in every case).

#117 - 10/25/2017 01:32 PM - Constantin Asofiei

Eugenie Lyzenko wrote:

> Constantin Asofiei wrote:
>
> > Otherwise we will have to implement the VARIANT data type in FWD Java types set.
>
> > If 4GL side receives a pointer or variant as a return value or from an OUTPUT argument, I don't think that can be used without specifying BY-POINTER or BY-POINTER-VALUE, when passing this pointer/variant as a value to another argument for a COM object method call. Do you know otherwise?
>
> I would prefer not to introduce extra artificial data types into FWD. The idea is: if some data type does not exist in 4GL -> there is no counterpart in FWD for it. There is no such data type as VARIANT in original 4GL. So we are leaving this data within native code, making appropriate transformations to 4GL native data(meaning FWD implementations of them) before returning parameter option or method return values. I have seen so-called "Anydata" in 4G documentation but I think this is just a generic to define the possibility to use any data type(but some particular, INTEGER, or INT64, or LOGICAL, or ... and so on in every case).

Are you saying that you want to send the OUTPUT/INPUT-OUTPUT arguments as BDT instances, too? If so, I'm OK with the native side sending BDT values for:

1. method return value
2. property getter
3. OUTPUT/INPUT-OUTPUT arguments

When executing a method call, I plan to send each argument as a ComParameter instance which will have these fields:

1. mode - parameter mode, I for input, O for output and U for input-output

2. value - the unmarshaled argument value, from BDT to a Java type
3. byPointer - flag indicating if BY-POINTER option was used
4. byVariantPointer - flag indicating if BY-VARIANT-POINTER option was used
5. dataType - an enum which contains the AS data-type option
6. fwdDataType - the BDT class name for this value (if none, it was received as a Java type)

Something else: if a COM object is used by FWD (via a pointer or something else) and was not created by the 4GL business logic, can I use a native API to determine if the COM object is valid? How does 4GL behave?

#118 - 10/25/2017 01:46 PM - Eugenie Lyzenko

Constantin Asofiei wrote:

> When executing a method call, I plan to send each argument as a ComParameter instance which will have these fields:
>
> 1. mode - parameter mode, I for input, O for output and U for input-output
> 2. value - the unmarshaled argument value, from BDT to a Java type
> 3. byPointer - flag indicating if BY-POINTER option was used
> 4. byVariantPointer - flag indicating if BY-VARIANT-POINTER option was used
> 5. dataType - an enum which contains the AS data-type option
> 6. fwdDataType - the BDT class name for this value (if none, it was received as a Java type)

Yes, it will be good for native code to know the data type passing for property setter and method call input parameter. I think exploring common jobject looking for particular data type is too expensive for native code(if we can easily avoid this).

> Something else: if a COM object is used by FWD (via a pointer or something else) and was not created by the 4GL business logic, can I use a native API to determine if the COM object is valid?

What do you mean by **native API to determine if the COM object is valid**?

#119 - 10/25/2017 01:52 PM - Constantin Asofiei

Eugenie Lyzenko wrote:

> Constantin Asofiei wrote:
>
> > When executing a method call, I plan to send each argument as a ComParameter instance which will have these fields:

1. mode - **char** parameter mode, I for input, O for output and U for input-output
2. value - **Object** the unmarshaled argument value, from BDT to a Java type
3. byPointer - **boolean** flag indicating if BY-POINTER option was used
4. byVariantPointer - **boolean** flag indicating if BY-VARIANT-POINTER option was used
5. dataType - **ComParameter$DataType** an enum which contains the AS data-type option
6. fwdDataType - **String** the BDT class name for this value (if none, it was received as a Java type)

> Yes, it will be good for native code to know the data type passing for property setter and method call input parameter.

Do you mean you want to know the arguments Java type, too, if the argument was not a BDT (i.e. this was a INPUT argument sent 'by value')?

Also, see above for the Java type of each ComParameter field described. If you want me to change something, please let me know.

> Something else: if a COM object is used by FWD (via a pointer or something else) and was not created by the 4GL business logic, can I use a native API to determine if the COM object is valid?

> What do you mean by **native API to determine if the COM object is valid**?

I mean this: lets say there is a chained call like comVar:propertyReturningComHandle:someOtherProperty. How does FWD side know that propertyReturningComHandle returns a valid COM object instance? More, what if a COM object gets deleted by the native side? The idea is, I don't think is enough to rely just on a deleted flag on FWD side... there should be some native-side check if this COM object is still alive.

#120 - 10/25/2017 02:32 PM - Eugenie Lyzenko

Constantin Asofiei wrote:

> What do you mean by **native API to determine if the COM object is valid**?

> I mean this: lets say there is a chained call like comVar:propertyReturningComHandle:someOtherProperty. How does FWD side know that propertyReturningComHandle returns a valid COM object instance? More, what if a COM object gets deleted by the native side? The idea is, I don't think is enough to rely just on a deleted flag on FWD side... there should be some native-side check if this COM object is still alive.

4GL makes no assumption for validity of the COM object handle when using it. If the COM object is not valid the run-time error is displaying.

#121 - 10/25/2017 02:34 PM - Constantin Asofiei

Eugenie Lyzenko wrote:

> Constantin Asofiei wrote:
>
> > What do you mean by **native API to determine if the COM object is valid**?
>
> I mean this: lets say there is a chained call like comVar:propertyReturningComHandle:someOtherProperty. How does FWD side know that propertyReturningComHandle returns a valid COM object instance? More, what if a COM object gets deleted by the native side? The idea is, I don't think is enough to rely just on a deleted flag on FWD side... there should be some native-side check if this COM object is still alive.
>
> 4GL makes no assumption for validity of the COM object handle when using it. If the COM object is not valid the run-time error is displaying.

OK, so does VALID-HANDLE return false only if RELEASE was called on a comhandle var?

#122 - 10/25/2017 02:41 PM - Eugenie Lyzenko

Constantin Asofiei wrote:

> Eugenie Lyzenko wrote:
>
> > There is another opportunity. The 4GL is still single threaded but re-enterable with recursive call to itself.
>
> Please simulate a long-running code in your test, something like a loop from 1 to 1 million which calls substring or creates a string (something

heave-duty, don't use PAUSE), log a message after the loop is finished and see if the COM event is processed by 4GL while the client executes this long-running code.

The idea here is that an event should never interrupt currently executing logic...

Consider the following test:

```
define variable fwdAppl as com-handle no-undo.

create "IFwdTest.object" fwdAppl.

no-return-value fwdAppl:enable-events("eventHandler").

fwdAppl:fireEvent(1).
message "Event1 fired".
fwdAppl:fireEvent(2).
message "Event2 fired".

release object fwdAppl.

procedure eventHandler.Event1:
   message "Event1 has been received".
end procedure.

procedure eventHandler.eVeNt2:
   message "Event2 has been received".
end procedure.
```

The message ordering will be:

```
Event1 has been received
Event1 fired
Event2 has been received
Event2 fired
```

This certainly meaning single-threaded approach.

#123 - 10/25/2017 02:44 PM - Eugenie Lyzenko

Constantin Asofiei wrote:

> 4GL makes no assumption for validity of the COM object handle when using it. If the COM object is not valid the run-time error is displaying.

OK, so does VALID-HANDLE return false only if RELEASE was called on a comhandle var?

Or if the comhandle var was not yet initialized(created in terms of 4GL).

#124 - 10/25/2017 02:46 PM - Constantin Asofiei

Eugenie Lyzenko wrote:

> Constantin Asofiei wrote:
>
> > Eugenie Lyzenko wrote:
> >
> > > There is another opportunity. The 4GL is still single threaded but re-enterable with recursive call to itself.
> >
> > Please simulate a long-running code in your test, something like a loop from 1 to 1 million which calls substring or creates a string (something heave-duty, don't use PAUSE), log a message after the loop is finished and see if the COM event is processed by 4GL while the client executes this long-running code.
> >
> > The idea here is that an event should never interrupt currently executing logic...
>
> Consider the following test:
>
> [...]
>
> The message ordering will be:
> [...]
>
> This certainly meaning single-threaded approach.

OK, so looks like these are not real events, but more of a way to call back into the 4GL code. This means that when the COM object raises an event, it does not push it on an event queue, but it immediately calls the registered 4GL procedure associated with that event - so the COM method call and event are synchronous:

- when the event is 'fired', the COM method call stops and wait for this event to finish
- when the 4GL procedure associated with this event has finished, the COM method continues execution.

I was confused by naming this as an 'event', which for me means it needs to be posted on an event queue, etc.

#125 - 10/25/2017 02:50 PM - Greg Shah

What about the firing of these events from native code? For example, if something happens in the COM object itself that fires the "event" (instead of 4GL code firing it), does that trigger execution of the 4GL callback immediately?

#126 - 10/25/2017 03:09 PM - Constantin Asofiei

Greg Shah wrote:

> What about the firing of these events from native code? For example, if something happens in the COM object itself that fires the "event" (instead of 4GL code firing it), does that trigger execution of the 4GL callback immediately?

It might be possible for 4GL runtime to know the control is now at a COM object method call, and invoke the event immediately... and otherwise, if control is on 4GL side, to post it on the event queue.

#127 - 10/25/2017 04:14 PM - Constantin Asofiei

Eugenie, the Java part (at least high-level design) is in 3262a. Unknowns at this time:

1. how will we pass extent arguments? Is it OK to make the ComParameter.value a Java array?
2. how does Java side receive extent output/input-output arguments? Is it OK to expected a BDT[] from the native side, for that argument? If so, some signatures in ComOleHelper need to be changed.
3. error handling for property setter/getter, method calls
4. parameter validation for method calls - I assume this will be done on the native side
5. property setter value validation - I assume this will be done on the native side
6. what happens if the event procedure does not exist? Is there an error shown or just silently ignored?
7. what happens if RETURN ERROR during event procedure execution, by 4GL side?
8. does the COM event access the RETURN-VALUE from the procedure call?

#128 - 10/25/2017 04:55 PM - Eugenie Lyzenko

Constantin Asofiei wrote:

> Do you mean you want to know the arguments Java type, too, if the argument was not a BDT (i.e. this was a INPUT argument sent 'by value')?
>
> Also, see above for the Java type of each ComParameter field described. If you want me to change something, please let me know.

...

> 2. value - Object the unmarshaled argument value, from BDT to a Java type

Do you mean int, long, boolean as "Java type"? What is the size of the value to store data? 64-bit or 32-bit?

What about a bit different approach for input parameters. We pass two arrays of simple types, one is long array with 32/64 bit value that can represent any possible variable including pointer to array, string and memory location. The other array is integer data type values for respective entries in the

first array. In this case we do not need to pass Object data type to native code and do not need to make access to Java code from native. Isn't it be better?

Having ComParameter is not a marshalling but making abstraction to put all real work down to native code. We have simple data type, create one more Java class and want the native code to simplify this Java class into simple data types again.

#129 - 10/25/2017 05:19 PM - Constantin Asofiei

Eugenie Lyzenko wrote:

> Constantin Asofiei wrote:
>
> > Do you mean you want to know the arguments Java type, too, if the argument was not a BDT (i.e. this was a INPUT argument sent 'by value')?
> >
> > Also, see above for the Java type of each ComParameter field described. If you want me to change something, please let me know.
>
> ...
>
> > 2. value - Object the unmarshaled argument value, from BDT to a Java type
>
> Do you mean int, long, boolean as "Java type"? What is the size of the value to store data? 64-bit or 32-bit?

I mean these exact types or values (as I coded them in ComObject.unmarshal):

1. null value for unknown FWD values
2. java.lang.String - for character or longchar
3. java.lang.Boolean - for logical
4. java.lang.Integer - for integer
5. java.lang.Long - for int64, memptr pointer address, comhandle's comId value,
6. java.lang.Double - for decimal
7. java.util.Date - for date, datetime, datetime-tz
8. byte[] array for raw

For all the cases above, the fwdDataType will be a BDT class name. I haven't touched yet extent variables, as these are complex on their own.

Also, there are these cases where INPUT parameters can be sent by the business logic as literals:

1. java.lang.String will have fwdDataType as character
2. java.lang.Boolean will have fwdDataType as logical
3. java.lang.Integer will have fwdDataType as integer
4. java.lang.Long will have fwdDataType as int64
5. java.lang.Double and java.lang.Float will have fwdDataType as decimal
6. java.util.Date will have fwdDataType as date

> What about a bit different approach for input parameters. We pass two arrays of simple types, one is long array with 32/64 bit value that can represent any possible variable including pointer to array, string and memory location.

I don't understand this one - how can FWD send a pointer for a string, to native side, when FWD works with java.lang.String?

> The other array is integer data type values for respective entries in the first array. In this case we do not need to pass Object data type to native code and do not need to make access to Java code from native. Isn't it be better?

Here too I'm confused by what you mean...

> Having ComParameter is not a marshalling but making abstraction to put all real work down to native code. We have simple data type, create one more Java class and want the native code to simplify this Java class into simple data types again.

No, input parameters can be either BDT sub-classes or Java types as explained above. And ComParameter wants to encapsulate all this information so that the native side can handle them in more common way. Also, remember there is the:

1. AS data-type which can be SHORT, FLOAT, CURRENCY, UNSIGNED_BYTE, ERROR_CODE or IUNKNOWN - these have no significance on the 4GL side data-types... I don't know the behaviour of this option, so I assume the native side will do something with that value, depending on the AS datatype clause.
2. BY-POINTER or BY-VARIANT-POINTER options - the Java side can't 'transform' an i.e. long value to a pointer, the native side must know that the caller intended this value to be a pointer and interpret it as such.

If the native part is close to have simple COM examples work in FWD, please go ahead and change the Java code as you are more comfortable.

#130 - 10/25/2017 11:52 PM - Eugenie Lyzenko

Task branch 3262a has been updated to revision 11191.

This adds more functionality in native code. Including major code to property setters and getters. The internal functions has been added to pack/unpack data from Java objects coming from Java side to VARIANT aggregate will be used in native code to map the data. Also added implementation for COM handle verification and getting name of the active COM object currently in use by 4GL.

The next step is to implement complex data handling and to add COM object method processing.

And some minor Javadoc fixes in related and unrelated code.

#131 - 10/26/2017 12:12 AM - Eugenie Lyzenko

Constantin Asofiei wrote:

> Eugenie Lyzenko wrote:
> ...
> I don't understand this one - how can FWD send a pointer for a string, to native side, when FWD works with java.lang.String?

Java pass String as jstring identifier. Although it is not a regular pointer(like char*) it can be packed into 64-bit integer I guess.

> Here too I'm confused by what you mean...

I mean consider setProperty(..., jlongArray varArr, jintArray typeArr, ...). For variable i the type is typeArr[i] which can be (int, double, bool) for (0, 1, 2) type values. So we will know type and can cast varArr[i] to respective data type.

For now I'm trying to use the current implementation, so anyway no need to change Java side approach - I'll adapt the current one due to time limitations.

#132 - 10/26/2017 12:05 PM - Eugenie Lyzenko

Constantin,

Consider the following method in ComParameter:

```
public boolean isInputOutput()
{
   return 'I' == mode;
}
```

According to your design the INPUT-OUTPUT mode is 'U' char so the right method is:

```
public boolean isInputOutput()
{
   return 'U' == mode;
}
```

Correct? Please confirm or decline.

#133 - 10/26/2017 12:07 PM - Constantin Asofiei

Eugenie Lyzenko wrote:

> Constantin,
>
> Consider the following method in ComParameter:
> [...]
>
> According to your design the INPUT-OUTPUT mode is 'U' char so the right method is:
> [...]
>
> Correct? Please confirm or decline.

Yes, that's correct.

#134 - 10/26/2017 01:55 PM - Constantin Asofiei

Eugenie, if you want me to help with something specific on the Java side, please let me know. Otherwise I'll look into what would be needed for extent arguments.

#135 - 10/26/2017 02:18 PM - Eugenie Lyzenko

Constantin Asofiei wrote:

> Eugenie, if you want me to help with something specific on the Java side, please let me know. Otherwise I'll look into what would be needed for extent arguments.

Thank you, I'm preparing some final steps(including comments added). I made generic implementation of the method JNI call. Now I need to finish all possible Java Object <-> VARIANT transformations. While I'm doing this work I need some of your feedback for current Java usage in JNI (there is a lot of Java object usage call).

I'm going to commit the current change in a 5 min.

#136 - 10/26/2017 02:29 PM - Constantin Asofiei

Eugenie, something else: please make sure to test comHandleVar BY-POINTER arguments in method calls, because this is used in the project we are currently working on.

#137 - 10/26/2017 02:32 PM - Eugenie Lyzenko

Constantin Asofiei wrote:

> Eugenie, something else: please make sure to test comHandleVar BY-POINTER arguments in method calls, because this is used in the project we are currently working on.

OK. As a method call parameter, correct?

#138 - 10/26/2017 02:37 PM - Eugenie Lyzenko

Task branch 3262a has been updated for review to revision 11192.

The COM method call implementation added.

#139 - 10/26/2017 02:39 PM - Constantin Asofiei

Eugenie Lyzenko wrote:

> Constantin Asofiei wrote:

> Eugenie, something else: please make sure to test comHandleVar BY-POINTER arguments in method calls, because this is used in the project we are currently working on.

> OK. As a method call parameter, correct?

Yes, with INPUT mode.

#140 - 10/26/2017 02:46 PM - Eugenie Lyzenko

One more question. Is it possible to change the ComOleHelper call?

From:

```
public static native Long create
```

to:

```
public static native long create
```

Is it important to have Long object class instead of simple jlong?

#141 - 10/26/2017 02:47 PM - Constantin Asofiei

Eugenie Lyzenko wrote:

> One more question. Is it possible to change the ComOleHelper call?
>
> From:
> [...]
> to:
> [...]
>
> Is it important to have Long object class instead of simple jlong?

If I can assume that 0 means create has failed, then it's OK.

#142 - 10/26/2017 03:20 PM - Eugenie Lyzenko

Constantin Asofiei wrote:

> Eugenie Lyzenko wrote:
>
> > One more question. Is it possible to change the ComOleHelper call?
> >
> > From:
> > [...]
> > to:
> > [...]
> >
> > Is it important to have Long object class instead of simple jlong?
>
> If I can assume that 0 means create has failed, then it's OK.

It is a memory object pointer and can not be 0. Accessing to object by address 0 is bad idea for application, so 0 means the COM object was not created.

#143 - 10/26/2017 03:29 PM - Constantin Asofiei

Eugenie Lyzenko wrote:

> It is a memory object pointer and can not be 0. Accessing to object by address 0 is bad idea for application, so 0 means the COM object was not created.

OK, I'm making this change and will commit it.

#144 - 10/27/2017 12:32 AM - Eugenie Lyzenko

Task branch 3262a has been updated for review to revision 11195.

Added the Java<->COM data transformations for property set/get and methods call. Improved the COM object creator to consider all create/connect options. Also COM subsystem termination function added.

Things to do:
1. Check COM-HANDLE passing approach, the bugs are possible there.
2. There is a memory leak now with string allocation. Need to add proper cleanup utility.
3. Need to find a ritght way to call general COM termination to clean up resources allocated by object registry and COM usage.

#145 - 10/27/2017 11:00 AM - Eugenie Lyzenko

Constantin,

Please help me to understand the COM-HANDLE FWD internals related to long comID we use to identify COM object. To get ID to store as parameter for COM method/property setter I need to:

1. Have the FWD p2j/util/comhandle class object.
2. Call comhandle.getResource() to get the class object ComObject
3. Call ComObject.id() to get java/util/Long class object for ID.
4. Call Long.longValue() to get reqiured long ID to use in native code as object identifier.

Is it correct sequence?

#146 - 10/27/2017 11:11 AM - Constantin Asofiei

Eugenie Lyzenko wrote:

> Constantin,
>
> Please help me to understand the COM-HANDLE FWD internals related to long comID we use to identify COM object. To get ID to store as parameter for COM method/property setter I need to:

See bellow for the correct steps:
1. Have the FWD p2j/util/comhandle class object.
2. Call comhandle.getResource() to get the class object ComObject
3. Cast ComObject to NativeComObject - if this is not a NativeComObject, abend, as it can't be used on native side
4. Call NativeComObject.getComId() to get a long value for COM object ID

If you want to send a COM object from the native side, then build a comhandle instance with its resource set to a new NativeComObject(long comId) object.

#147 - 10/27/2017 12:25 PM - Eugenie Lyzenko

Constantin Asofiei wrote:

> Eugenie Lyzenko wrote:
>
> > Constantin,
> >
> > Please help me to understand the COM-HANDLE FWD internals related to long comID we use to identify COM object. To get ID to store as parameter for COM method/property setter I need to:
>
> See bellow for the correct steps:
> 1. Have the FWD p2j/util/comhandle class object.
> 2. Call comhandle.getResource() to get the class object ComObject
> 3. Cast ComObject to NativeComObject - if this is not a NativeComObject, abend, as it can't be used on native side
> 4. Call NativeComObject.getComId() to get a long value for COM object ID
>
> If you want to send a COM object from the native side, then build a comhandle instance with its resource set to a new NativeComObject(long comId) object.

OK. I'm going to commit mys recent changes in a 5 min. if you do not mind.

#148 - 10/27/2017 12:41 PM - Eugenie Lyzenko

Task branch 3262a has been updated for review to revision 11196.

Changed COM-HANDLE ID processing to exchange Java<->COM. Added clean up tool to fix the memory leak caused by SysAllocString() call.

Now trying to understand how the DATE data type is implemented in native C code. Looks like we will have to scan MinGW package headers for any explanations.

#149 - 10/27/2017 05:28 PM - Eugenie Lyzenko

I have next drop to commit for 3262a. Planning to do this in a 5 min if nobody objects.

#150 - 10/27/2017 05:54 PM - Eugenie Lyzenko

Task branch 3262a has been updated for review to revision 11197.

Added handling for DATE variables exchange from Java to COM and back. Also now the character based data are handles as Unicode strings which is native format to COM.

Next planning to implement consideration of the helper enum DataType introduced in Java for AS datatype 4GL clause. This can help to make better transformation for Java->COM part.

#151 - 10/27/2017 06:38 PM - Eugenie Lyzenko

Of course DataType is enum, not union.

#152 - 10/27/2017 09:14 PM - Eugenie Lyzenko

Task branch 3262a has been updated for review to revision 11198.

Added usage of DataType helper option support. For now it is used to convert DECIMAL value to CY COM data type. Preparing to rebase with recent trunk. So please do not make any update until my rebase complete note.

#153 - 10/27/2017 09:38 PM - Eugenie Lyzenko

Task branch 3262a has been rebased with trunk 11183, new revision is 11199.

So the currently requested JNI are ready. Planning to do some testing for samples used with IFwdTest COM object. Because we have got the level to verify already implemented code I think. Correct me if I'm wrong but for now the COM event processing is not supported in FWD.

#154 - 10/28/2017 10:45 AM - Constantin Asofiei

Eugenie Lyzenko wrote:

> Correct me if I'm wrong but for now the COM event processing is not supported in FWD.

COM events can be fired via the ComOleHelper.fire API - you can change the signature of this API as you need.

#155 - 10/28/2017 04:19 PM - Eugenie Lyzenko

Going to rebase 3262a in a 5 min of nobody objects.

#156 - 10/28/2017 04:37 PM - Eugenie Lyzenko

Task branch 3262a has been rebased with trunk 11184, new revision is 11200.

#157 - 10/29/2017 09:19 AM - Eugenie Lyzenko

Looks like the Java part is OK (creation, property get/set and method call converted correctly). While the window native module changes are highly possible(I've found at least one change).

So my offering is to start regression testing now. Meantime I will finalize Windows specific changes(we can commit them later just because Windows module is out of the egression testing area). What do you think?

#158 - 10/29/2017 04:05 PM - Eugenie Lyzenko

Constantin,

I have the changes for ComParameter class, two method to write/read the class over network:

```
...
   public void writeExternal(ObjectOutput out)
   throws IOException
   {
      out.writeChar(mode);
      out.writeObject(value);
      out.writeBoolean(byPointer);
      out.writeBoolean(byVariantPointer);
      out.writeInt(dataType != null ? dataType.ordinal() : -1);
      out.writeObject(fwdDataType);
   }
...
   public void readExternal(ObjectInput in)
   throws IOException,
          ClassNotFoundException
   {
      mode = in.readChar();
      value = in.readObject();
      byPointer = in.readBoolean();
      byVariantPointer = in.readBoolean();
      int dtOrdinal = in.readInt();
      if (dtOrdinal != -1)
      {
         dataType = DataType.values()[dtOrdinal];
      }
      fwdDataType = (String) in.readObject();
   }
...
```

Without this checking I have NPE because sometimes the dataType id NULL (not using as-datatype option) and we have the abend in this condition.

Do you agree with this change? Or have something better to implement?

#159 - 10/29/2017 09:11 PM - Greg Shah

> So my offering is to start regression testing now. Meantime I will finalize Windows specific changes(we can commit them later just because Windows module is out of the egression testing area). What do you think?

Yes, this makes sense. Please go ahead.

#160 - 10/29/2017 09:15 PM - Eugenie Lyzenko

Greg Shah wrote:

> Yes, this makes sense. Please go ahead.

OK. I'm planning to make the branch update soon(including one Java class). Then start testing.

Do we need the conversion tests to run?

#161 - 10/29/2017 09:46 PM - Eugenie Lyzenko

Task branch 3262a has been updated for review to revision 11201.

Bug fixing in native code and ComParameter java class. Starting the regression testing.

#162 - 10/29/2017 10:57 PM - Greg Shah

> Do we need the conversion tests to run?

Yes, the progress.g and com_access.rules changes are enough to require it.

#163 - 10/30/2017 06:59 AM - Eugenie Lyzenko

Greg Shah wrote:

> > Do we need the conversion tests to run?

> Yes, the progress.g and com_access.rules changes are enough to require it.

OK.

#164 - 10/30/2017 07:17 AM - Eugenie Lyzenko

Task branch 3262a has been rebased with trunk 11185, new revision is 11202.

The testings restarted.

#165 - 10/30/2017 09:44 AM - Constantin Asofiei

I've been looking through how COM are used and I found these:

1. one usage of com:enable-events - so we need event support. Eugenie, can you intercept the event from the native side and delegate the call to ComOleHelper.fire? This would work only if the native code which intercepts this event knows the event name...
2. two usages of a HWND attribute passed as method argument, like in comVar:comMethod(CURRENT-WINDOW:HWND).

#166 - 10/30/2017 09:50 AM - Greg Shah

> two usages of a HWND attribute passed as method argument, like in comVar:comMethod(CURRENT-WINDOW:HWND).

These must have been added recently. These are invalid and will need to be removed. It simply won't work in the converted code.

#167 - 10/30/2017 09:52 AM - Eugenie Lyzenko

Greg Shah wrote:

> > two usages of a HWND attribute passed as method argument, like in comVar:comMethod(CURRENT-WINDOW:HWND).
>
> These must have been added recently. These are invalid and will need to be removed. It simply won't work in the converted code.

Widget handles can not be used as COM parameters for method or property set/get.

#168 - 10/30/2017 09:54 AM - Greg Shah

The HWND is not a widget handle. It is the actual WIN32 HWND that matches the OpenEdge window-widget's WIN32 native window. FWD has a completely different implementation with no native WIN32 HWND available.

#169 - 10/30/2017 10:15 AM - Constantin Asofiei

Eugenie, another question: is enable-events an API which can be used by COM OLE automation regardless of 4GL environment (i.e. there is an equivalent COM OLE method which is called, in the native code)? Or this is a 4GL-specific API?

#170 - 10/30/2017 10:36 AM - Eugenie Lyzenko

Constantin Asofiei wrote:

> Eugenie, another question: is enable-events an API which can be used by COM OLE automation regardless of 4GL environment (i.e. there is an equivalent COM OLE method which is called, in the native code)? Or this is a 4GL-specific API?

To be clear:
1. enable-events() is the 4GL API that uses COM object feature to set up events handler as callbacks.
2. To make this work we really need to intercept COM events in native call(and to impement JNI ComOleHelper.enableEvent())
3. This is complex task. Because at the time of the enable-events() call we need to have handler's procedure addresses for every handler we want to be called.

I see the following possible approach(it is still under design thinking):
1. When enable-events() arriving to native call we must have the COM object ID as input parameter.
2. For this COM we enumerate all events the native com can fire and create interface handler template in native code. Then install this template as callback handler. Note we will have to know all details about callback(input params mode number and type, return type...)
3. When COM event fired, our native handler will be called. Then and only then we can emit the respective procedure call for our FWD code.

If you see another opportunity - let me know. I think it can not be implemented today due to complexity. We can just avoid this call(enable-events()) in target application now.

#171 - 10/30/2017 10:39 AM - Constantin Asofiei

Eugenie Lyzenko wrote:

> If you see another opportunity - let me know. I think it can not be implemented today due to complexity. We can just avoid this call(enable-events()) in target application now.

The enable-events single argument is a prefix for an internal procedure, right? And the procedure associated with the event must be named somePrefix.EventName, correct?

If above is correct, I can't find in the source code any internal procedure with the above syntax. So yes, is OK to postpone this.

#172 - 10/30/2017 10:43 AM - Eugenie Lyzenko

Constantin Asofiei wrote:

> Eugenie Lyzenko wrote:
>
> > If you see another opportunity - let me know. I think it can not be implemented today due to complexity. We can just avoid this call(enable-events()) in target application now.
>
> The enable-events single argument is a prefix for an internal procedure, right? And the procedure associated with the event must be named somePrefix.EventName, correct?

Exactly. The 4gl code muste be:

```
enable-events("prefix").

procedure prefix.event1:
end procedure.
```

#173 - 10/30/2017 10:45 AM - Eugenie Lyzenko

Greg Shah wrote:

> Yes, the progress.g and com_access.rules changes are enough to require it.

The conversion testing completed. Generated code are identical. Continue with runtime testing.

#174 - 10/30/2017 11:29 AM - Eugenie Lyzenko

Another issue in Java code:
ComObject.unmarshal() produces NPE for empty input parameters for method call. like obj.About():

```
   protected static ComParameter[] unmarshal(Object[] aVals)
   {
```

```
      ComParameter[] result = new ComParameter[aVals.length]; <--- (aVals == null)
...
```

#175 - 10/30/2017 11:31 AM - Constantin Asofiei

Eugenie Lyzenko wrote:

> Another issue in Java code:
> ComObject.unmarshal() produces NPE for empty input parameters for method call. like obj.About():
> [...]

Eugenie, please fix these as you find them.

#176 - 10/30/2017 01:09 PM - Eugenie Lyzenko

Constantin Asofiei wrote:

> Eugenie, please fix these as you find them.

Task branch 3262a has been updated for review to revision 11204.

Added fix for Java code NPE in ComObject.

The CTRL-C testing completed without errors.

Constantin, have you made commit for 3262a?:

```
 M  rules/convert/language_statements.rules
All changes applied successfully.
Updated to revision 11203
```

So I will have to re-run conversion test again, right?

#177 - 10/30/2017 01:09 PM - Constantin Asofiei

Eugenie Lyzenko wrote:

> Constantin, have you made commit for 3262a?:
> [...]
>
> So I will have to re-run conversion test again, right?

The commit is safe, it just adds import for p2j.comauto package in certain cases. So you don't need to reconvert.

#178 - 10/30/2017 05:18 PM - Eugenie Lyzenko

The status of the 3262a branch:

1. The first main cycle of the testing completed. The results are prettu good. But there are the failures and I'm going to start another round after DB restore.
2. The native module is close to be finished for some stable state. The method call and the property (set|get) is working. The issues investigated and root causes are now fixed, need to make respective changed in all currently supported data types.

So my plan is:
1. While the next regression round completes I'm finalizing the native module(this is 3.5 hours for test round to complete). I do not expect Java code changes anymore.
2. Once main regression cycle is OK(CTRL-C is already done with positive result) - I will do rebase for 3262a and merge into trunk.

Is it acceptable plan?

#179 - 10/30/2017 05:38 PM - Greg Shah

> Is it acceptable plan?

Yes.

#180 - 10/30/2017 09:48 PM - Eugenie Lyzenko

Testing completed. No regressions have found. The results: 3262a_11204_a5eeab0_20171030_evl.zip.

So planning to prepare code to rebase, rebase with recent trunk and merge final 3262a update into trunk. I guess it can take 30-40 min due to massive updates performed in trunk since recent rebase.

#181 - 10/30/2017 09:51 PM - Greg Shah

Good plan.

#182 - 10/30/2017 10:56 PM - Eugenie Lyzenko

Task branch 3262a has been updated for review to revision 11205.

This is bug fixing and cleanup for COM automation implementation initial release.

Starting the rebase and merge into trunk.

#183 - 10/30/2017 11:24 PM - Eugenie Lyzenko

Task branch 3262a has been rebased with trunk 11189, new revision is 11209.

Starting merge into trunk.

#184 - 10/30/2017 11:32 PM - Eugenie Lyzenko

Branch 3262a was merged to trunk as revno 11190 then it was archived.

Sorry it has taken more time I have expected. A lot of my debug hooks to remove for release version.

#185 - 10/31/2017 12:09 AM - Eugenie Lyzenko

Now I would like to note about some limitations and known issues:

1. As we discussed - no events firing support from COM

2. Working with COM-HANDLE as parameter getter causes the strange exception:
NotSerializableException in comhandle:

```
   public void readExternal(ObjectInput in)
   throws IOException,
          ClassNotFoundException
   {
      if (!(value instanceof Externalizable))
      {
         throw new NotSerializableException("comhandle"); <---
...
```

This is strange bcause NativeConObject implements Externalizable.

3. I need to clarify the NativeComObject:

```
   public void readExternal(ObjectInput in)
   throws IOException
   {
      comId = in.readInt();
   };
```

The comId is long. Why do not we use comId = in.readLong(); here? Although the does not affect the current COM functionality.

4. The MS Windows API to convert string to date and back works bad. Or needs another string format to use. We need to make additional investigation for better approach for storing Date variable in COM object.

5. And finally a big surprise for 64-bit Windows users. I'm not 100% sure however having deep suggestion the Java has one bug in Windows package. The long value can not be passed from Java to JNI without losses. Only 32-bit portion of long is handling correct for 64-bit long. This is another argument BTW to think the COM subsystem is 32-bit aware even for 64-bit OS. Just because passing pointer to COM as long we can call the object methods - this means the pointer is valid and this means it can be passed via 32-bit int.

#186 - 10/31/2017 11:18 AM - Greg Shah

> I'm not 100% sure however having deep suggestion the Java has one bug in Windows package. The long value can not be passed from Java to JNI without losses. Only 32-bit portion of long is handling correct for 64-bit long.

I don't think this is a limitation. We pass 64-bit memory addresses to/from JNI for memptr support and also for library calls. Perhaps it is worth looking at those examples to see how it may be different from the COM automation usage.

#187 - 10/31/2017 11:32 AM - Eugenie Lyzenko

Created task branch 3262b from trunk revision 11193.

#188 - 10/31/2017 10:07 PM - Eugenie Lyzenko

Task branch 3262b has been updated for review to revision 11194.

This fixes the DATE conversion issue for Java to COM and back. Investigation shows the COM DATE is a simple double value in format days.part_of_the_day. The base for OLE date is: 31 December 1899. And this is 25568 days since Java Date class base - 01 January 1970. The conversion implementation uses these constants to calculate Java and OLE dates respectively. Java Date is considered as long milliseconds value.

#189 - 10/31/2017 10:19 PM - Eugenie Lyzenko

Greg Shah wrote:

> > I'm not 100% sure however having deep suggestion the Java has one bug in Windows package. The long value can not be passed from Java to JNI without losses. Only 32-bit portion of long is handling correct for 64-bit long.
>
> I don't think this is a limitation. We pass 64-bit memory addresses to/from JNI for memptr support and also for library calls. Perhaps it is worth looking at those examples to see how it may be different from the COM automation usage.

You are right and my suggestion was wrong. The Java passes long correctly. The issue I saw was improper usage fwprintf(stdout, "format %ld", jlong) to display jlong value. The correct usage is: fwprintf(stdout, "format %I64d", jlong). In this case I see full range 64-bit integer.

#190 - 11/01/2017 09:56 PM - Eugenie Lyzenko

Task branch 3262b has been updated for review to revision 11195.

This is the fix for COM-HANDLE attribute getting issue with NotSerializableException exception. When reading result from client to server after requesting attribute value the comhandle assumes the value member exists. But for this case it is null and must be read by in.readObject() call before we can consider if it is externalizable or not.

Also the update fixes the bug with string representation of the COM-HANDLE variable. We must use long ID value here.

#191 - 11/02/2017 11:25 AM - Constantin Asofiei

Eugenie, I've looked at 3262b rev 11195 and I have these comments for the Java changes:

- NativeComObject - the id() and getComId() are different IDs and can't be the same. id() is used internally by FWD for mapping the HANDLE attribute, and getComId() is an internal ID used by only the COM object. So the NativeComObject.id() is incorrect. Why do you need them to be the same?
- ComObject c'tor - why do you need the check for NativeComObject? As it extends ComObject, the check is not needed

#192 - 11/02/2017 12:39 PM - Eugenie Lyzenko

Constantin Asofiei wrote:

> Eugenie, I've looked at 3262b rev 11195 and I have these comments for the Java changes:
> 2. ComObject c'tor - why do you need the check for NativeComObject? As it extends ComObject, the check is not needed

The ComObject class was not changed since branch creation. What do you mean? Please clarify.

#193 - 11/02/2017 12:46 PM - Eugenie Lyzenko

Constantin Asofiei wrote:

> Eugenie, I've looked at 3262b rev 11195 and I have these comments for the Java changes:
>
> - NativeComObject - the id() and getComId() are different IDs and can't be the same. id() is used internally by FWD for mapping the HANDLE attribute, and getComId() is an internal ID used by only the COM object. So the NativeComObject.id() is incorrect. Why do you need them to be the same?

The problem is the incorrect conversion of the NativeComObject to String to use with

```
message ComHahdleVar.
```

4GL statement. In this case the class returns internal ID instead of expected comID. The other way is to implement toString() or some part of it for NativeComObject

#194 - 11/02/2017 12:56 PM - Constantin Asofiei

Eugenie Lyzenko wrote:

> Constantin Asofiei wrote:
>
> > Eugenie, I've looked at 3262b rev 11195 and I have these comments for the Java changes:
> >
> > - NativeComObject - the id() and getComId() are different IDs and can't be the same. id() is used internally by FWD for mapping the HANDLE attribute, and getComId() is an internal ID used by only the COM object. So the NativeComObject.id() is incorrect. Why do you need them to be the same?
>
> The problem is the incorrect conversion of the NativeComObject to String to use with
> [...]
> 4GL statement. In this case the class returns internal ID instead of expected comID. The other way is to implement toString() or some part of it for NativeComObject

Can the com-handle variable be used with HANDLE function? I mean, re-construct a com-handle from its ID representation as a string?

What does 4GL show for message comHandleVar.? Also, does a com-handle var have a HANDLE attribute?

#195 - 11/02/2017 12:58 PM - Constantin Asofiei

Eugenie Lyzenko wrote:

> Constantin Asofiei wrote:
>
> > Eugenie, I've looked at 3262b rev 11195 and I have these comments for the Java changes:
> > 2. ComObject c'tor - why do you need the check for NativeComObject? As it extends ComObject, the check is not needed
>
> The ComObject class was not changed since branch creation. What do you mean? Please clarify.

Sorry, I mean this change in comhandle:

```
ca@xuxa:/working/gcd.files/gcd.bzr/branches/3262b$ bzr diff --revision 11193 src/com/goldencode/p2j/util/comha
ndle.java
=== modified file 'src/com/goldencode/p2j/util/comhandle.java' (properties changed: +x to -x)
--- src/com/goldencode/p2j/util/comhandle.java  2017-10-26 19:28:33 +0000
+++ src/com/goldencode/p2j/util/comhandle.java  2017-11-02 13:20:14 +0000
```

```
@@ -9,6 +9,7 @@
 ** 002 SIY 20101014 Changed signatures.
 ** 003 OM  20170523 Added new methods and completed implementation of existing ones.
 ** 004 CA  20171026 Changes for native COM automation support.
+** 005 EVL 20171101 Fix for COM object handle reading with client-server exchange.
 */

 /*
@@ -452,6 +453,10 @@
             this.value = that.value;
          }
       }
+      else if (value instanceof NativeComObject)
+      {
+         this.value = (NativeComObject) value;
+      }
```

#196 - 11/02/2017 01:54 PM - Eugenie Lyzenko

- File com_handle_prop_correct.jpg added

- File com_handle_prop_incorrect.jpg added

Constantin Asofiei wrote:

> Can the com-handle variable be used with HANDLE function?

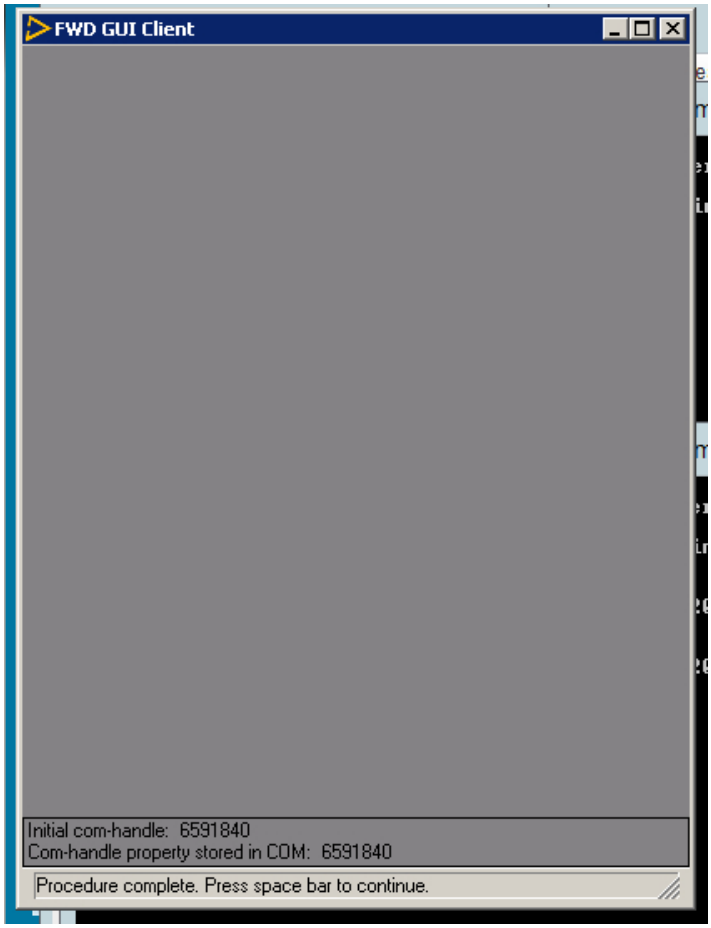
No, incompatible data type.

> I mean, re-construct a com-handle from its ID representation as a string?

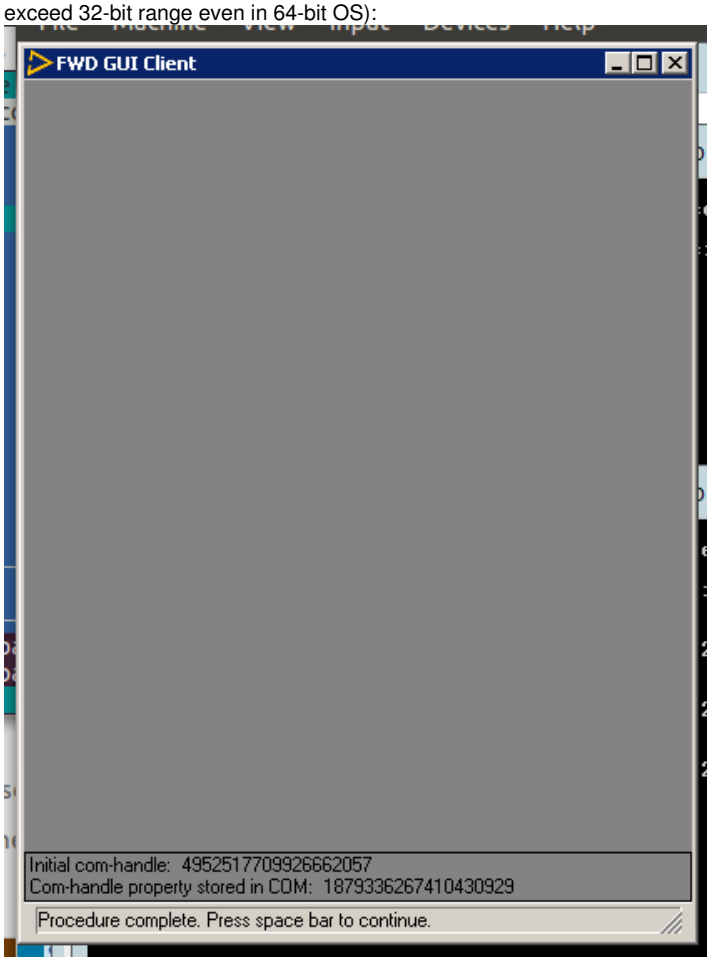
No, see above.

> What does 4GL show for message comHandleVar.?

This is set_get_com_handle_prop.p output.
The correct screen, note both numbers are the same:



Incorrect screen, both numbers are different but must be the same and both numbers are too big(it looks weird but COM-HANDLE ID does not exceed 32-bit range even in 64-bit OS):



Also, does a com-handle var have a HANDLE attribute?

No.

#197 - 11/02/2017 02:03 PM - Constantin Asofiei

Eugenie Lyzenko wrote:

> Incorrect screen, both numbers are different but must be the same and both numbers are too big(it looks weird but COM-HANDLE ID does not exceed 32-bit range even in 64-bit OS):
> ...
>
> > Also, does a com-handle var have a HANDLE attribute?
>
> No.

Thanks, then I need to make some changes to not register a comhandle resources the same way handle resources are registered. Your change related to NativeComObject.id() now makes sense.

#198 - 11/02/2017 02:29 PM - Eugenie Lyzenko

Constantin Asofiei wrote:

> Eugenie Lyzenko wrote:
>
> > Constantin Asofiei wrote:
> >
> > > Eugenie, I've looked at 3262b rev 11195 and I have these comments for the Java changes:
> > > 2. ComObject c'tor - why do you need the check for NativeComObject? As it extends ComObject, the check is not needed
> >
> > The ComObject class was not changed since branch creation. What do you mean? Please clarify.
>
> Sorry, I mean this change in comhandle:
> [...]

Yes, you are right, this is not necessary. I've made this change while debugging COM-HANDLE code to investigate what is wrong. But forgot to undo this change on cleanup. Thanks.

I'm going to commit changed comhandle source in a 10 min if no objections.

#199 - 11/02/2017 02:52 PM - Eugenie Lyzenko

The cleanup for comhandle has been uploaded as rev 11196.

#200 - 11/02/2017 04:37 PM - Constantin Asofiei

3262b rev 11197 contains the changes to make the comhandle id unique for FWD's COM object representation and not re-use the handle management ID.

#201 - 11/02/2017 08:57 PM - Eugenie Lyzenko

Constantin,

As you know we need to add support for arrays produced from 4GL extent option. My plan is following:

1. Add one more member and method to the ComParameter class to return true if the ComParameter object includes the array inside(value is and array of object instead single one). This will be enough I guess for native code to differentiate the processing. I just would prefer this call to be integer, not boolean. Because in native code I need to know the array size. So for example value 0 will mean this is the single object, value > 0 will mean the array with respective size. The return value from COM to Java can also be array of BaseDataType based class.

2. The point where this input array is constructing can be the ComObject.unmarshal(Object) method where you implemented instanceof BaseDataType check. We can simply ask if it is instanceof BaseDataType[] or call Object.getClass().isArray().

So this way we can use same calls for set/get property and method for both single objects and arrays.

Will you implement extent support? Or this is mine task? Let me know your suggestions, I need to have exact understanding of what is your part and what is mine in this task.

#202 - 11/02/2017 10:04 PM - Constantin Asofiei

Eugenie Lyzenko wrote:

> Constantin,
>
> As you know we need to add support for arrays produced from 4GL extent option. My plan is following:
>
> 1. Add one more member and method to the ComParameter class to return true if the ComParameter object includes the array inside(value is and array of object instead single one). This will be enough I guess for native code to differentiate the processing. I just would prefer this call to be integer, not boolean. Because in native code I need to know the array size. So for example value 0 will mean this is the single object, value > 0 will mean the array with respective size. The return value from COM to Java can also be array of BaseDataType based class.

The variable's extent can be 0 (for dynamic extent), 1 or anything else greater than 1. So is not enough to have single field with extent value, we need a flag marking this as an extent AND the var's extent.

Where it gets complicated is the OUTPUT/INPUT-OUTPUT parameters: ComOleHelper.call needs to change its return type from BaseDataType[] to something else... I'm OK with a Object[] return type, where each element can be either a single BDT instance or BDT[] array. Do you prefer something else? Same for getProperty...

> 2. The point where this input array is constructing can be the ComObject.unmarshal(Object) method where you implemented instanceof BaseDataType check. We can simply ask if it is instanceof BaseDataType[] or call Object.getClass().isArray().

Yes, this is where it will be, but that will not be the single change needed for extent.

> So this way we can use same calls for set/get property and method for both single objects and arrays.
>
> Will you implement extent support? Or this is mine task? Let me know your suggestions, I need to have exact understanding of what is your part and what is mine in this task.

Yes, I'll work on this.

#203 - 11/02/2017 11:57 PM - Eugenie Lyzenko

Constantin Asofiei wrote:

> Where it gets complicated is the OUTPUT/INPUT-OUTPUT parameters: ComOleHelper.call needs to change its return type from BaseDataType[] to something else... I'm OK with a Object[] return type, where each element can be either a single BDT instance or BDT[] array. Do you prefer something else?

What is the difference between usage BaseDataType[] and Object[]? I think BaseDataType[] is OK because all data types we are going to support are extending BaseDataType. Upon receiving data from native code Java can check if the object is an array or single instance. Getting data from COM the native code knows if this is array or single so this is not an issue for native code.

#204 - 11/03/2017 12:03 AM - Eugenie Lyzenko

More checks shows the additional issue with date/datetime/datetime-tz data type support. We loose precision passing data to COM and getting back. The method used to get the date value does not include time so this info is removing from value passed to COM. Working on the solution.

#205 - 11/03/2017 01:14 PM - Constantin Asofiei

Eugenie Lyzenko wrote:

> Constantin Asofiei wrote:
>
> > Where it gets complicated is the OUTPUT/INPUT-OUTPUT parameters: ComOleHelper.call needs to change its return type from BaseDataType[] to something else... I'm OK with a Object[] return type, where each element can be either a single BDT instance or BDT[] array. Do you prefer something else?
>
> What is the difference between usage BaseDataType[] and Object[]? I think BaseDataType[] is OK because all data types we are going to support are extending BaseDataType.

No, extent values are kept in FWD as Java arrays - i.e. an def var i as int extent 5 will be defined as new integer[5] in FWD. So, the return type can't handle a BDT[] unidimensional array. At most, it needs to be 2-dimensional, so you can have on index 1 a character variable and on index 2 a integer[5] extent variable.

> Upon receiving data from native code Java can check if the object is an array or single instance. Getting data from COM the native code knows if this is array or single so this is not an issue for native code.

I don't understand this one - Java doesn't let you keep an array as an element for a uni-dimensional array; you must use a 2-dimensional array for an element to be an array on its own.

#206 - 11/03/2017 05:38 PM - Constantin Asofiei

Greg, I need an advice. 4GL allows a construct like this at parse time:

```
def var y as int extent.
no-return-value fwdAppl:get2(output y).
```

I want to convert this to something like this, so that the Java reference is updated for an OUTPUT dynamic extent argument:

```
   public void execute()
   {
      integer[] y = UndoableFactory.integerExtent();

      externalProcedure(ComExtent.this, new Block((Body) () ->
      {
         fwdAppl.call("get2", ComParameter.output(y, (_y) -> y = (integer[]) _y));
      }));
   }
```

but Java doesn't compile as "y is not effectively final". I can promote this to instance field... but although this solves the external procedure case, there is also the internal procedure case which has the same problem.

Any other idea how I can solve this?

#207 - 11/03/2017 05:47 PM - Constantin Asofiei

Eugenie, 3262b rev 11198 contains the first pass at extent support from Java side. Some other questions:

- can a COM property be an extent?
- can a COM method return an extent value?

#208 - 11/03/2017 06:11 PM - Eugenie Lyzenko

Constantin Asofiei wrote:

> Eugenie, 3262b rev 11198 contains the first pass at extent support from Java side.

Are you planning to do rebase with the recent trunk today? I also have the changes to upload and then planning to do rebase if you have no plans to rebase earlier.

> Some other questions:
> 1. can a COM property be an extent?
> 2. can a COM method return an extent value?

For both points the answer is similar. From the native COM viewpoint there is no such data as extent. The 4GL testing shows it is possible to use extent based array as property get/set option and method parameter/return type. It is up to 4GL to convert extent into COM compatible data type. In IFwdTest.object I use VARIANT to store data coming from 4GL for methods and properties to store extent array data and it works fine. The attempt to use SAFEARRAY to keep array inside COM does not work.

So we will have to implement proper data transformation in our native module.

#209 - 11/03/2017 07:56 PM - Eugenie Lyzenko

Task branch 3262b has been updated for review to revision 11199.

This is the fixes for date/datetime/datetime-tz data types processing with COM object. Plus a bit more.
1. The class datetimetz had a bug within one of it's constructor (date, NumberType, NumberType) setting offset value as date. Fixed.
2. The native module now uses datetime as base class to build return value for all date related variables. This keeps time value to be properly stored.
3. The function converting COM DATE to Java milliseconds long now uses roudl() function to make correct calculation. So the date and time values are now processing in a right way.

If there is no objection I will rebase the 3262b branch with a recent trunk in a 5 min.

#210 - 11/03/2017 08:10 PM - Eugenie Lyzenko

Task branch 3262b has been rebased with trunk 11195, new revision is 11201.

#211 - 11/03/2017 09:20 PM - Eugenie Lyzenko

Constantin,

Please update class source files header for your comauto modifications revision 11200.

Greg, Constantin,

My near future implementation plan.
1. Finish with single object remaining data types processing. For now it is RAW and MEMPTR, need to make 4GL tests to use these data. Also need to check precision issues for DECIMAL data type.
2. Implement extent arrays support.
3. Test different variable access type.
4. Test and debug methods with multiple input parameters with different input modes.
5. Test FWD COM functionality with real COM objects other than IFwdTest.object
6. Misc test and debugging not included in items above.

The point 2 and 3 can be under construction in parallel(at least partially) because they can have crossing points.

#212 - 11/04/2017 10:09 AM - Greg Shah

> but Java doesn't compile as "y is not effectively final". I can promote this to instance field... but although this solves the external procedure case, there is also the internal procedure case which has the same problem.
>
> Any other idea how I can solve this?

I think the primary problem with the promotion for the internal procedure case is recursion, right?

The alternative is to wrap the instance as a member of some container object. The container object is never reassigned, but the contents can be modified. If you used a custom inner class for the container, then the members could be referenced directly (no getter/setter methods needed). This would require all references to be rewritten with the qualifier.

I fact, I was thinking about doing this in general for each top-level block. It would solve te "too many parameters" issue with lambdas and would also theoretically perform better (though I have not been able to measure any actual impact of this approach).

The primary down side is the extra qualifier text for each "local" variable reference. If only used for this case and not all local var references, then this dereferencing is not a problem.

#213 - 11/06/2017 03:07 PM - Eugenie Lyzenko

The bzr repo updated to revision 1688.

Added the RAW and MEMPTR data type tests. Conclusion - the 4GL does not support direct passing of the MEMPTR datatype to COM OLE objects. Yes, we can use pointer getting from GET-POINTER-VALUE(MEMPTR) and then pass INT64 value to the COM object(as integer). But direct usage comObj:property = memPtrVar is not possible.

Constantin,

So we need to remove memptr related code from ComObject.marshal()|unmarshal() that trying to handle memptr.

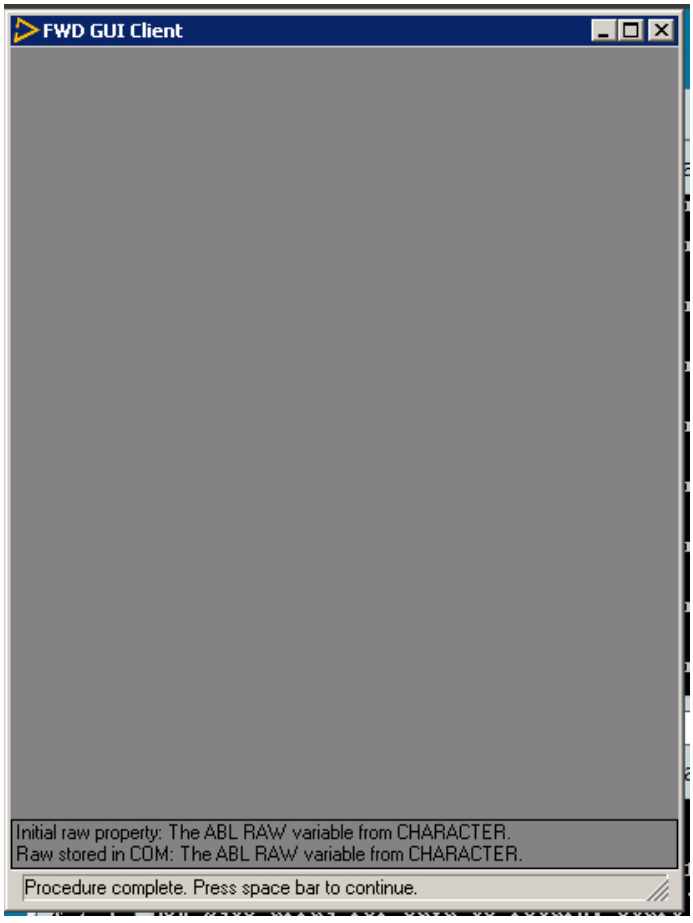
Soon I'm going to upload RAW data type handling in native code which is fully supported in 4GL<->COM. And I will remove all stubs in native code that planned to handle pure MEMPTR.

#214 - 11/06/2017 03:51 PM - Eugenie Lyzenko

- File raw_prop_set_get_test_20171106a.jpg added

Task branch 3262b has been updated for review to revision 11202.

The update has full support for RAW data type processing. It looks like:



Preparing to rebase with recent trunk. In a 10 min if nobody objects.

#215 - 11/06/2017 04:08 PM - Eugenie Lyzenko

Task branch 3262b has been rebased with trunk 11196, new revision is 11203.

#216 - 11/06/2017 06:11 PM - Eugenie Lyzenko

Some additional consideration for MEMPTR. Technically I think we can implement support for this data type in our FWD<->COM native code. Meaning passing MEMPTR object to COM we transform the data type to single VT_BYREF to store only pointer value inside COM and get then it back. Looks like there is a place in VARIANT to store this - (pvoid)VARIANT.byref. No memory data transfer is happening for MEMPTR.

Greg, what do you think? Do we need this feature to be tried to implement in our FWD native COM automation support?

#217 - 11/06/2017 06:25 PM - Greg Shah

> Greg, what do you think? Do we need this feature to be tried to implement in our FWD native COM automation support?

Does it work in the 4GL?

The 4GL documentation states this:

> ABL does not convert the BLOB, CLOB, MEMPTR, ROWID, or WIDGET-HANDLE data types to COM data types.

If it doesn't work, then we don't need to support it.

#218 - 11/06/2017 06:29 PM - Eugenie Lyzenko

Greg Shah wrote:

> > Greg, what do you think? Do we need this feature to be tried to implement in our FWD native COM automation support?
>
> Does it work in the 4GL?

No, it does not work in 4GL.

> The 4GL documentation states this:
>
> > ABL does not convert the BLOB, CLOB, MEMPTR, ROWID, or WIDGET-HANDLE data types to COM data types.

Yes, here the 4GL documentation is exactly matching the reality. MEMPTR is not supported.

> If it doesn't work, then we don't need to support it.

OK. I just thought about it as a bonus for our project.

#219 - 11/06/2017 06:50 PM - Greg Shah

I appreciate the idea. But since COM Automation is platform specific and hard coded to Windows, it is not a feature that we want to encourage use of. Like native library support, it is something we want to thoroughly and correctly implement for those cases that are needed, but we won't extend it with new features.

#220 - 11/06/2017 09:35 PM - Eugenie Lyzenko

Greg Shah wrote:

> I appreciate the idea. But since COM Automation is platform specific and hard coded to Windows, it is not a feature that we want to encourage use of. Like native library support, it is something we want to thoroughly and correctly implement for those cases that are needed, but we won't extend it with new features.

OK.

#221 - 11/07/2017 11:34 AM - Eugenie Lyzenko

Constantin,

I'm implementing native support for extent based array. And looks like I need you to make some changes on the Java side for me to continue. Now I'm passing the Objects array as parameter for property setter and looks like it works. But I need the array data returning from COM to Java to be accepted properly. Now I have class casting error because getProperty() wants the single BaseDataType object to be returned. But in case of extent array it is the BaseDataType[] or one of it's extend classes like integer[].

Can you change the getProperty() call from BaseDataType getProperty(String) to BaseDataType[] getProperty(String)?

#222 - 11/07/2017 11:36 AM - Eugenie Lyzenko

This is small update:

The single not array return value will be just a 1 item size array case.

#223 - 11/07/2017 11:50 AM - Constantin Asofiei

Eugenie Lyzenko wrote:

> This is small update:
>
> The single not array return value will be just a 1 item size array case.

Unfortunately is not that simple. Consider this code:

```
i = comVar:some-extent-prop.
```

which gets converted to:

```
i.assign(comVar.getProperty("some-extent-prop"))
```

The compile-time doesn't know what kind of data-type is some-extent-prop... and if getProperty returns BDT[] it will break all BDT.assign APIs.

I've been thinking about this, but I'm not sure how to solve it. We may need some hints to mark which property is extent ... and use something like getPropertyExtent for them. The same issue is with the 4GL COM method calls, if a method returns extent, its return type must be BDT[], not BDT, so we would need a callExtent. The downside here is, if 4GL makes the assignment data-type check at runtime, and shows an error in case of problems, this approach will not be able to pass compile-time.

It's a little tricky... same as OUTPUT or INPUT-OUTPUT extent arguments.

#224 - 11/07/2017 12:12 PM - Greg Shah

Can we hide this inside the runtime? For example, instead of this:

```
i.assign(comVar.getProperty("some-extent-prop"))
```

Can we do this:

```
comhandle.assignHelper(i, comVar.getProperty("some-extent-prop"))
```

The idea is to hide the logic in some code that is specific to COM Automation. I don't want to put the logic directly in each of the BDT classes. It doesn't have to be in the comhandle class, but that is an option.

#225 - 11/07/2017 12:17 PM - Eugenie Lyzenko

Constantin Asofiei wrote:

> Eugenie Lyzenko wrote:
>
> > This is small update:
> >
> > The single not array return value will be just a 1 item size array case.
>
> Unfortunately is not that simple. Consider this code:
> [...]
> which gets converted to:
> [...]
>
> The compile-time doesn't know what kind of data-type is some-extent-prop... and if getProperty returns BDT[] it will break all BDT.assign APIs.

The 4GL also does not perform compile time checking. It is completely up to programmer to know if the COM feature is compatible with 4GL code. In case of error the data type mismatch runtime error is generating.

> I've been thinking about this, but I'm not sure how to solve it. We may need some hints to mark which property is extent ... and use something like getPropertyExtent for them. The same issue is with the 4GL COM method calls, if a method returns extent, its return type must be BDT[], not BDT, so we would need a callExtent. The downside here is, if 4GL makes the assignment data-type check at runtime, and shows an error in case of problems, this approach will not be able to pass compile-time.
>
> It's a little tricky... same as OUTPUT or INPUT-OUTPUT extent arguments.

Can we use Object as return type? It can be checked for array or single class on the Java side and properly transformed to respective class(BaseDataType or BaseDataType[]).

#226 - 11/07/2017 02:11 PM - Constantin Asofiei

Greg Shah wrote:

> The idea is to hide the logic in some code that is specific to COM Automation. I don't want to put the logic directly in each of the BDT classes. It doesn't have to be in the comhandle class, but that is an option.

This solves the assignment, but I don't know how it will handle other code (like in a complex expression).

> Can we use Object as return type? It can be checked for array or single class on the Java side and properly transformed to respective class(BaseDataType or BaseDataType[]).

Yes, I think so, but only if we can use comhandle.assignHelper.

#227 - 11/07/2017 06:14 PM - Eugenie Lyzenko

Task branch 3262b has been updated for review to revision 11204.

This is extent support adding for native module. The first half of the implementation. For integer data type the both Java->COM and COM->Java transformations have been added. For rest data types - only Java->COM. The code looks almost the same for different data types but there are little but important differences(primitive type and pointer increasing step size, constants and so on), intermixing with common code. So for now it is not so easy to separate common code from data specific. Need to think more for possible better code structure.

Continue with COM->Java transformation for remaining data types.

#228 - 11/08/2017 12:06 PM - Eugenie Lyzenko

Task branch 3262b has been updated for review to revision 11205.

Finished adding COM->Java transformation for remaining data types base extent array handling. One small bug for INT64 datatype has been fixed.

Also I have changed /util/Text class replacing string specific (read|write)UTF() to more generic (read|write)Object(). The reason - some Unicode strings coming from native side can not be properly formatted with OutputObjectWriter.writeUTF() and causes application to crash with formatting exception. I think this is because Unicode string can have single 0x00 byte in a middle of(string terminator is double byte 0x0000). But if there are notes/objections for this change please let me know.

And one more change to native code. The strcmp() == 0 calls for data type checking were replaced with fast and simple integer values comparisons. The unique integer value is the first 4 bytes of the datatype string. It always has at least 4 bytes so can be considered as integer value. The date, datetime and datetime-tz are all processing the same way.

Continue with testing the multiple parameters for method calls with single return value.

#229 - 11/08/2017 01:39 PM - Eugenie Lyzenko

The bzr repo updated to revision 1691.

This adds the testcase to check the method accepting 3 input parameters and returning single value. Going to use as starting point for native code debugging. The IFwdTest object has also been updated.

#230 - 11/08/2017 06:43 PM - Eugenie Lyzenko

Task branch 3262b has been updated for review to revision 11206.

This is the COM method return value fix to reflect Java side change in method return signature. Now native module returns BaseDataType[][] 2d array. Tested for all supported data types with single return values, taking [0][0] slot. Continue debugging the multiple parameters passing task.

Going to rebase with recent trunk in a 10 min if nobody objects.

#231 - 11/08/2017 06:59 PM - Eugenie Lyzenko

Task branch 3262b has been rebased with trunk 11197, new revision is 11207.

#232 - 11/09/2017 07:46 AM - Constantin Asofiei

Eugenie, can you check if 4GL allows something like this, i = i + comVar:extentProp[1] ? Currently FWD can't convert code like this.

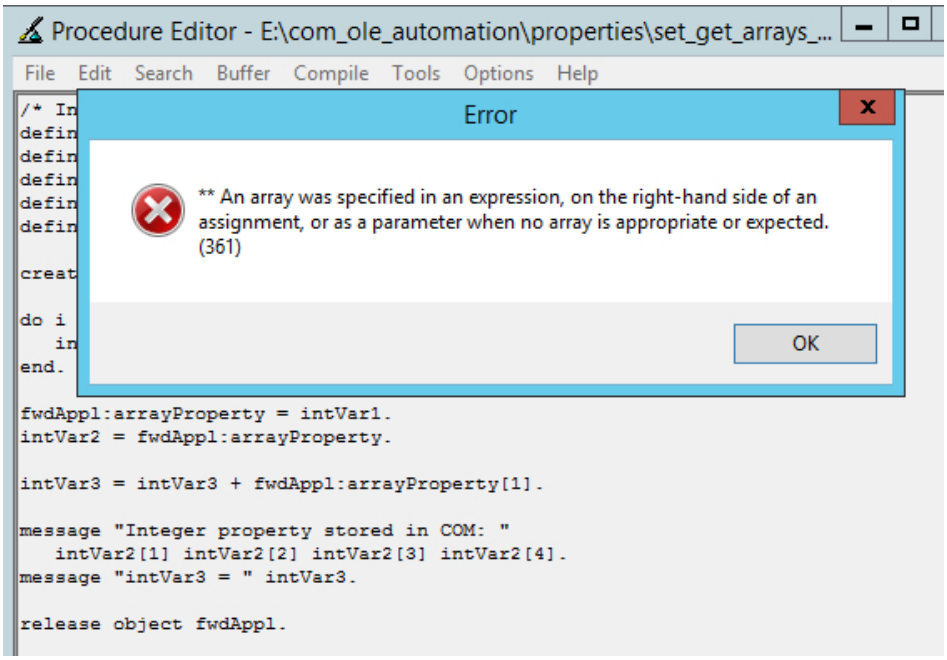
#233 - 11/09/2017 10:42 AM - Eugenie Lyzenko

- File com_wrong_array_prop.jpg added

Constantin Asofiei wrote:

> Eugenie, can you check if 4GL allows something like this, i = i + comVar:extentProp[1] ? Currently FWD can't convert code like this.

It is not possible and generates error on runtime. No compile time error. The screen:



I think it is because internally arrays are packed into VARIANT in COM and VARIANT is not an array itself, just a storage to keep everything supported. And trying to reference it as an array produces the error. The 4GL does not perform data compatibility for COM objects. It is advised for

4GL programmers to learn the COM object features before usage to avoid runtime issues.

#234 - 11/09/2017 06:14 PM - Eugenie Lyzenko

Task branch 3262b has been updated for review to revision 11208.

Small improvement to unify the common code in separate function. The old commented code still there for possible re-usage or debugging when array support is ready. The testcases repo has been updated to revision 1692. Adding the method taking 4 input parameter one of them is an array. The native code tested with multiple input method parameters and conclusion is: it works fine including the case when one of the parameter is an array.

I'm planning to rebase the 3262b branch with recent trunk in a 10 min if no objections.

#235 - 11/09/2017 06:35 PM - Eugenie Lyzenko

Task branch 3262b has been rebased with trunk 11198, new revision is 11209.

#236 - 11/09/2017 09:11 PM - Eugenie Lyzenko

The testcases repo has been updated to revision 1693, 1694.

This adds testcase to check different modes including INPUT-OUTPUT and OUTPUT.

#237 - 11/10/2017 05:07 PM - Eugenie Lyzenko

Constantin,

Some consideration about usage BY-POINTER and INPUT-OUTPUT or OUTPUT data modifiers. The usage INPUT-OUTPUT or OUTPUT forces the passing variable by pointer so it is not necessary to use BY-POINTER suffix mode when using INPUT-OUTPUT or OUTPUT. However missing INPUT-OUTPUT or OUTPUT and using only BY-POINTER mode does not cause the parameter value to be returned to 4GL and FWD. This is the important point because we need to properly prepare initial data before calling method, otherwise inconsistent data will be returned.

Currently the Java side reports ComParameter.isByPointer() is false when ComParameter.getMode() is 'O' or 'U'. I think it make sense to adjust Java side accordingly to return ComParameter.isByPointer() is true. Please let me know your considerations.

#238 - 11/10/2017 05:27 PM - Constantin Asofiei

Eugenie Lyzenko wrote:

> Constantin,
>
> Some consideration about usage BY-POINTER and INPUT-OUTPUT or OUTPUT data modifiers. The usage INPUT-OUTPUT or OUTPUT forces the passing variable by pointer ...

I think you are seeing this in a 4GL's point of view... in FWD's point-of-view, we can't pass 'by pointer' the BDT variable. Instead, the native side in FWD must use the received value for that argument, pass it to the COM call, and when it returns back to FWD, pass the current value for that argument.

If you want to change BY-POINTER to TRUE in FWD, for OUTPUT and INPUT-OUTPUT arguments, I'm OK with this, if it's easier to handle it on the native side. BTW, if BY-POINTER is used for INPUT, I assume the variable on that argument will be changed? If so, then we can assume INPUT-OUTPUT mode if BY-POINTER is present for INPUT, in FWD.

#239 - 11/10/2017 05:39 PM - Greg Shah

> BTW, if BY-POINTER is used for INPUT, I assume the variable on that argument will be changed?

Actually, I think this is NOT the case. The docs say the value is passed as a pointer but the 4GL does not copy any changes back to the 4GL variable.

The key here is that the 4GL has to create native data structures for the COM automation (just like we do). It does not directly provide access via pointer to the 4GL data structure.

The docs may be wrong, but this description is consistent with how they do things in the library calls too. There is an option there to pass by pointer and INPUT mode parameters are not copied back (even if the called API makes changes). I did test that and implemented it exactly that way.

#240 - 11/10/2017 06:00 PM - Eugenie Lyzenko

Constantin Asofiei wrote:

> Eugenie Lyzenko wrote:
>
> > Constantin,
> >
> > Some consideration about usage BY-POINTER and INPUT-OUTPUT or OUTPUT data modifiers. The usage INPUT-OUTPUT or OUTPUT forces the passing variable by pointer ...
>
> I think you are seeing this in a 4GL's point of view... in FWD's point-of-view, we can't pass 'by pointer' the BDT variable. Instead, the native side in FWD must use the received value for that argument, pass it to the COM call, and when it returns back to FWD, pass the current value for that argument.
>
> If you want to change BY-POINTER to TRUE in FWD, for OUTPUT and INPUT-OUTPUT arguments, I'm OK with this, if it's easier to handle it on the native side. BTW, if BY-POINTER is used for INPUT, I assume the variable on that argument will be changed? If so, then we can assume INPUT-OUTPUT mode if BY-POINTER is present for INPUT, in FWD.

No. It is not true. If BY-POINTER is used for INPUT the argument is **NOT** changing. Moreover for the FWD Java side the BY-POINTER option is useless because we can not pass the pointer to Java object. I have to use internal logic to prepare memory and provide pointers to COM object inside our native module. And this work fine (today I have added required functionality). I just need the Java side to provide sync info about parameter mode to use.

#241 - 11/10/2017 08:03 PM - Eugenie Lyzenko

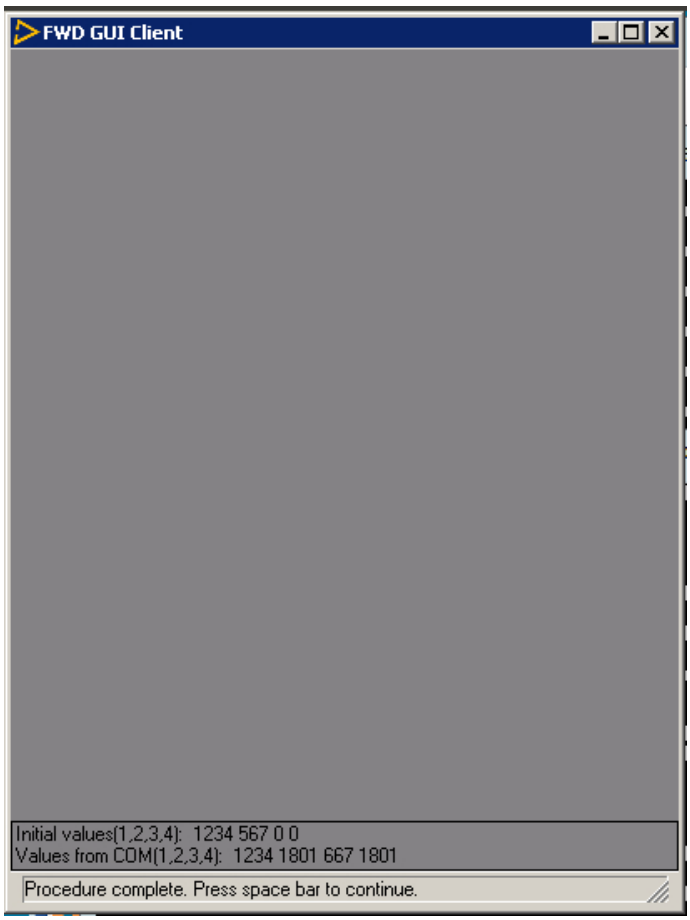
- File input_output_integer_method.jpg added

Task branch 3262b has been updated for review to revision 11210.

The update adds support for OUTPUT and INPUT-OUTPUT data access modes to native module. The idea is ho have static storage for all variables passed by-pointer to the COM object. The memory slots must be consistent until method that uses variables completes the work. We do not need to allocate/free dynamic memory because we actually need small amount of memory chunks and the current limit of 4K bytes practically unreachable by single method call - we need to have more than 1024 integer parameters with OUTPUT/INPUT-OUTPUT modes per single method call for example.

Also the update has a fix for Java side bug for merging parameters access for the cases when INPUT mode of one parameter is mixed with OUTPUT/INPUT-OUTPUT mode of others in single method call.

The sample working screen for methods/input_output_integer.p testcase:



#242 - 11/12/2017 11:10 AM - Eugenie Lyzenko

Task branch 3262b has been rebased with trunk 11199, new revision is 11211.

#243 - 11/13/2017 06:16 PM - Eugenie Lyzenko

Constantin, Greg,

I'm debugging different data input modes for COM object method call and found one issue to discuss.

Consider the following model when we have for example 4 method parameters and some of them are missing due to "" specification like this:

```
comObj:method1(opt1, opt2, opt3, )
```

or

```
comObj:method1(, opt2, , opt4)
```

In both cases FWD generates different numbers of ComParameter() instances to pack into aParam[] array, 3 and 2 for method respectively. In general it is not an big issue when optional parameter is the last one in a method signature. But if we have method1(, opt2, , opt4) 4GL code - we need to have more specific conversion processing because FWD code assume the first two are mandatory and last two - optional like in 4GL code: method1(opt1, opt2, ,). And this is wrong logic. The situation can become even worse if not all parameters are optional and we pass the value for optional parameter and miss for mandatory one.

I think we need to change method call processing on conversion level to have converted code like this:

```
call("setOpt4VarsData", ComParameter.empty(), ComParameter.input(intVar2), ComParameter.empty(), ComParameter.
input(intVar4));
```

Actually when COM method is calling the native code can set the data type as VT_ERROR and value as DISP_E_PARAMNOTFOUND = -2147352572 so we can also construct ComParameter.input(-2147352572, DataType.ERROR_CODE).

Either way the current Java side implementation of this does not work properly and can produce severe errors.

#244 - 11/13/2017 06:37 PM - Eugenie Lyzenko

Other findings for today:

1. The option BY-VARIANT-POINTER is never used in our large 4GL code.
2. The option BY-POINTER is used for COM-HANDLE variable. I have tested with IFwdTest object and it works fine in our current implementation.
3. The option AS datatype is never used in our large 4GL code. Except AS SHORT in DEFINE VARIABLE statement to declare procedure return value. Meaning it is also not a COM related case.

In a 10 min. I'm planning to update 3262b with native module change with adding support for VT_ERROR datatype processing in COM->Java data path section. Then I'm going to rebase the 3262b with the recent trunk if there are no objections.

#245 - 11/13/2017 07:18 PM - Eugenie Lyzenko

Task branch 3262b has been updated for review to revision 11212. Rebased with recent trunk, new revision is 11213.

Added the native module support for VT_ERROR data type returning from COM to Java. Checked for missed optional parameters for method calls. The documented restrictions have been found.

#246 - 11/13/2017 07:30 PM - Eugenie Lyzenko

The testcases repo has been updated to revision 1696.

Added tests to verify optional parameter mode for method call, testcase to use COM-HANDLE variable with BY-POINTER mode. Also the IFwdTest object updated to support new testcases.

#247 - 11/14/2017 07:44 PM - Eugenie Lyzenko

The investigation and debugging results for today.

1. The large application has object creation modes: CREATE "ComObject" or CREATE "ComObject" CONNECT NO-ERROR. Both of them are handling good with our IFwdTest and IFwdExe objects. The updated IFwdExe will be uploaded soon.

2. The mode CREATE "ComObject" CONNECT TO "some/file" is not required to be supported for now because there are no use cases in 4GL source files. However our native code handles is too for object creation. But after object ID returned the FWD server falls into exception:

```
Caused by: java.lang.ClassCastException: com.goldencode.p2j.comauto.NativeComObject cannot be cast to com.gold
encode.p2j.comauto.ServerComObject
        at com.goldencode.p2j.comauto.ComServer.create(ComServer.java:307)
...
```

The problematic code is ComServer:

```
...
         }

         ServerComObject comServer = (ServerComObject) aServer;  <-- Line 307
         ComObject automation = comServer.open(documentPath);
...
```

The class ServerComObject is not implemented properly for NativeComObject to be cast into it. We need t fix it. But as noted above the current 4GL code does not use this functionality for now.

3. There are some general consideration for real life COM object we are going to test with. First, MS does not recommend to use Office applications

as automation server due to security reasons and possible stability issues. This means nobody will guarantee MS Office application to work as expected with automation facility. Moreover I guess we will have to test coverted 4GL application with MS Office version that is exactly the same as used with original 4GL code. Just because Office functionality including COM automation support can vary from version to version(and not always version number upgrade means extending and/or improving support).

4. There is one point for COM object creation to be discussed. Imagine the application requests the object which is already running/used in the system. We have two options in this case, to return cached object handle of the already created object or to ask the OS to make the same thing. The both ways are now working. In second case the object ID will be different but the object name will be the same (if it was already loaded). Actually I guess we can omit request to OS for new ID and just increase the object usage counter. However this will fail if CREATE ... CONNECT statement internal is to make a copy of the current instance and must return com handle of the clone, not exactly original object.

Continue working.

#248 - 11/14/2017 08:14 PM - Eugenie Lyzenko

The testcases repo has been updated to revision 1697.

Added testcases to verify CREATE ..., CREATE ... CONNECT, CREATE ... CONNECT TO .... The native IFwdExe object is also updated to meet the 4GL test CONNECT mode. The CONNECT TO mode uses Office COM object.

#249 - 11/15/2017 09:10 AM - Greg Shah

> I think we need to change method call processing on conversion level to have converted code like this:
>
> > call("setOpt4VarsData", ComParameter.empty(), ComParameter.input(intVar2), ComParameter.empty(), ComParameter.input(intVar4));

I'm OK with this approach. We could also pass null instead.

Constantin: Any opinion?

#250 - 11/15/2017 04:58 PM - Constantin Asofiei

Greg Shah wrote:

> > I think we need to change method call processing on conversion level to have converted code like this:
> >
> > > call("setOpt4VarsData", ComParameter.empty(), ComParameter.input(intVar2), ComParameter.empty(), ComParameter.input(intVar4));
>
> I'm OK with this approach. We could also pass null instead.

Constantin: Any opinion?

I think is better a constant like ComParamter.EMPTY = null, it's easier to understand in the converted code. So it will be call("someMethod", ComParamter.EMPTY,

#251 - 11/15/2017 05:14 PM - Eugenie Lyzenko

Guys,

I'm debugging the 4GL COM automation indexed properties. There are a some issues while converting this feature. Consider the following code 4GL:

```
intVar1 = 1234321.
fwdAppl:int1DIndexedProperty(3) = intVar1.
intVar2 = fwdAppl:int1DIndexedProperty(3).
```

The fwdAppl is the COM object, int1DIndexedProperty(3) is an indexed property of the object. This converts to:

```
        intVar1.assign(1234321);
        isEqual(fwdAppl.call("int1DIndexedProperty", ComParameter.input(3)), intVar1);
        intVar2.assign(fwdAppl.call("int1DIndexedProperty", ComParameter.input(3)));
```

As you can see there are two issues here:
1. The indexed property assignment incorrectly converts to comparison operation considering = char as is equal operation.
2. The indexed property is considered as method call. It is pretty understandable because actually it is not possible to differentiate fwdAppl:int1DIndexedProperty(3) indexed property from method call fwdAppl:int1DIndexedProperty(3) with same name and one parameter.

Is it possible to differentiate method call from indexed property on the conversion level. I do not think so and I guess we need to make additional effort on the native level to find out if the given name is the name of the method or property. What do you think about conversion level?

For the first issue I'm not sure if it is possible to fix on conversion level or not. Because conversion is the place where it must be fixed I guess. Please share your knowledge here.

Also I think we need to check how often the indexed property is currently used in our large scale 4GL application. Is there an easy way to find it out? Except manually scan for source code tree to get the COM object and it's properties usage. Can I use the report analyzer HTML interface (report server) to easily look up all COM properties usage?

#252 - 11/15/2017 05:21 PM - Constantin Asofiei

> I'm debugging the 4GL COM automation indexed properties. There are a some issues while converting this feature. Consider the following code 4GL:

So they're using round brackets instead of square ones for indexed properties in COM? That's interesting...

> 2. The indexed property is considered as method call. It is pretty understandable because actually it is not possible to differentiate fwdAppl:int1DIndexedProperty(3) indexed property from method call fwdAppl:int1DIndexedProperty(3) with same name and one parameter.

We can infer that is a a COM property and not a method call, in cases when the construct appears as a LVALUE, as is not possible to have a method (of any kind) as a LVALUE.

We must do this at the parser level - maybe change the AST type from COM_METHOD to COM_PROPERTY, if it's parent is a KW_ASSIGN and this node is the first child, with round brackets? Otherwise, in the post-parse-fixups phase...

> Also I think we need to check how often the indexed property is currently used in our large scale 4GL application. Is there an easy way to find it out? Except manually scan for source code tree to get the COM object and it's properties usage. Can I use the report analyzer HTML interface (report server) to easily look up all COM properties usage?

I think we can check this if COM_METHOD is first child of a KW_ASSIGN. How is the AST looking?

#253 - 11/15/2017 05:22 PM - Constantin Asofiei

Eugenie, for the third case in your example: that one needs to be disambiguated by the FWD native side - i.e. check if there is a property with this name and use that instead of method; maybe consider that there is a single integer parameter, too.

#254 - 11/15/2017 06:07 PM - Eugenie Lyzenko

Constantin Asofiei wrote:

> ...
> I think we can check this if COM_METHOD is first child of a KW_ASSIGN. How is the AST looking?

```
<ast col="0" id="11334418694201" line="0" text="assignment" type="ASSIGNMENT">
  <ast col="33" id="11334418694202" line="9" text="=" type="EQUALS">
    <annotation datatype="java.lang.Long" key="support_level" value="16400"/>
    <annotation datatype="java.lang.Long" key="peerid" value="11338713661498"/>
    <ast col="8" id="11334418694205" line="9" text=":" type="COM_INVOCATION">
      <annotation datatype="java.lang.Long" key="peerid" value="11338713661499"/>
      <ast col="1" id="11334418694206" line="9" text="fwdAppl" type="VAR_COM_HANDLE">
        <annotation datatype="java.lang.Long" key="oldtype" value="2490"/>
        <annotation datatype="java.lang.Long" key="refid" value="11334418694147"/>
        <annotation datatype="java.lang.Long" key="support_level" value="514"/>
        <annotation datatype="java.lang.Long" key="peerid" value="11338713661500"/>
      </ast>
      <ast col="9" id="11334418694208" line="9" text="int1DIndexedProperty" type="COM_METHOD">
        <annotation datatype="java.lang.Long" key="oldtype" value="2490"/>
        <annotation datatype="java.lang.Boolean" key="com-call" value="true"/>
        <annotation datatype="java.lang.Long" key="peerid" value="11338713661501"/>
        <ast col="0" id="11334418694209" line="0" text="" type="COM_PARAMETER">
          <annotation datatype="java.lang.Long" key="peerid" value="11338713661502"/>
          <ast col="0" id="11334418694210" line="0" text="expression" type="EXPRESSION">
            <annotation datatype="java.lang.Long" key="support_level" value="16400"/>
            <annotation datatype="java.lang.Long" key="peerid" value="11338713661503"/>
            <ast col="30" id="11334418694211" line="9" text="3" type="NUM_LITERAL">
              <annotation datatype="java.lang.Boolean" key="is-literal" value="true"/>
              <annotation datatype="java.lang.Long" key="support_level" value="16400"/>
              <annotation datatype="java.lang.Boolean" key="use64bit" value="false"/>
              <annotation datatype="java.lang.Long" key="peerid" value="11338713661504"/>
            </ast>
          </ast>
        </ast>
      </ast>
    </ast>
    <ast col="0" id="11334418694212" line="0" text="expression" type="EXPRESSION">
      <annotation datatype="java.lang.Long" key="support_level" value="16400"/>
      <annotation datatype="java.lang.Long" key="peerid" value="11338713661505"/>
      <ast col="35" id="11334418694213" line="9" text="intVar1" type="VAR_INT">
        <annotation datatype="java.lang.Long" key="oldtype" value="2490"/>
        <annotation datatype="java.lang.Long" key="refid" value="11334418694161"/>
        <annotation datatype="java.lang.Long" key="support_level" value="16400"/>
        <annotation datatype="java.lang.Long" key="peerid" value="11338713661506"/>
      </ast>
    </ast>
  </ast>
</ast>
```

#255 - 11/15/2017 07:34 PM - Eugenie Lyzenko

Constantin Asofiei wrote:

> Eugenie, for the third case in your example: that one needs to be disambiguated by the FWD native side - i.e. check if there is a property with this name and use that instead of method; maybe consider that there is a single integer parameter, too.

There are 3 possible cases for object usage:
1. Property set
2. Property get
3. Method call

All of them uses different requests to the native COM object and in most cases are not mutually replaceable. Even property set and get. They must confirm special conventions that exactly declares what we want the COM object to do. In the case we want to use property but as COM module to call a method, even if we know what exact parameters number the property or method has, we do not exactly know if we want to call the get or set version of property. Especially when the incoming parameters can vary.

What I suggest is to separate at least the indexed property usage in fwdAppl:int1DIndexedProperty(3) = intVar1. from method call on conversion level. This 4GL construct can not be a method call, right? This way we narrow the issue to the problem of separation the method call from indexed property getter. Here I guess we can not do anything on conversion level and will have to do remaining separation on native module level.

BTW the property index does not have to be integer value, it can be arbitrary supported data type. Because we can not know how the COM object use the index with object logic, in may be not an simple array index but something other.

#256 - 11/15/2017 09:09 PM - Eugenie Lyzenko

The testcases repo has been updated to revision 1698.

Added testcases to verify indexed property feature and object release function.

Task branch 3262b has been updated for review to revision 11212.

The small fix to object releasing has been added to Java code. Without adding COM to local registry the object is always considering as invalid and moreover the release operation is not executed. Now the COM is properly releasing and guess we can consider this feature is working fine.

#257 - 11/15/2017 09:16 PM - Greg Shah

> ComParamter.EMPTY

I like this.

#258 - 11/15/2017 09:21 PM - Greg Shah

> What I suggest is to separate at least the indexed property usage in fwdAppl:int1DIndexedProperty(3) = intVar1. from method call on conversion level. This 4GL construct can not be a method call, right? This way we narrow the issue to the problem of separation the method call from indexed property getter. Here I guess we can not do anything on conversion level and will have to do remaining separation on native module level.

Correct. We can handle the assignment (setter access) to indexed property at conversion time, but we must handle the getter access to the indexed property at runtime in the native code.

#259 - 11/16/2017 11:07 AM - Constantin Asofiei

The parse, conversion and Java ComOleHelper API changes are in 3262b rev 11215.

Eugenie: looks like 4GL allows a construct like comVar:comProperty(1,2,3) = someVal., that's why there is another ComParameter[] array...

#260 - 11/16/2017 12:21 PM - Eugenie Lyzenko

Constantin Asofiei wrote:

> Eugenie: looks like 4GL allows a construct like comVar:comProperty(1,2,3) = someVal.,

Yes it is valid construct for indexed property setter.

#261 - 11/17/2017 12:04 AM - Eugenie Lyzenko

Task branch 3262b has been updated for review to revision 11216.

This adds support for indexed properties handling with set and get options. And some minor fixes for regressions for recent changes. Will be rebased with recent trunk in a 5 min if no objections.

#262 - 11/17/2017 12:24 AM - Eugenie Lyzenko

Rebased with recent trunk, new revision is 11217.

Some implementation details. Property set for indexed case is now mostly handling on the conversion level thanks to Constantin's changes. The rest of work for property setter is implemented in native module for possible indexed optional parameter.

For property getter I have added function to check if the method is property setter. Investigation shows calling convention for method and indexed property set is similar. Moreover the native COM backside not always reports correct function type. Fortunately this is not an issue. Sometimes the method is defined as not property, in this case we have to consider the call as pure method, in other cases it is possible to mix call as (METHOD | PROPERTY_GET).

So for now the remaining missing features on Java side are extent arrays as return values and optional parameters convention/handling.

#263 - 11/17/2017 06:52 PM - Eugenie Lyzenko

The required change to support optional parameters for method call has been prepared for native code. The idea is based on discussed form:

```
call("methodName", ComParameter.input(1), ComParameter.input(2), ComParameter.EMPTY, ComParameter.input(4))
```

Where ComParameter.EMPTY is equal null. So the native code receives null for jobject input variable in input array. The expected structure is:

```
inVars array:
[0] - ComParameter object
[1] - null
[2] - ComParameter object
...
```

If there will be another Java side implementation, please let me know. For example as I see the ComParameter.EMPTY is now going to be new Object() instance which is new approach for me. I need to know how I can detect if the Object() refers to empty object? Why do not we use simple null?

As soon as 3262b repo become unlocked I will upload the native support change.

#264 - 11/17/2017 08:02 PM - Eugenie Lyzenko

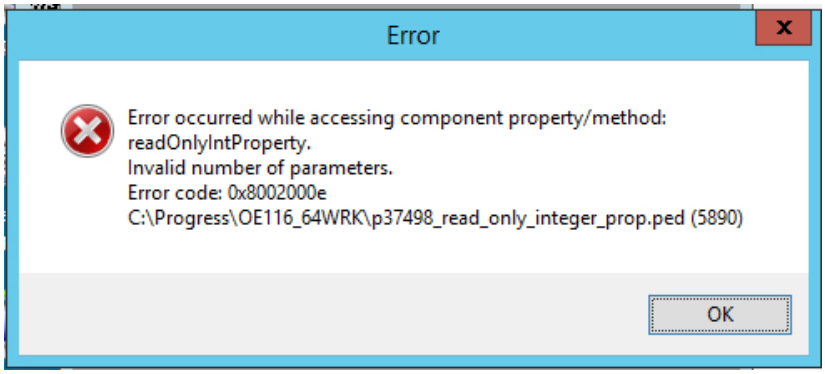
- File ro_property_set2.jpg added

- File ro_property_set1.jpg added

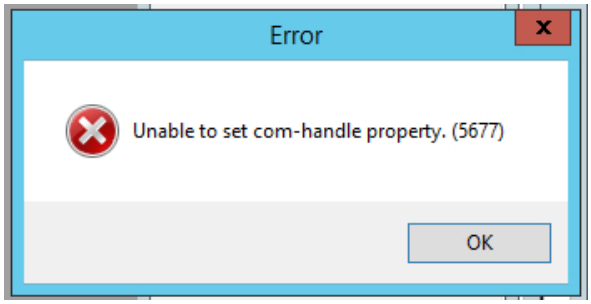
The testcases repo has been updated to revision 1699.

The read-only property test has been added. With respective test object modification. The attempt to set read-only property generates two sequential error messages.

First:



And second:



#265 - 11/18/2017 06:44 AM - Eugenie Lyzenko

Task branch 3262b has been updated for review to revision 11219.

The update adds native support for optional parameters based on null object incoming.

#266 - 11/18/2017 12:01 PM - Eugenie Lyzenko

Constantin,

The current Java side support for optional parameters is incomplete. The using comhandle.EMPTY(new Object) causes the exception to be thrown in ComObject, line 374:

```
...
                fwdDataType[0] = "date";
            }
            else
            {
Line 374->      throw new IllegalArgumentException(
                          value.getClass() + " data-type is not supported by native COM arguments!");
            }
         }
```

The value is Object. We need to add mode processing here to get input parameter that can be detected as "missed" by native code.

#267 - 11/18/2017 04:29 PM - Constantin Asofiei

Eugenie Lyzenko wrote:

> Constantin,
>
> The current Java side support for optional parameters is incomplete. The using comhandle.EMPTY(new Object) causes the exception to be thrown in ComObject, line 374:
> [...]
>
> The value is Object. We need to add mode processing here to get input parameter that can be detected as "missed" by native code.

Yes, I missed this part - how do you want it to work? Just set that value to null, and the native code will just check for null?

#268 - 11/18/2017 04:39 PM - Eugenie Lyzenko

Constantin Asofiei wrote:

> Yes, I missed this part - how do you want it to work? Just set that value to null, and the native code will just check for null?

Now the native code just check if the element of the parameter's array is null. This is considered by native code as parameter is missing. I think it is the simplest way to make it work.

The other way is to simulate data type as "ERROR" with error code value as -2147352572 (native MS SDK constant DISP_E_PARAMNOTFOUND).

I guess the null case is better(at least simpler).

#269 - 11/18/2017 05:47 PM - Constantin Asofiei

Eugenie, see 11220 - it should work, I'm sending null value for empty parameters.

#270 - 11/18/2017 06:20 PM - Eugenie Lyzenko

Constantin Asofiei wrote:

> Eugenie, see 11220 - it should work, I'm sending null value for empty parameters.

Yes, this works fine with current native module (from revision 11219) and testcases for missed parameters.

#271 - 11/20/2017 09:56 PM - Eugenie Lyzenko

The testcases repo has been updated to revision 1700.

The real MS Office testcases have been added. The current work is the debugging automation chain calls. The current implementation for constructs like:

```
comObject:prop1:prop2:(prop|method()).
```

We assume the variables inside this call are property so the real call become:

```
comObject1 = comObject:prop1.
comObject2 = comObject1:prop2.
comObject2:(prop|method()).
```

For now there are the issues under debugging for step 1-2 above. More investigation is required to make the chain calling work related to MS Office application. Planning to finish while extent support will be ready for Java level.

#272 - 11/21/2017 01:16 PM - Eugenie Lyzenko

Guys,

I have two notes at this time for Java side with real Excel application:

1. The NativeComObject must perform the same processing inside call() method like we have in getProperty(), if comOps returns comhandle - we need to create new NativeComObject instance calling createComHandle(). Otherwise we will not able to release the object obtained from COM method call(). And this is mandatory because every request to COM handle lock the object and resource remains alive in memory after program termination.

2. Consider the following 4GL code:

```
chWorksheet:NAME = "test".
```

This is converted to:

```
        chWorksheet.unwrapComObject().name(new character("test"));
```

I think this is incorrect conversion and we have to get something like:

```
        chWorksheet.setProperty("NAME", new character("test"));
```

Because we actually have the property set here.

For number 1 above I have made already changes to make it work. But not yet committed.

#273 - 11/21/2017 01:20 PM - Constantin Asofiei

Eugenie Lyzenko wrote:

> Guys,
>
> I have two notes at this time for Java side with real Excel application:
>
> 1. The NativeComObject must perform the same processing inside call() method like we have in getProperty(), if comOps returns comhandle - we need to create new NativeComObject instance calling createComHandle(). Otherwise we will not able to release the object obtained from COM method call(). And this is mandatory because every request to COM handle lock the object and resource remains alive in memory after program termination.

This should be converted to chainCall, if you have something like comVar:comMethod():comMethod(), i.e. comVar.chainCall("comMethod").call("comMethod").

> 2. Consider the following 4GL code:
> [...]
>
> This is converted to:
> [...]
>
> I think this is incorrect conversion and we have to get something like:
> [...]
>
> Because we actually have the property set here.

Correct, this should be converted as a setProperty - I guess we need to bypass the methods_attributes.rules (and maybe post-process these) so that parser doesn't emit FWD attributes or methods tokens, if we have COM chains.

#274 - 11/21/2017 01:54 PM - Eugenie Lyzenko

Constantin Asofiei wrote:

> Eugenie Lyzenko wrote:
>
> > Guys,

I have two notes at this time for Java side with real Excel application:

1. The NativeComObject must perform the same processing inside call() method like we have in getProperty(), if comOps returns comhandle - we need to create new NativeComObject instance calling createComHandle(). Otherwise we will not able to release the object obtained from COM method call(). And this is mandatory because every request to COM handle lock the object and resource remains alive in memory after program termination.

This should be converted to chainCall, if you have something like comVar:comMethod():comMethod(), i.e. comVar.chainCall("comMethod").call("comMethod").

Not necessary, consider the following 4gl:

```
chTemp = chExcel:Workbooks.
chTemp1 = chExcel:Workbooks().
```

converting to:

```
        chTemp.assign(chExcel.getProperty("Workbooks"));
        chTemp1.assign(chExcel.call("Workbooks"));
```

This is correct conversion and we have property getter and method call. Both of them return comhandle and in both cases we need to create NativeComObject, register it and release after program completes. So **every** single method call can return comhandle and we have to take this into account.

#275 - 11/21/2017 02:13 PM - Constantin Asofiei

Eugenie Lyzenko wrote:

This is correct conversion and we have property getter and method call. Both of them return comhandle and in both cases we need to create

> NativeComObject, register it and release after program completes. So **every** single method call can return comhandle and we have to take this into account.

Well, you need to mark it so that Java side can create a NativeComHandle or, as alternative, create it directly from the FWD native side (maybe by calling an API from ComOleHelper). These (getProperty and call) currently return a BaseDataType, so either way works.

#276 - 11/21/2017 02:55 PM - Eugenie Lyzenko

Constantin Asofiei wrote:

> Eugenie Lyzenko wrote:
>
> > 2. Consider the following 4GL code:
> > [...]
> >
> > This is converted to:
> > [...]
> >
> > I think this is incorrect conversion and we have to get something like:
> > [...]
> >
> > Because we actually have the property set here.
>
> Correct, this should be converted as a setProperty - I guess we need to bypass the methods_attributes.rules (and maybe post-process these) so that parser doesn't emit FWD attributes or methods tokens, if we have COM chains.

The interesting point is the other propery setters are convertimg fine. Something special in the line?

```
chWorksheet:NAME = "test".
```

The NAME is a reserved keyword? What makes NAME property to process differently?

#277 - 11/21/2017 02:57 PM - Constantin Asofiei

Eugenie Lyzenko wrote:

> The NAME is a reserved keyword? What makes NAME property to process differently?

Is a handle attribute. Try something else, like width-chars, sensitive, visible, etc - I assume you will get the same problem.

#278 - 11/21/2017 04:56 PM - Eugenie Lyzenko

Constantin Asofiei wrote:

> Eugenie Lyzenko wrote:
>
> > The NAME is a reserved keyword? What makes NAME property to process differently?
>
> Is a handle attribute. Try something else, like width-chars, sensitive, visible, etc - I assume you will get the same problem.

It's a kind of surprise but width-chars, sensitive, visible are converting properly as property set.

#279 - 11/21/2017 05:32 PM - Eugenie Lyzenko

I would suggest to change com_access.rules to:

```
            <!-- generic map lookup -->
            <rule>methodText.length() == 0
               <action>item = descriptors.get(ftype)</action>

               <rule>item != null and amref.type != prog.com_property
                  <action>hwrap = item.wrapper</action>
                  <action>methodText = item.getter</action>
```

The wrong name attribute is coming from above map in the file com_access.rules:

```
      <function name="load_descriptors">
         <return name="list" type="java.util.Map" />

         <rule>list = create("java.util.HashMap")</rule>
```

```
        <!-- Setter should be null for read-only attributes AND for methods -->
```

Honestly I do not understand why we need attributes support for COM-HANDLE. COM-HANDLE does not have any attributes, it has a properties instead. Do you agree with the change I'm suggesting?

#280 - 11/21/2017 05:37 PM - Constantin Asofiei

Eugenie Lyzenko wrote:

> Honestly I do not understand why we need attributes support for COM-HANDLE. COM-HANDLE does not have any attributes, it has a properties instead. Do you agree with the change I'm suggesting?

I think we can get rid of the descriptors usage completely - COM has no knowledge at compile time of how a property/method will behave, so everything will be dynamic.

#281 - 11/21/2017 06:07 PM - Eugenie Lyzenko

Constantin Asofiei wrote:

> Eugenie Lyzenko wrote:
>
> > Honestly I do not understand why we need attributes support for COM-HANDLE. COM-HANDLE does not have any attributes, it has a properties instead. Do you agree with the change I'm suggesting?
>
> I think we can get rid of the descriptors usage completely - COM has no knowledge at compile time of how a property/method will behave, so everything will be dynamic.

Agree. I'll make update for current state of the branch in a 5 min if nobody objects.

#282 - 11/21/2017 06:33 PM - Eugenie Lyzenko

Task branch 3262b has been updated for review to revision 11221.

This updates has changes to make chain calls working. Minor fix for COM-HANDLE conversion issue for properties included. Also made method call be compatible with handle coming from native side.

The native code has a lot of debugging output to be clear soon. The idea - there is one issue left to handle with chain calling. The intermediate COM handles created with lines like comObject:prop1:prop2:(prop|method()) is not explicitly released when main comObject releasing. But if not to do this we will have running application (Excel for example) after FWD program finishes. So I'm going to implement the approach of cleaning up all internal objects when main object is terminating.

The "internal" COM objects are now storing inside native registry with names produced from hexadecimal object ID to string. This is what the Java side will see requesting internal object name.

Also I have updated MS Office related testcases (rev 1701). The good news is the main test excel_create.p is now working. The constraint - we need to clean up all COM-OBJECT handles for Excel to finish after FWD application stopping. Continue working.

#283 - 11/22/2017 06:05 AM - Greg Shah

> The intermediate COM handles created with lines like comObject:prop1:prop2:(prop|method()) is not explicitly released when main comObject releasing. But if not to do this we will have running application (Excel for example) after FWD program finishes. So I'm going to implement the approach of cleaning up all internal objects when main object is terminating.

I don't think we can do this. First, I think the 4GL does not do this. Second, the code might save off the secondary com-handles and use them in other ways that our runtime won't know about.

From the 4GL docs (OpenEdge Development: Programming Interfaces):

```
Managing COM object resources

When working with COM objects, especially Automation objects, it can be very easy to instantiate
many object instances (for example, when searching many objects in a collection).This accumulation
of COM objects can impose a burden on system resources. To alleviate this burden, ABL provides
the RELEASE OBJECT statement. This statement releases the COM object associated with the
specified component handle, making it available for garbage collection if no other COM object has
a reference to it. In general, it is good practice to release any COM handles that you no longer
need for your application. It is only necessary to release COM objects that have been assigned to
a component-handle variable.

Releases and deletes

Note that you release COM objects, but delete widgets. A release is different from a delete because,
by convention, a COM object stays around until there are no other references to it. In the case of
an ActiveX control, the control-frame is a COM object that references the control (see Introduction
to External Program Interfaces on page 417). If you set a component handle variable to the control,
this is a second reference. However, no matter how many component handle variables you set to
the same control, this represents a single reference from ABL. (ABL takes care of this for you.) If
you release the control through a component handle while its control-frame is still instantiated, the control
remains instantiated because the control-frame COM object still references the control. In this case, you must
 delete the control-frame widget to finally free the ActiveX control. In a similar way, multiple COM object re
ferences can also keep an Automation object and its resources tied up longer than necessary.

Release strategy

In general, to maximize the reusability of COM object resources, always release a COM object
when you no longer need it. Because of references between COM objects, the status of references
to a particular COM object might not always be obvious (especially for Automation objects). By
```

```
releasing COM objects as soon as you know they are not needed in an application, you have the
best chance of ensuring that their resources are available for reuse as soon as possible.

Releases and component handles

When you release a COM object, this invalidates every component handle that references the
same object. Any further attempt to use an invalidated component handle results in an error
indicating that an invalid action was performed on an invalid COM-HANDLE value.
On the other hand, if you instantiate a different COM object after releasing the first one, a previously inval
idated component handle might point to the new COM object. The component handle might be reused because the ob
ject server might reuse memory left over from the released COM object.
In this case, the component handle might be seen as valid. However, the object that the component
handle references is different, so this might result in errors or successful method calls with results that ar
e invalid for your application.  In general, to avoid errors and confusion with invalidated COM handles, it is
 a good practice to set any COM-HANDLE variables to the Unknown value (?) once you have released it.

Note: ABL maintains no relationship between one component handle and another, such as when
you derive one component handle from a property value on another component handle. The two
component handles are completely independent of each other. This is different from the parent/child
relationships that some COM objects (especially Automation objects) might maintain among
themselves. In this case, invoking the right method on a "parent" COM object might well trigger
the release of many other "related" COM objects. For information on any such cascading object
relationships, see the documentation on a particular Automation Server or ActiveX control.
```

If I understand correctly, they are stating that the user is responsible for releasing all of the COM objects that are created during calls to methods or obtained via access to properties.

Please check to see if the 4GL auto-releases such instances. If not, then we should not do that either.

#284 - 11/22/2017 08:05 AM - Eugenie Lyzenko

Greg Shah wrote:

> If I understand correctly, they are stating that the user is responsible for releasing all of the COM objects that are created during calls to methods or obtained via access to properties.
>
> Please check to see if the 4GL auto-releases such instances. If not, then we should not do that either.

Yes, the 4GL does cleanup for internal objects created while property set/method call. But it can not do this without user's direct command for

releasing "main" object. That's why I think it is noted in doc to always explicitly release the object that is no longer in use. And it is critical point because without complete cleanup further usage of the same object well used before can become problematic.

I'm not going to implement auto-clean for the object the user can release but not did. Just for internal objects that are parts of the main object the user requested for release. I think this is the same as 4GL do.

#285 - 11/22/2017 02:00 PM - Eugenie Lyzenko

Task branch 3262b has been updated for review to revision 11222.

This is code partial cleanup and adding internal object release approach when main object releasing. Some debug code is still there. So now the sample completely free the COM subsystem, no remaining Excel application running after FWD code completes. Also made the COM calls locale neutral to avoid data distortion in COM objects with different locale than user default. And release code improved to take into account re-use the objects, if it was taken more than once - we need to release the same times.

Continue working on internal object release improvement to be more smart with parent-child relationship.

And I'm going to rebase the branch with recent trunk in a 10 min if no objections.

#286 - 11/22/2017 02:17 PM - Eugenie Lyzenko

Task branch 3262b has been rebased with trunk 11202, new revision is 11223.

#287 - 11/22/2017 03:50 PM - Greg Shah

Eugenie: Can you write the Java-side array changes that are needed? Constantin is busy with another task.

What else is left to do?

#288 - 11/22/2017 04:36 PM - Eugenie Lyzenko

Greg Shah wrote:

> Eugenie: Can you write the Java-side array changes that are needed? Constantin is busy with another task.

OK. I'll start to work on Java side for extent array support.

> What else is left to do?

I would say nothing but I'm not so sure. I guess we need to run our large scale application with COM related modules to use and we will see what is left because there is something we just do not see for now I think.

As minor task I'm thinking on object garbage collection improving, code clean up and check if we can do something to unify the native code. Also I need to re-view and discuss some constraint currently exists inside native code and actions for the cases the application is trying to go over limitation. The current limitations:
1. Maximum number of the COM objects can be active simultaneously - 1024
2. Maximum number of locally allocated strings - 1024 per single method call or property set/get
3. Maximum number of locally allocated arrays - 1024 per single method call or property set/get
4. Maximum total size of local storage for primitive types passed by pointer - 4096 bytes per single method call or property set/get
Every constraint is completely adjustable and does not related to any particular machine features, just my guess for reasonable approach. So we can change all if required.

For object handle garbage collector improvement. Current implementation that uploaded into branch and working good enough is based on the idea the every object can be divided into "root" that become active with CREATE ... objHandle statement and the objects that is getting from already existed objHandle via property get or method call or indirect chain calls that produce temporary handles. So we can mark every com handle as "root" or child with reference to the "root" handle. In this model when we release "root" object we can easily release all "child" objects as well. The case this model has disadvantage is when we release "child" object that also has children we need to release. In this case current implementation will release only specified child remaining children of this child be active until "root" object release. So if the "root" object is not releasing or releasing not in time the object leakage is possible. So thinking about more effective solution for objects tree release. Without heavy recursive code and too complex processing. We need something simple and fast or at least fast.

That's my plan.

#289 - 11/22/2017 09:42 PM - Eugenie Lyzenko

Task branch 3262b has been updated for review to revision 11224.

This is minor cleanup update. I have designed and testing the new approach for object garbage collector with deleting object tree for child object. But I have found interesting thing. The COM objects has no actual parent-child relationships. The same object can be obtained for re-use from different "parents" so this schema is just our artificial model to simplify object releasing. I need some time more to think about. While the current model works pretty good.

Meantime I'm shifting to extent array support on Java side level.

#290 - 11/23/2017 06:25 AM - Greg Shah

> The COM objects has no actual parent-child relationships. The same object can be obtained for re-use from different "parents" so this schema is just our artificial model to simplify object releasing.

My primary objective is to ensure that we implement the same behavior as the 4GL. No **less** object releasing and no **more** object releasing.

#291 - 11/23/2017 06:42 PM - Eugenie Lyzenko

Task branch 3262b has been updated for review to revision 11225.

This update adds extent array support for property get functions and method calls. Some changes in ../p2j/util/* located data classes to accept new approach. Also changed the object release. We do not need to keep usage count inside our module because the amount of the required release call can be taken from particular COM object.

The testcases repo updated to revision 1702.

The next step will be to run large scale project and test the used COM features to clean up and debug native module.

#292 - 11/23/2017 07:05 PM - Eugenie Lyzenko

Greg,

Looks like we also need to change gap report processing rules to reflect the added implementation, this is one more TODO thing, correct?

#293 - 11/23/2017 09:11 PM - Eugenie Lyzenko

Greg Shah wrote:

> The COM objects has no actual parent-child relationships. The same object can be obtained for re-use from different "parents" so this schema is just our artificial model to simplify object releasing.

My primary objective is to ensure that we implement the same behavior as the 4GL. No **less** object releasing and no **more** object releasing.

Yes, agree, this is what I'm currently investigating for.

#294 - 11/23/2017 10:45 PM - Eugenie Lyzenko

OK. Looks like the 4GL COM garbage collector approach has been clarified. The internal object taken from given COM object must be released while the main object releasing. We can not postpone releasing due to resource exhausting potential issue.

I have prepared approach to implement this. It allows to clean up related COM handles in single object list pass. Need a bit time more to optimize item construction to avoid significant delay while getting parents list in case of the large objects amount. I think several hours more required to complete implementation and testing.

When completed I guess we will have release candidate. Need to do some real testing with big code, fix possible bugs and then I think we can do regression testing.

#295 - 11/24/2017 06:45 AM - Greg Shah

> Looks like we also need to change gap report processing rules to reflect the added implementation, this is one more TODO thing, correct?

Yes, this will be done in the future, not now.

#296 - 11/24/2017 02:20 PM - Eugenie Lyzenko

Task branch 3262b has been updated for review to revision 11226, 11227.

This update implements improved object garbage collector to meet the 4GL behavior. In addition static memory consumption reduced, we use process heap to get memory on demand. Also code is cleaned from debug code and commented out lines. The total objects amount limitation expanded to have 4096 simultaneous COM handles. Guess it is big enough. Also the parent scan is now implements one pass approach so every object now requires only one list pass to analyze - we have linear increasing of the processing time depending on the object total amount.

The branch will be rebased soon with the recent trunk to re-test.

#297 - 11/24/2017 03:39 PM - Eugenie Lyzenko

Task branch 3262b has been rebased with trunk 11204, new revision is 11229.

#298 - 11/25/2017 05:06 PM - Eugenie Lyzenko

Task branch 3262b has been rebased with trunk 11205, new revision is 11230.

#299 - 11/25/2017 08:09 PM - Eugenie Lyzenko

Task branch 3262b has been updated for review to revision 11231.

Fix for regression caused by shifting from single object return to array for property get and method.

#300 - 11/26/2017 04:54 PM - Eugenie Lyzenko

Task branch 3262b has been updated for review to revision 11232.

Added fix for propath variation. Allows having ':' or ';' or both in propath string. I understand the strict requirement to have only ',' in propath string is not obvious because path in regular understanding has ';' or ':'. So with this fix we become much more tolerant to directory settings with propath related strings.

#301 - 11/27/2017 06:15 PM - Eugenie Lyzenko

Task branch 3262b has been updated for review to revision 11233, rebased with recent trunk, new version is 11234.

The update changes the search path fix to replace only Windows ';' with PROGRESS separator ','. This keeps the path entries like C:\some_dir for incorrect interpretation. So only Windows ';' separator is replacing and we have to use either ';' or neutral ',' separators in propath.

As to other branch changes. I have checked the big source application and found the cases that use COM prefixed with starting EXCEL.EXE as subprocess which is currently not supported in FWD GUI driver. If to disable starting EXE - excel compatible file is creating properly(when it is possible). But in this case the COM subsystem is not involved.

So I'm inclining to suggest the regression test execution now, the merge into trunk. I have the real COM application that uses EXCEL and it is working. I think we need to fix the issues that does not allow to use COM and then test the COM with 3262b integrated into trunk. Just because the COM part of the FWD is pretty independent from other runtime and I would like to finish this part of work with commit.

Is it OK plan? Or I will have to wait with 3262b testing and commit?

#302 - 11/28/2017 12:23 PM - Greg Shah

Code Review Task Branch 3262b Revision 11234

1. In assignments.rules, the newly added code is disabled because of the preceding false and ... part of the expression. Is this intentional?

2. The type-specific parameter processing code in ComObject seems like it could be factored differently. I think it probably could be done in separate methods and some of the code seems a bit repetitive. That being said, I don't think there is an urgent need to change this right now.

3. The EnvironmentOps change is not correct. Even if it worked, it would implement an asymmetric behavior (different behavior for : than for ;, which is confusing. But since ; is in fact a valid character to use in a Linux/UNIX file system (in the shell, you have to enclose such names in single quotes like 'some;file' but it does work). For this reason, can cannot just convert ; to ,.

In the directory.xml, I think we should require usage of , in searchpath, searchpath-fixed and probably some others. Please remove your changes to EnvironmentOps. I'm going to work this issue in #3396.

4. The changes in comauto_win.c are so extensive that it is difficult to consume all of the code. I have "skimmed" the code and my impression is that there is a great deal of code duplication. I think a different factoring could reduce the amount of code needed. At this point we don't have the time to rewrite it, so leave it for now.

5. assignments.rules, java_templates.tpl each need a history entry.

#303 - 11/28/2017 12:25 PM - Greg Shah

> found the cases that use COM prefixed with starting EXCEL.EXE as subprocess which is currently not supported in FWD GUI driver.

Can you explain what you mean by this? As far as I know, we can launch child processes in converted GUI code on Windows without an issue.

> So I'm inclining to suggest the regression test execution now, the merge into trunk.

Yes, I prefer this. First do the cleanups from the code review. Then put it into testing.

#304 - 11/28/2017 12:26 PM - Greg Shah

Constantin: Do you have any concerns about the changes in this branch?

#305 - 11/28/2017 01:56 PM - Eugenie Lyzenko

Greg Shah wrote:

> found the cases that use COM prefixed with starting EXCEL.EXE as subprocess which is currently not supported in FWD GUI driver.

Can you explain what you mean by this? As far as I know, we can launch child processes in converted GUI code on Windows without an issue.

The FWD executes ProcessDaemon.launch() for START EXCEL.EXE some_file.csv. This causes the ChildProcessFactory.getChildProcess() which in turns just return null for GUI driver(factory is an AbstractGuiDriver, line 502). Then we have immediately NullPointerException in the ProcessDaemon.launch(), line 240 stopping loading the EXCEL.EXE as child process.

According to the AbstractGuiDriver comments, looks like currently we support only ChUI driver to able child process to start.

#306 - 11/28/2017 01:58 PM - Eugenie Lyzenko

Greg Shah wrote:

> Code Review Task Branch 3262b Revision 11234
>
> 3. The EnvironmentOps change is not correct. Even if it worked, it would implement an asymmetric behavior (different behavior for : than for ;, which is confusing. But since ; is in fact a valid character to use in a Linux/UNIX file system (in the shell, you have to enclose such names in single quotes like 'some;file' but it does work). For this reason, can cannot just convert ; to ,.
>
> In the directory.xml, I think we should require usage of , in searchpath, searchpath-fixed and probably some others. Please remove your changes to EnvironmentOps. I'm going to work this issue in #3396.

OK. Be uploaded soon.

#307 - 11/28/2017 02:00 PM - Constantin Asofiei

Greg Shah wrote:

> Constantin: Do you have any concerns about the changes in this branch?

I've fixed some cases (removed the assignments.rules change), I'm looking again for missing history entries.

The issue I have are the extent changes: is not enough to fix assign and a few other cases. If you recall, we discussed that we need a way of wrapping the COM attribute/method calls at runtime so that it knows what it returns (either a BDT or a BDT[]), and not return always an array.

Eugenie: the large GUI app we are using has constructs like STRING(com-var:property) - this will not work with your changes as comhandle.getProperty returns BDT[]. And there are other cases, too, and we can't chase them all and duplicate the signature to accept BDT[] as parameter - there's just too many permutations.

At this time, I expect the large GUI app build to fail with your latest 3262b branch. So, please revert the extent changes and postpone these to another branch. To resume, the incomplete extent support is this:

1. in case of dynamic, uninitialized, extent com vars passed as OUTPUT or INPUT-OUTPUT parameters to COM method calls, we need to update the reference for the Java variable; this is not supported yet, a lambda expression is emitted, but it doesn't compile as the var reference is final. We need to extract these into an inner class and use that as a container which is final, so that field reference in this class can be updated.
2. the 'wrapping' mode for the COM property and method calls: depending on context, these will be wrapped into specific constructs. the comhandle APIs will be made to return Object, and real returned value will be either BDT or BDT[] array, and the wrapper API will just do all the required checks for this return value.

#308 - 11/28/2017 02:13 PM - Greg Shah

It seems we should just continue the work in this branch to finish the extent support, rather than reverting it all.

#309 - 11/28/2017 02:14 PM - Eugenie Lyzenko

Greg Shah wrote:

> Code Review Task Branch 3262b Revision 11234
>
> 1. In assignments.rules, the newly added code is disabled because of the preceding false and ... part of the expression. Is this intentional?
> ...
> 5. assignments.rules, java_templates.tpl each need a history entry.

Constantin, please clarify this. As author of these changes you have complete info about approach being used.

#310 - 11/28/2017 02:15 PM - Constantin Asofiei

Greg Shah wrote:

> It seems we should just continue the work in this branch to finish the extent support, rather than reverting it all.

I meant only some of the Java code, which is problematic. We can let the FWD native code (and related Java APIs) as is, and just log something if the native code has sent an Array (for a property or method return/parameter).

#311 - 11/28/2017 02:16 PM - Constantin Asofiei

Eugenie Lyzenko wrote:

> Greg Shah wrote:
>
> > Code Review Task Branch 3262b Revision 11234
> >
> > 1. In assignments.rules, the newly added code is disabled because of the preceding false and ... part of the expression. Is this intentional?
> > ...
> > 5. assignments.rules, java_templates.tpl each need a history entry.
>
> Constantin, please clarify this. As author of these changes you have complete info about approach being used.

These are fixed in 3262b 11236/11237.

#312 - 11/28/2017 02:24 PM - Greg Shah

> The FWD executes ProcessDaemon.launch() for START EXCEL.EXE some_file.csv. This causes the ChildProcessFactory.getChildProcess() which in turns just return null for GUI driver(factory is an AbstractGuiDriver, line 502). Then we have immediately NullPointerException in the ProcessDaemon.launch(), line 240 stopping loading the EXCEL.EXE as child process.
>
> According to the AbstractGuiDriver comments, looks like currently we support only ChUI driver to able child process to start.

I was not expecting this limitation. I guess I forgot this was deferred. This only affects the ProcessDaemon.launch(String[] cmdlist, boolean silent,

boolean wait) which is the interactive child process launch. I know we have tested the non-interactive version on Windows.

I suspect that the interactive child process launching may be different in GUI than in ChUI. I suspect that the suspend/resume and blocking behavior probably does not exist at all for GUI.

Eugenie: Would you please run some testcases to see how the interactive child process launching works under GUI? I think we will need you to fix this in this branch.

#313 - 11/28/2017 02:25 PM - Greg Shah

Constantin Asofiei wrote:

> Greg Shah wrote:
>
> > It seems we should just continue the work in this branch to finish the extent support, rather than reverting it all.
>
> I meant only some of the Java code, which is problematic. We can let the FWD native code (and related Java APIs) as is, and just log something if the native code has sent an Array (for a property or method return/parameter).

OK.

#314 - 11/28/2017 02:27 PM - Eugenie Lyzenko

Constantin Asofiei wrote:

> Eugenie: the large GUI app we are using has constructs like STRING - this will not work with your changes as comhandle.getProperty returns BDT[].

This can easily be handled with adding STRING(BDT[]) routing to STRING(BDT[0]).

#315 - 11/28/2017 02:29 PM - Constantin Asofiei

Eugenie Lyzenko wrote:

> Constantin Asofiei wrote:

> Eugenie: the large GUI app we are using has constructs like STRING - this will not work with your changes as comhandle.getProperty returns BDT[].

This can easily be handled with adding STRING(BDT[]) routing to STRING(BDT[0]).

Yes, but is not just this: there are hundreds of 4GL handle methods, builtin functions, parameter support and a lot of other stuff, where a COM property or method call can be used. We can't just overhaul everything.

#316 - 11/28/2017 02:42 PM - Eugenie Lyzenko

Task branch 3262b has been updated for review to revision 11238.

Undo the EnvironmentOps change that fixes the search path issue.

#317 - 11/28/2017 03:06 PM - Eugenie Lyzenko

Greg Shah wrote:

> Code Review Task Branch 3262b Revision 11234
> ...
> 4. The changes in comauto_win.c are so extensive that it is difficult to consume all of the code. I have "skimmed" the code and my impression is that there is a great deal of code duplication. I think a different factoring could reduce the amount of code needed. At this point we don't have the time to rewrite it, so leave it for now.

Yes, this is a problem I thought many times. In the cases I had any ability to make the code common I've made this. The two main "root causes" are the big using of Java classes inside native code and using the MS VARIANT struct which is actually the union. Different data types use different union fields to store itself. So unification here can cause the code to become even more complicated than now. But I agree we will have to return to this point later to improve.

#318 - 11/28/2017 03:17 PM - Eugenie Lyzenko

Constantin Asofiei wrote:

> Eugenie Lyzenko wrote:
>
> > Constantin Asofiei wrote:

> > > Eugenie: the large GUI app we are using has constructs like STRING - this will not work with your changes as comhandle.getProperty returns BDT[].
>
> > > This can easily be handled with adding STRING(BDT[]) routing to STRING(BDT[0]).
>
> > Yes, but is not just this: there are hundreds of 4GL handle methods, builtin functions, parameter support and a lot of other stuff, where a COM property or method call can be used. We can't just overhaul everything.
>
> I think only the real cases inside our big app can answer the question whether this code is working or not.

#319 - 11/28/2017 03:19 PM - Constantin Asofiei

Eugenie Lyzenko wrote:

> Constantin Asofiei wrote:
>
> > Eugenie Lyzenko wrote:
> >
> > > Constantin Asofiei wrote:
> > >
> > > > Eugenie: the large GUI app we are using has constructs like STRING - this will not work with your changes as comhandle.getProperty returns BDT[].
> > >
> > > This can easily be handled with adding STRING(BDT[]) routing to STRING(BDT[0]).
> >
> > Yes, but is not just this: there are hundreds of 4GL handle methods, builtin functions, parameter support and a lot of other stuff, where a COM property or method call can be used. We can't just overhaul everything.
>
> I think only the real cases inside our big app can answer the question whether this code is working or not.

On the other side, even if it would compile (and I expect not to compile, at least because of the STRING(comVar:comProperty) case), we can't be chasing this stuff for each new converted app. We need a generic solution, and the solution is the wrapper approach as described.

#320 - 11/28/2017 03:27 PM - Eugenie Lyzenko

Greg Shah wrote:

> Eugenie: Would you please run some testcases to see how the interactive child process launching works under GUI? I think we will need you to fix this in this branch.

Do you mean 4GL code? I can tell how it currently works in big app 4GL. The child process is starting as separate process, accepts the data and remains running even when the main 4GL application closed until user explicitly closes the application, EXCEL.EXE in our case.

#321 - 11/28/2017 03:32 PM - Eugenie Lyzenko

Planning to rebase the branch with the recent trunk in a 10 min. if nobody objects.

#322 - 11/28/2017 03:42 PM - Constantin Asofiei

Greg/Eugenie: I plan to make the Java code in 3262b generic enough so that we will be able to add the missing extent support described in #3262-307 easily.

#323 - 11/28/2017 03:44 PM - Greg Shah

> Eugenie: Would you please run some testcases to see how the interactive child process launching works under GUI? I think we will need you to fix this in this branch.

> Do you mean 4GL code?

Yes.

> I can tell how it currently works in big app 4GL. The child process is starting as separate process, accepts the data and remains running even when the main 4GL application closed until user explicitly closes the application, EXCEL.EXE in our case.

Please check the NO-WAIT, SILENT and NO-CONSOLE options. I suspect the NO-WAIT is probably implicitly ON in the Windows GUI case. If so, then there is no difference for that keyword being there or not.

#324 - 11/28/2017 03:45 PM - Greg Shah

Constantin Asofiei wrote:

> Greg/Eugenie: I plan to make the Java code in 3262b generic enough so that we will be able to add the missing extent support described in #3262-307 easily.

Good.

#325 - 11/28/2017 03:47 PM - Eugenie Lyzenko

Starting the rebase.

#326 - 11/28/2017 03:56 PM - Eugenie Lyzenko

Task branch 3262b has been rebased with trunk 11207, new revision is 11239.

#327 - 11/28/2017 05:59 PM - Eugenie Lyzenko

Greg Shah wrote:

> Please check the NO-WAIT, SILENT and NO-CONSOLE options. I suspect the NO-WAIT is probably implicitly ON in the Windows GUI case. If so, then there is no difference for that keyword being there or not.

No options - another console is opening and starts the child process, waits for child to complete including new opened console.
SILENT - starts the child process and continue to run without waiting for child to complete, no additional console.
NO-WAIT - starts the child process and continue to run without waiting for child to complete, no additional console.
NO-CONSOLE - does not start any child process in GUI.

So I do not see the difference between SILENT and NO-WAIT options in GUI mode.

#328 - 11/28/2017 06:51 PM - Eugenie Lyzenko

The testcases repo has been updated to revision 1705.

The child process starting testcases added for GUI mode with different options.

#329 - 11/28/2017 09:22 PM - Eugenie Lyzenko

Task branch 3262b has been updated for review to revision 11240.

The change adds initial step to turn the child process support on.

The testcases repo has been updated to revision 1705.

This update reflects the usage child process in real big app code. The testing shows for 11240 branch version the SILENT mode and default modes works as expected. The NO-WAIT mode works almost fine, no main process execution pause but console windows are visible and must be explicitly closed. The NO-CONSOLE mode is to be investigated additionally but in FWD it will start word/excel. And this is an issue because in 4GL this mode ignores command for application EXE. Continue working.

#330 - 11/29/2017 06:02 AM - Greg Shah

I will review the changes shortly.

> NO-CONSOLE - does not start any child process in GUI.

I don't understand what this does. The entire idea of OS-COMMAND is to start a child process. If it does nothing, then why is this there?

My expectation was that it would launch the command more directly in WIN32, without the intermediate shell (cmd.exe).

#331 - 11/29/2017 07:39 AM - Eugenie Lyzenko

Greg Shah wrote:

> I will review the changes shortly.
>
> > NO-CONSOLE - does not start any child process in GUI.
>
> I don't understand what this does. The entire idea of OS-COMMAND is to start a child process. If it does nothing, then why is this there?
>
> My expectation was that it would launch the command more directly in WIN32, without the intermediate shell (cmd.exe).

The option do nothing even for case:

```
os-command no-console "dir > c:\tmp\dir.log".
```

#332 - 11/29/2017 07:43 AM - Greg Shah

dir is an internal feature of the cmd.exe shell. It won't work without the shell. But launching a .exe program that doesn't require the shell should work. Otherwise why would the option even exist?

#333 - 11/29/2017 08:11 AM - Eugenie Lyzenko

Greg Shah wrote:

> dir is an internal feature of the cmd.exe shell. It won't work without the shell. But launching a .exe program that doesn't require the shell should work. Otherwise why would the option even exist?

Strange enough but os-command no-console excel.exe do nothing as well. May be it does not work at all in current 4GL.

#334 - 11/29/2017 08:24 AM - Greg Shah

Are you testing with OpenEdge 11.6?

#335 - 11/29/2017 08:59 AM - Greg Shah

Try running a program that has no GUI and which also can be made to run without STDIN/STDOUT/STDERR. In other words, it should be completely non-interactive and have no output.

#336 - 11/29/2017 10:27 AM - Hynek Cihlar

Greg Shah wrote:

> Try running a program that has no GUI and which also can be made to run without STDIN/STDOUT/STDERR. In other words, it should be completely non-interactive and have no output.

Eugenie, you may also try a system trace app to see whether the process is spawn. Possibly Sysinternals Procmon could help (https://docs.microsoft.com/en-us/sysinternals/downloads/procmon).

#337 - 11/29/2017 10:28 AM - Constantin Asofiei

Eugenie, I've tried this simple program, with pro, prowin and from procedure editor, and it opens the notepad.exe:

```
os-command no-console "notepad.exe".
```

BTW, there is no excel in our VM...

#338 - 11/29/2017 11:05 AM - Eugenie Lyzenko

Greg Shah wrote:

> Are you testing with OpenEdge 11.6?

11.6.3.

#339 - 11/29/2017 11:12 AM - Eugenie Lyzenko

Yes, notepad.exe, regedit.exe, also write.exe can be started with no-console option. However the behavior is slightly different. For example notepad.exe and regedit.exe cause the 4GL application to finish external app to continue after os-command, while in case of using write.exe the 4GL application does not wait for write.exe to finish and continue processing the lines after os-command no-console ....

#340 - 11/29/2017 11:15 AM - Eugenie Lyzenko

Constantin Asofiei wrote:

> Eugenie, I've tried this simple program, with pro, prowin and from procedure editor, and it opens the notepad.exe:
> [...]
>
> BTW, there is no excel in our VM...

I know. It is my own MS Office. But in order to test the COM and use it within big app we have to install it.

#341 - 11/29/2017 11:19 AM - Greg Shah

> For example notepad.exe and regedit.exe cause the 4GL application to finish external app to continue after os-command

NO-WAIT will make a difference in those cases.

> write.exe the 4GL application does not wait for write.exe to finish and continue processing the lines after os-command no-console ....

Most likely, write.exe does some kind of child process launch such that the original process exits, unblocking the 4GL processing.

#342 - 11/29/2017 02:58 PM - Eugenie Lyzenko

For Windows SILENT option I suggest to change the ProcessDaemon class to take into account the same reaction for SILENT and NO-WAIT:

```
...
      if (!silent && (client.inBatchMode() || (!wait && PlatformHelper.isUnderWindowsFamily())))
      {
         silent = true;
      }
...
```

Is this OK?

Observing the rule file for process launch I see we do not have any support for NO-CONSOLE option. Do we need to have such support at this time?

#343 - 11/29/2017 05:15 PM - Greg Shah

> Is this OK?

Yes.

> Observing the rule file for process launch I see we do not have any support for NO-CONSOLE option. Do we need to have such support at this time?

If it isn't too much work, I'd like to add it so that it is complete. The conversion side is pretty simple since this is just an optional boolean flag. I'm hoping the runtime is about removing the shell (cmd.exe) from the prepared command line, but not much else.

#344 - 11/29/2017 09:27 PM - Eugenie Lyzenko

Task branch 3262b has been updated for review to revision 11241.

This update fixes the handling of the NO-WAIT Windows processing and adding the NO-CONSOLE option conversion and runtime support.

Planning to rebase with recent trunk in a 10 min if no objection.

#345 - 11/29/2017 09:47 PM - Eugenie Lyzenko

Task branch 3262b has been rebased with recent trunk 11208, new version is 11242.

#346 - 11/29/2017 10:42 PM - Eugenie Lyzenko

Task branch 3262b has been updated for review to revision 11243.

This is minor javadoc fixes for my recent changes.

#347 - 11/30/2017 06:07 AM - Greg Shah

Code Review Task Branch 3262b Revision 11243

1. Please make the 4th parameter to launch() optional. This means that you would add a ProcessOps.launch(String[] cmdlist, boolean silent, boolean wait) and change process_launch.rules to **not** emit the parameter if no-console has not been explicitly specified.

2. In ProcessDaemon, is import com.goldencode.p2j.ui.client.*; needed? We have designed it to have a pluggable factory that provides the UI features without a direct dependency. That is why we do not have any imports of UI packages.

3. Can you use no-console in Linux?

#348 - 11/30/2017 08:47 AM - Eugenie Lyzenko

Greg Shah wrote:

> Code Review Task Branch 3262b Revision 11243
>
> 1. Please make the 4th parameter to launch() optional. This means that you would add a ProcessOps.launch(String[] cmdlist, boolean silent, boolean wait) and change process_launch.rules to **not** emit the parameter if no-console has not been explicitly specified.

OK.

> 2. In ProcessDaemon, is import com.goldencode.p2j.ui.client.*; needed? We have designed it to have a pluggable factory that provides the UI features without a direct dependency. That is why we do not have any imports of UI packages.

I used it in some experiments. Removed.

> 3. Can you use no-console in Linux?

I can "use" (no error at compile time) it in Linux ChUI but looks like no effect, I do not see any difference between having this option and missing. However it has effect only if GUI exists in system.

#349 - 11/30/2017 09:28 AM - Eugenie Lyzenko

Task branch 3262b has been updated for review to revision 11244.

The notes resolution changes. Optional NO-CONSOLE conversion and respective runtime changes.

#350 - 11/30/2017 11:00 AM - Greg Shah

Code Review Task Branch 3262b Revision 11244

I've checked in some changes as 11245. I'm good with that version. If child process launching works, then focus on the large app testing.

#351 - 11/30/2017 11:53 AM - Eugenie Lyzenko

Greg Shah wrote:

> Code Review Task Branch 3262b Revision 11244
>
> I've checked in some changes as 11245. I'm good with that version. If child process launching works, then focus on the large app testing.

Yes, the child process launching works as expected.

#352 - 11/30/2017 12:57 PM - Eugenie Lyzenko

Yes, we have signature match issue. Like this when FWD expects comhandle.call() returning BDT while our comhandle.call() returns BDT[].

Constantin,

Is this what you have told about?

```
Caused by: java.lang.NoSuchMethodError: com.goldencode.p2j.util.comhandle.call(Ljava/lang/String;[Ljava/lang/O
bject;)Lcom/goldencode/p2j/util/BaseDataType;
        at app.lambda$mainLoop$4(DashboardBook.java:146)
        at com.goldencode.p2j.util.Block.body(Block.java:604)
        at com.goldencode.p2j.util.BlockManager.processBody(BlockManager.java:6985)
        at com.goldencode.p2j.util.BlockManager.topLevelBlock(BlockManager.java:6776)
        at com.goldencode.p2j.util.BlockManager.internalProcedure(BlockManager.java:369)
        at com.goldencode.p2j.util.BlockManager.internalProcedure(BlockManager.java:355)
        at app.DashboardBook.mainLoop(DashboardBook.java:143)
        at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
        at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)
```

#353 - 11/30/2017 01:01 PM - Constantin Asofiei

Yes. I didn't start yet to fix the branch, I'll try tonight or tomorrow the latest.

#354 - 11/30/2017 04:15 PM - Constantin Asofiei

Eugenie, please see 3262b rev 11246. Note that you will need to have some changes in the FWD native COM part, because the signature for ComOleHelper.getProperty has changed - now it returns BaseDataType. We will postpone the full COM extent support to another branch. Avoid local tests which use extent for now.

As a side note, we will need for ComOleHelper.getProperty to return Object, as we will not be able to distinguish if native side has returned a value of extent 1 or a value of non-extent - so we will need native side to return either BDT or BDT[]. Same for COM method... the return type can't be distinguished; the parameters I think are OK because the runtime knows what it expects.

#355 - 11/30/2017 04:50 PM - Eugenie Lyzenko

Constantin Asofiei wrote:

> Eugenie, please see 3262b rev 11246. Note that you will need to have some changes in the FWD native COM part, because the signature for ComOleHelper.getProperty has changed - now it returns BaseDataType. We will postpone the full COM extent support to another branch. Avoid local tests which use extent for now.
>
> As a side note, we will need for ComOleHelper.getProperty to return Object, as we will not be able to distinguish if native side has returned a value of extent 1 or a value of non-extent - so we will need native side to return either BDT or BDT[]. Same for COM method... the return type can't be distinguished; the parameters I think are OK because the runtime knows what it expects.

Another word as I can see you removed (undone) all recent changes prepared to handle extent array support, right?

Why do not we use Object as wrapper for BDT[]? My plan is to leave returning BDT[] up to comhandle class level. Then we cast BDT[] to Object and return it from comhandle. Then the classes receiving Object will check if this is BDT[] and perform respective transformations. Is it a bad idea?

So for now you offer to have nothing extent array support. Does our big app not need this at all?

#356 - 11/30/2017 04:55 PM - Constantin Asofiei

Eugenie Lyzenko wrote:

> Another word as I can see you removed (undone) all recent changes prepared to handle extent array support, right?

Correct.

> Why do not we use Object as wrapper for BDT[]? My plan is to leave returning BDT[] up to comhandle class level. Then we cast BDT[] to Object and return it from comhandle. Then the classes receiving Object will check if this is BDT[] and perform respective transformations. Is it a bad idea?

The main problem here is how we distinguish between 1-length array and a simple value - that's why this needs to be disambiguated at the native side, so it will either return a simple BDT or a BDT[], for i.e. getProperty. That's why I removed the signature change for getProperty. Plus, the FWD conversion will need to emit some kind of wrappers which, depending on the surrounding context where the i.e. COM property is used, it will check if the property's value is compatible with that usage.

> So for now you offer to have nothing extent array support.

Yes, because the correct changes are more complex than my initial view.

> Does our big app not need this at all?

From my search, I couldn't find extent-like usage of COM.

#357 - 11/30/2017 05:14 PM - Eugenie Lyzenko

Guys,

More complex approach can be to introduce new class for results coming from native side to Java, say ComReturn with respective array or single object inside and proper methods to extract the real data on the Java side. This way getProperty() will return ComReturn and COM method call() will return ComReturn[].

What do you think?

#358 - 11/30/2017 07:14 PM - Eugenie Lyzenko

Constantin Asofiei wrote:

> > So for now you offer to have nothing extent array support.
>
> Yes, because the correct changes are more complex than my initial view.

Greg,

Is it OK to leave this feature postponed? I will have to change the native code, retest everything so please let me know the nearest plans in this area.

#359 - 11/30/2017 09:50 PM - Eugenie Lyzenko

Task branch 3262b has been updated for review to revision 11247.

OK. This update returns native code to the state of missing extent based array support for getProperty() call.

#360 - 11/30/2017 10:11 PM - Greg Shah

> Is it OK to leave this feature postponed?

Yes, you can match the support level of the Java code.

#361 - 11/30/2017 10:45 PM - Eugenie Lyzenko

During testing COM feature with big application found new issues:
1. The file resources to be taken from application jar is not properly handled, we have null instead as filename.
2. Passing character parameter as null to the COM method causes FWD crash likely in native method. Need to add such protection into native module.

#362 - 12/01/2017 10:47 AM - Eugenie Lyzenko

Task branch 3262b has been updated for review to revision 11248.

Adding protection for null characters passed as method parameter to avoid JVM crash in native code.

#363 - 12/16/2017 09:02 AM - Eugenie Lyzenko

Task branch 3262b has been rebased with recent trunk 11210, new version is 11250.

#364 - 12/20/2017 09:45 AM - Greg Shah

How safe is the 3262b branch at this point? Is it suitable for merge to trunk or is there known work needed?

I know that there is still the extent support to complete, but that is being deferred. What other work is needed before this task is done for the purposes of the current project? Any additional work needed for the current project could be done in a 3262c branch, if 3262b is safe enough to merge to trunk.

#365 - 12/20/2017 10:09 AM - Eugenie Lyzenko

Greg Shah wrote:

> How safe is the 3262b branch at this point? Is it suitable for merge to trunk or is there known work needed?

My testing experience in this area at the time the work was suspended:
1. There is an issues with getting the file resources (Excel files templates) from application *.jar files.
2. Without this issue fix the COM usage in big app become problematic.

3. If the COM calls conditions from point 2 are resolved (they are not related to COM implementation itself).

> I know that there is still the extent support to complete, but that is being deferred. What other work is needed before this task is done for the purposes of the current project? Any additional work needed for the current project could be done in a 3262c branch, if 3262b is safe enough to merge to trunk.

I think we can test and merge 3262b into trunk and defer other work to 3262c. Currently the COM usage fist step is to attach to *.xls file to be loaded from *.jar(or other possible locations like file system). Failing in this step does not allow to use COM functionality (and event to make next step ahead in debugging/testing).

#366 - 12/20/2017 10:26 AM - Greg Shah

> There is an issues with getting the file resources (Excel files templates) from application *.jar files.

Are these files that are checked in to source code control for the application? If so, then you are correct they should be in the jar file.

Is this really 2 issues?

1. There is probably a build issue. We don't jar these files as part of the build.
2. A FWD functional issue where we only try to open image files from the jar, but not other files.

> I think we can test and merge 3262b into trunk and defer other work to 3262c.

What testing is needed?

#367 - 12/20/2017 10:33 AM - Eugenie Lyzenko

Greg Shah wrote:

> > I think we can test and merge 3262b into trunk and defer other work to 3262c.

> What testing is needed?

The rules were changed, so conversion test is required I guess. Also safe harness run is desirable to make sure we will have no regression from COM related changes. This mean runtime ChUI regression testing. This is what I meant telling about tests to run.

#368 - 12/20/2017 10:39 AM - Greg Shah

> The rules were changed, so conversion test is required I guess. Also safe harness run is desirable to make sure we will have no regression from COM related changes. This mean runtime ChUI regression testing.

Please go ahead with ChUI regression testing.

#369 - 12/20/2017 10:41 AM - Eugenie Lyzenko

Greg Shah wrote:

> > The rules were changed, so conversion test is required I guess. Also safe harness run is desirable to make sure we will have no regression from COM related changes. This mean runtime ChUI regression testing.
>
> Please go ahead with ChUI regression testing.

OK. I'll rebase with current trunk and start testing.

#370 - 12/20/2017 10:53 AM - Eugenie Lyzenko

Greg Shah wrote:

> > There is an issues with getting the file resources (Excel files templates) from application *.jar files.
>
> Are these files that are checked in to source code control for the application? If so, then you are correct they should be in the jar file.

> Is this really 2 issues?
>
> 1. There is probably a build issue. We don't jar these files as part of the build.

Certainly we have no required *.xls files in our jar.

> 2. A FWD functional issue where we only try to open image files from the jar, but not other files.

I do not know for sure if we have working functionality if we had required files in jar. I have not investigated this deeply yet.

#371 - 12/20/2017 11:19 AM - Eugenie Lyzenko

Task branch 3262b has been rebased with recent trunk 11211, new version is 11251.

#372 - 12/20/2017 04:44 PM - Eugenie Lyzenko

Conversion testing has been passed, code trees are identical. Continue with runtime testing.

#373 - 12/20/2017 08:07 PM - Eugenie Lyzenko

The main cycle completed. No regression. Running CTRL-C part.

Can I merge into trunk if no regressions for CTRL-C will be found?

#374 - 12/20/2017 09:39 PM - Eugenie Lyzenko

Testing completed. No regressions have been found. The results: 3262b_11251_a5eeab0_20171220_evl.zip.

So the branch is ready to be merged into the trunk. Let me know please when I can do merging.

#375 - 12/21/2017 04:56 AM - Greg Shah

Please merge 3262b to trunk.

#376 - 12/21/2017 05:57 AM - Eugenie Lyzenko

Greg Shah wrote:

> Please merge 3262b to trunk.

OK. Starting.

#377 - 12/21/2017 06:03 AM - Eugenie Lyzenko

Branch 3262b was merged to trunk as revno 11212 then it was archived.

#378 - 12/22/2017 05:22 PM - Constantin Asofiei

There is an issue with how we convert LOAD-CONTROLS. We've decide to treat every COM access as a generic call (without mapping to dedicated FWD Java APIs), so no direct Java calls will be used.

We need to retest the existing implementations in FWD of COM objects (i.e. PSTimer) to see if these will work properly. We already use ComMethod annotation to identify the Java method, so it should be OK.

3394a 11308 fixes this issue.

#379 - 12/22/2017 06:35 PM - Eugenie Lyzenko

The revision 11308 gives the same converted code for COM sample testcases I have used in COM native implementation. And because we have no runtime changes the 11308 revision is safe from COM OLE automation subsystem perspective.

#380 - 03/08/2018 01:12 PM - Greg Shah

Is it fair to say that the only remaining work for COM automation support is:

- CONNECT TO in CREATE OBJECT
- array property get/set
- method parameters and return values that are arrays

Did I misstate anything?

Am I missing anything?

#381 - 03/08/2018 02:21 PM - Eugenie Lyzenko

Greg Shah wrote:

> Is it fair to say that the only remaining work for COM automation support is:
>
> - CONNECT TO in CREATE OBJECT
> - array property get/set
> - method parameters and return values that are arrays
>
> Did I misstate anything?
>
> Am I missing anything?

All states are correct as far as I understand.

#382 - 03/09/2018 11:24 AM - Greg Shah

- Related to Feature #3505: implement CONNECT TO support for CREATE OBJECT added

#383 - 03/09/2018 11:27 AM - Greg Shah

- Related to Feature #3506: implement array support for COM automation methods and properties added

#384 - 03/09/2018 11:30 AM - Greg Shah

- Status changed from New to Closed

- % Done changed from 0 to 100

I'm closing this task since all the support needed for the current application is implemented and has been tested as working (at least so far).

Tasks #3505 and #3506 have been opened for the development items that were deferred.

#385 - 09/16/2019 02:12 PM - Greg Shah

- Related to Feature #3858: implement a widget that replaces the Microsoft ProgressBar OCX (VB 6.0 common control) added

Files

create_object_error.jpg	156 KB	10/06/2017	Eugenie Lyzenko
decimal_precision_issue.jpg	185 KB	10/14/2017	Eugenie Lyzenko
com_error_regular.jpg	187 KB	10/24/2017	Eugenie Lyzenko
com_no-error_option.jpg	20.6 KB	10/24/2017	Eugenie Lyzenko
com_handle_prop_correct.jpg	41.8 KB	11/02/2017	Eugenie Lyzenko
com_handle_prop_incorrect.jpg	49.4 KB	11/02/2017	Eugenie Lyzenko
raw_prop_set_get_test_20171106a.jpg	43.3 KB	11/06/2017	Eugenie Lyzenko
com_wrong_array_prop.jpg	154 KB	11/09/2017	Eugenie Lyzenko
input_output_integer_method.jpg	36.7 KB	11/11/2017	Eugenie Lyzenko
ro_property_set1.jpg	66.2 KB	11/18/2017	Eugenie Lyzenko
ro_property_set2.jpg	26.6 KB	11/18/2017	Eugenie Lyzenko