

*This is an excerpt from the IRC chat on Saturday, October 6, 2001. Andy Ireland, of **PlugWare.Com Solutions, Ltd** talked about interfaces, COM objects and how to use them in Clarion. Side conversations, parts and joins and other notices are removed for clarity. Also, the conversation is reformatted to read like an interview.*

Andy: I'll be able to start the ActiveX SDK next weekend depending on how quickly I get this other job done. I've got most of the interfaces defined. I will start with the basic support. COM objects I think I can do now.

Q: ActiveX SDK? That's something to look forward to.

A: Yes, after I ship the container support. I have a test app with the web browser control running in it in Clarion, all hand coded in Clarion classes. Pure API.

ActiveX, COM and OLE are the things I want to concentrate on getting good support for into Clarion. COM is done, ActiveX use is almost done, creation of COM and ActiveX objects next and then full-blown OLE support.

Q: Andy - OK, let's cut to the chase. What can the average Clarion developer do with these classes?

A: Use ActiveX controls without them going bang! Uses COM with relative ease, then eventually create ActiveX and COM objects.

Q: In what context would it be good to have an ActiveX control in my application?

A: There are many more 3rd party products then at your disposal.

Q: Like what kind of "zing" does this bring to my apps that my clients would like?

A: Anything from calendar controls to imaging and mapping controls to full-blown report engines etc. With a MicroSoft feel to the UI unlike many [existing] market products.

Q: Sheridan Calendar controls.

A: Exactly.

Q: Imaging controls, graphing controls, I even know of a neural net solution that's ActiveX.

A: I could do a wrapper for GCAL Pro really quickly for instance using the ActiveX.

Q: OK, this sounds good. I am tired of looking at battleship gray.

A: It took me 5 minutes to do the web browser control. Here's the example code (not many lines):

```

OleWindow WINDOW('OleWindow'), AT(, , 205, 179), SYSTEM, GRAY, RESIZE, IMM
    END
OleIniter          COLEIniter
WebBrwsr          WebBrowser
CtrlSite          &COLECtrlSite
CtrlContainer     COLECtrlContainer
fBrowserCreated   bool
ResizeCount       long

code
if OleIniter.IsInitialised()
    open(OleWindow)
    accept
        case event()
            of event:openwindow
                CreateBrowser
            of event:sized
                if fBrowserCreated and ResizeCount > 0
                    CtrlContainer.Resize().
                    ResizeCount += 1
                end
            end
        end
    end
end
if ~CtrlSite &= null
    dispose(CtrlSite).

```

and then...

```

CreateBrowser procedure
rcWindow      like(_RECT_)
szURL         cstring('www.microsoft.com')
ProgID        CStr
szProgID      &cstring
lpszProgID    long
hr            HRESULT, auto

code
hr = ProgIDFromCLSID(address(__IWebBrowser), lpszProgID)
if hr = S_OK
    if ProgID.Init(lpszProgID, true, 0, false)
        szProgID &= ProgID.GetCStr()
    end
    CoTaskMemFree(lpszProgID)
end
GetClientRect(OleWindow{prop:clienthandle}, rcWindow)
if CtrlContainer.Init(OleWindow{prop:handle}, OleWindow{prop:clienthandle})
    CtrlSite &= new COLECtrlSite
    CtrlSite.Init(CtrlContainer, OleWindow{prop:clienthandle}, rcWindow)
    hr = WebBrwsr.Init(CtrlSite)
    if hr = S_OK
        hr = WebBrwsr.Navigate(szURL)
        fBrowserCreated = true
    end
end
end

```

See? Not difficult at all. Ignore the ProgID code, was testing the looking up of names from IID's so I could instantiate based upon simple name such as webbrowser.2 or whatever its name is.

Q: OK, really dumb question: How does this fit with an ABC template generated application? How much work to add these features?

A: No templates planned just yet. The classes are ABC compliant.

Q: Will I be able to receive events from an IE control?

A: Yes. I do already.

Q: Andy - OK, then assuming this code is complete, functional and perfect, is it a lot of work for the average Clarion developer to insert into their app files?

A: No, it is simple, but can be [complex] if they want to take complete control and override the default code i.e. pass in their own control container interfaces for custom handling.

Q: Andy, I would love to see it. Especially if I can get events from IE.

A: Optionally pass in the events Interface or derive the OnEvent method. Events from IE are a trivial thing. You guys want to know how it works? I'll be happy to talk about it.

Q: Andy; I'm game. Lets see some code.

A: Ok, the key is something called an Event Sink. What happens is your application registers it's event sink (an interface in your code that the ActiveX can talk to). Actually there can be many but usually only one is called the primary. When the control is created it talks to an interface called IoleClientSite.

It asks this interface for the event sink interface. Now, here's the crux. If you do not support the native interface, in the case of the webbrowser control it's IwebBrowserEvents.

You give it an IDispatch interface. This interface is a completely generic interface. This is what Clarion always uses. It knows nothing about your ActiveX and therefore has to ask the control questions such as, "This eventID is what?"

Somewhere in this process, the RTL is screwing up. My code allows you to either use this IDispatch or pass in the native interface. If you pass in the native interface, it calls into it, passing you the info as you expect it. If not it is passed in a generic format and needs some processing.

Ok, I'll pause for questions. Shall I explain what the interfaces are and do?

Q: Please!

A: OK, there are two main interfaces in COM: IUnknown and IDispatch. IDispatch is derived from Iunknown. So IDispatch has all the methods of IUnknown and then also it's own.

Q: Andy - question. The "I" is for "Interface, right?

A: Yes

Q: Please explain the Dispatch and Unknown labels.

A: The IUnknown is the generic interface for talking to a COM object and it contains the following....

```
IUnknown
QueryInterface
AddRef
Release
Interface
  procedure(REFIID riid, *long ppvObject), long, pascal
  procedure(), long, pascal
  procedure(), long, pascal, proc
end
```

Q: IUnknown meaning "I don't know what it could possibly be, but give it to me anyway"?

A: No, IUnknown meaning "I do not know or care what the object is but I need to be able to talk to it in some generic manner."

Q: OK. That makes sense.

A: How else could I say..."Give me something if you support it"?

Q: And what does Dispatch mean?

A: That where QueryInterface comes in, but hold one... not finished with IUnknown.

Q: OK.

A: [To continue,] AddRef and Release allow me to keep a count of how many instances of code are using my interface.

Q: OK, that makes sense.

A: [Further,] when the release method is called and if the count hits 0, the object is destroyed and cleaned up.

Q: Kind of like what the old CheckOpen function did.

A: The QueryInterface allows me to ask for a supported interface but with open files. I.e. if you support a given interface, give it to me. The ref count on that interface gets incremented. When I am done with it, I release it. [This is] easy so far.

Q: Yes, perfect sense. Anyone not follow?

A: All COM objects, be they ActiveX, COM or OLE support this. They must. Ok, so where does IDispatch come in?

IDispatch extends this model to allow me to give an object commands. For instance set/get a property, execute a method or handle an event. The interface is:

```
Idispatch          interface(IUnknown)
GetTypeInfoCount  procedure(*long pctinfo),HRESULT,pascal
GetTypeInfo       procedure(long iTInfo, long lcid, long ppTInfo),HRESULT,pascal
GetIDsOfNames     procedure(long riid, long prgszNames, long cNames, |
                  long lcid, long prgDispId),HRESULT,pascal
Invoke            procedure(long dispIdMember, long riid, long lcid, short wFlags, |
                  long pDispParams, long pVarResult, long pExcepInfo, |
                  long puArgErr),HRESULT,pascal
end
```

Q: So IUnknown says "give me your supported interface and I will track the instances for you.

A: yes and no. IUnknown allows me as a consumer to ask and receive an interface the object that gives me the interface increments the ref count. I release it when done. If my code is the object, I increment, the consumer releases

Q: IDispatch says "I will handle any communication you need", correct?

A: You are correct, but IDispatch goes a little further. You can also ask it to give you the detailed information about the control or object such as what it's interfaces are and the prototypes on the fly as this is stored in the registry.

This allows you to dynamically create interfaces as needed that match the object rather than being generic. Generic comes at a cost -- it is slower. This is because everything must then be interpreted instead of calls made directly into the relevant native interfaces.

Q: So I (the developer) do not really need to know the interface makeup?

A: Right. Is this clear so far?

Q: OK, here is the next question. If I do not know the context or makeup of an interface, how do I use it? Or even if I want to use it?

A: The difference is called late binding for generic, early binding for native. You need to know what some of the info is which is why IDispatch supports you asking. For example, lets say I use the WebBrowser control.

If I want to trap an event, I need to know what the ID for that event is whether I like it or not, for it to be useful. Otherwise I may as well not bother. So late binding is what VB uses so that I do not need to code anything special to use a control. But there are some things I either must know or must ask otherwise my object just looks pretty on the screen.

Q: Like which events are even relevant for the control?

A: Yes. Here is another example. If I use the web browser control, how do I go to a web page if I do not know how to ask the question? Therefore as an implementer I still must know implementation details specific to the control. The difference is, my container does not need to know this therefore only my apps business logic changes, not my classes.

Q: Andy - would you give an example of one event for a web browser and how it would be used?

A: Lets say I ask the browser to navigate to www.softvelocity.com. When it does this, it fires two events, one before navigation, and one after. It actually fires more but this is for simplicity. The one before allows me to return true or false to allow navigation or not i.e. a link may have been clicked but I want to trap and disallow this is where I trap the event change the value and return.

It's easy. But we can be more creative. Lets say I create a web page. On this page is a set of links, which are controls. Each of those represents valid controls in a screen editor. When one of those is clicked, I trap the event, stop the navigation click on the window, it sends a msg and creates that control.

Now, I have created a window formatter whose control list is an html page and it uses dynamic modules to define the controls so not only can I add controls to the system but also amend the html accordingly.

So the events can be used in many ways. Is that clear? Likewise that could control navigation in an app and be the menu, which can be changed without changing code.

OK, so where are we at? We know what the base interface is for COM objects.

We know all objects be them ActiveX, OLE or COM are COM objects. It's all MS semantics. COM added some features but for arguments sake this is the case.

So where does a control differ? A control has a GUI and it needs to know where, how and on what it is displaying itself. This requires a more extensive conversation, i.e. more info, more advanced data, and if it relied on the two bases interfaces, it would be so chatty that it would run slower than water in the north pole.

So GUI controls (ActiveX controls) add some interfaces for this purpose. But the interfaces are needed on both sides of the conversation, so it talks to me and I respond in kind. If it talks German, so must I.

By that I mean if it has a GUI conversation, I need to support that. It asks where it is displaying itself and how to present itself, I respond with a window, a frame and some details of the format i.e. display as a BMP on a surface which is for a printer or for a screen.

Additionally, I tell it how big it can be and where it can and can't draw. For this I need extra interfaces. On the container side these include:

```
IoleInPlaceFrame
IoleInPlaceUIWindow
IoleInPlaceSite
IoleClientSite
IAdviseSink
IoleControlSite
IDispatch (Ambient properties)
IPropertyNotifySink
IDispatch (Events)
```

These need to be implemented in an ActiveX container and amounts to about 2000 lines of code give or take. But it depends on the amount of support you wish to allow, not all of this is mandatory.

Q: The problem, I think, is that there is so much of it. It is hard to take a piece of data, absorb that, take another etc.

A: Then we should break up into lessons if people are interested. COM is easy, there is just lots of it. Here's a snippet of code:

```
COLECtrlSite.CreateCtrl procedure(long rclsid, long riid, |
                                long renderopt, *long pvObject)
hr          HRESULT, auto
lFormatEtc like(FORMATETC)

code
? assert(~self.InPlaceSite &= null)
? assert(~self.ClientSite &= null)
? assert(~self.AdviseSink &= null)
? assert(~self.AmbientDispatch &= null)
? assert(~self.EventDispatch &= null)
? assert(~self.Storage &= null)
if (~self.InPlaceSite &= null) and (~self.ClientSite &= null) and |
    (~self.AdviseSink &= null) and (~self.AmbientDispatch &= null) and |
    (~self.EventDispatch &= null) and (~self.Storage &= null)
lFormatEtc.cfFormat = CF_BITMAP
lFormatEtc.ptd = 0
lFormatEtc.dwAspect = DVASPECT_CONTENT
lFormatEtc.tymed = TYMED_GDI
lFormatEtc.lIndex = -1
hr = self.Storage.Init()
if hr = S_OK
    hr = OleCreate(rclsid, riid, renderopt, 0, self.ClientSite.IoleClientSite, |
                self.Storage.GetStorage(), pvObject)
if hr = S_OK
    self.pObj = pvObject
    hr = self.Advise()
    if hr = S_OK
        hr = self.SetEventDispatch()
```

```

        if hr = S_OK
            self.bInitialised = true.
        end
    end
end
end
return hr

```

This creates a control, i.e. create an ActiveX before it displays however, lots of conversations take place so one must implement everything before creation will occur.

See where I tell it how to display? For a window, this is the simplest way to setup display info. I allow this to be overridden so COM experts can do more complex handling.

See the OleCreate line. This is where I give it my IOleClientSite interface. It asks this for whatever it needs from me. Think of this interface, as it's main conversation conduit for it to talk to me. I also give it a storage interface for it's caching and it returns its base interface in pvObject. I hold onto this interface and release at the end this is my way to talk to it.

Therefore, a conversation is setup. Then I must simply support whatever it needs. Advise is where I tell it what my *AdviseSink* is and *SetEventDispatch* is where I tell it what my event interface is.

When an event occurs, it passes it to me via this, and then it starts getting really heavy! But if you understand the basics, it is very simple just lots to do.

Are there any final questions? We'll end the lesson here. Shall I continue next week?

Q: Please!

A: I'll do a brief recap on this then go further.

Q: But in essence, I created a communication bridge (interface) and then send messages back and forth.

A: Yes

Maybe leave GUI controls and talk about COM, DCOM and something like ADO, but several of them with different roles in the grand scheme of things.

Q: Once I know the messages, then I have my conditional code.

A: Yes, for instance, Navigate.

Q: Just trying to make as simplistic overview as I possibly can.

A: This is just a method ID. It says a method of x address is mapped to this equate, when I ask for it, call it.

Q: I like that.

A: That is what late binding does hence why late binding is slow and why VB apps tend to be slow. Early binding is yes I know the interface, in fact I have asked for it explicitly and I will call directly, hence just a call in Asm.

Q: If it is just a reference to an address, does that not mean an extra lookup? So why so slow if this is the case?

A: Yes, but also passing the params in a generic way i.e. as a safearray of variant structures hence the extra work involved.

Q: I see, the extra data conversion?

A: This must be marshaled. I will leave marshalling for another time as is complex.

The crux though is this, marshalling is essentially a way to wrap the data up and pass it cross process boundaries, etc. Hence there is a certain amount of synchronization going on.

Oh, for those that don't know, a variant is a group that allows you to wrap up a data type in a generic way like an ANY variable.

Q: Ah! That safearray problem was the big issue with my CW/CorelDraw COM interface. Never did find an understandable solution.

A: I posted code on comp.lang.clarion to resolve that plus the equates etc and prototypes plus some asm code for calling methods that need a variant passed by value as CW does not support this.

Q: Andy - Thanks a lot.

A: You're all welcome. Always willing to help the community. Oh and a cheap plug --Plugware Solutions.com Ltd will be releasing an ActiveX control container library and all sources as a product. It is a complete replacement of the OLE support in the RTL. Initially it will support ActiveX, but later, full-blown OLE container support. Then onto ActiveX / COM object creation SDK for Clarion.

Q: Andy - Not that it matters much, but what is the \$\$\$?

A: I've, not decided. But I will do two versions, one without source (cheap), one with (expensive'ish). COM I will post for free.

Q: Andy - I thought you did this for ADO. COM you reserved.

A: I need to put the COM classes into OpenSource. Yes, COM is mine, but it ships as source with the ADO so I may as well make it OpenSource.

Q: Andy - Don't forget the Clarion Magazine open source thingy.

A: It's for the community. Hope this talk was of value to people. Oh, and as another shameless plug, my company does consulting work for anyone looking for advanced features.

Group: Andy - thanks so much for the data.