

Clarion Magazine

online sales and delivery
for your applications & tools

Developer *PLUS*

A Progress Bar For Multiple Processes

Henry Plotkin does a lot of year-end processing. Process template procedures, one for each file, can do the work. But the user ends up looking at windows popping up all over the place. In this article, Henry shows how to use one progress bar window when running multiple processes.

Posted Thursday, July 08, 2004

PDF for June, 2005

All Clarion Magazine articles for June, 2004 in PDF format.

Posted Thursday, July 08, 2004

Recording Audio: An Introduction To OCXs, Part 1

OCX technology has been around for years, but not that many Clarion developers use OCX components. Ben Brady shows how to use a readily-available OCX to record and encode audio. Part 1 of 3.

Posted Thursday, July 08, 2004

Recording Audio: An Introduction To OCXs, Part 2

OCX technology has been around for years, but not that many Clarion developers use OCX components. Ben Brady shows how to use a readily-

Articles: 

News: 



News

[Rights to Use and Redistribute MSDE 2000](#)

[No JAGUAR At DevCon](#)

[xSearch C6.1 Compatible](#)

[Clarion 6.1 Production Release](#)

["Powered by Clarion" Image](#)

[Fomin Report Builder 2.88](#)

[DLL Tutorial](#)

[vuFileTools Available At Clarionshop](#)

[Simplified Software Templates C6.1 Compatible](#)

[Lodestar Software C6.1 Installs Available](#)

[UltraTree And Clarion 6.1](#)

[50% Off Firebird Lessons](#)

[BST Ver 3.61 Patch](#)

[New Passwords For All Super Templates](#)


[Problem With BoxSoft Super Stuff 6.15 Installation](#)

[ClarionMantis Update](#)


SURVEY

What security product do you use/recommend for Clarion applications?


Gitano's G-Reg/G-Sec

 6.3%


CapeSoft's SecWin

 39.1%

ComSoft's Big Registration Template

 3.1%


BoxSoft Templates

 4.7%


DAS Registration

0.0%


Armadillo

 10.9%

Other

 10.9%

None

 25.0%

64 responses

[Previous Surveys](#)

available OCX to record and encode audio. Part 2 of 3.

Posted Thursday, July 08, 2004

Recording Audio: An Introduction To OCXs, Part 3

OCX technology has been around for years, but not that many Clarion developers use OCX components. Ben Brady shows how to use a readily-available OCX to record and encode audio. Part 3 of 3.

Posted Wednesday, July 14, 2004

Bio: Andrew Guidroz II

Andrew Guidroz II isn't shy about his Cajun heritage. Actually he isn't shy, period. But even if you think you know Andrew, you'll probably find a few surprises here.

Posted Friday, July 16, 2004

A Function to Return Clarion Standard Time

Mark Riffey made the request: "I want to do TIME('12','24','06') and get a Clarion [standard] time just like I would do DATE('12','24','2004') and get a Clarion date." Steve Parker, with a little help from his friends, shows how it's done.

Posted Friday, July 16, 2004

Generating A Unique Registration Code

Registering software usually involves entering some sort of registration code. But how do you go about creating such a code to give to your customers, and then how do you validate in the

[PlugIT 3.0 Delayed](#)

[Icons Sale Ends Soon](#)

[Gitano \\$50 Summer Sale Continues](#)

[AdiTech Holiday Schedule](#)

[PDF-XChange/Tools V3 New Build](#)

[GWBSQLAutoInc Updated](#)

[BST 3.6](#)

[Clarion Release Candidate \(RC6\)](#)

[BG XML DICO New Release](#)

[In-Memory Database Driver Patch](#)

[CPCS Support](#)

[Gitano Software Updates](#)

[Fenix ASP.NET Generator At UK Meeting](#)

[Linking MySQL and SQL Server](#)

[EasyCOM2INC 1.12](#)

[Clarion 6.1 Release Candidate Build 5](#)

[Short Takes by Robert Zauere](#)

[xDataBackup Manager Lite, xNotes CR4 Compatibility](#)

[ImageEx 3 Beta](#)

[SQL Server 2005 Express Edition](#)

[UK Clarion User Group All-Day Meeting July 12](#)

[46 Best-Ever Freeware Utilities](#)

[Developers Logo Special - Correction](#)

One Year Ago In CM

[We Will Miss This Fine Lady](#)

[Weekly PDF For July 13-19, 2003](#)

[Breaking Reports On Computed Fields](#)

Two Years Ago In CM

[Another Look At Saving Printer Settings](#)

[Shrink-Wrapped Controls](#)

[The Program's Finished. Now What? \(Part 3\)](#)

Three Years Ago In CM

[Understanding Recursion - Part 1](#)

[Using The Web Browser OCX](#)

[Clarion News - August 2001](#)

Four Years Ago In CM

customer's application? Vince Du Beau demonstrates a registration scheme using a product type code and a user name.

Posted Monday, July 26, 2004

[Reading XML With The CenterPoint Classes](#)

Clarion 6 ships with wrapper classes for the open source CenterPoint XML class library. David Harms shows how to use these classes to read an XML file.

Posted Friday, July 30, 2004

[Using Virtual PC 2004](#)

Many developers keep old PCs around so they can run older operating systems for compatibility testing. But as Michael Lawson explains, you can now have this same capability on just one PC, using Microsoft's Virtual PC software.

Posted Friday, July 30, 2004

Looking for more? Check out the **[site index](#)**, or **[search the back issues](#)**.

This site now contains more than 700 articles and a total of over a million words of Clarion-related information.

[Search the news archive](#)

[Legacy to ABC: There is Another Way!](#)

[Clarion News - July 2000](#)

[Clarion 5.5 Gold Candidate: A First Look](#)

Copyright © 1999-2004 by [CoveComm Inc.](#) All Rights Reserved. Reproduction in any form without the express written consent of CoveComm Inc., except as described in the [subscription agreement](#), is prohibited.

Reborn Free

CLARION
online

A Progress Bar For Multiple Processes

by Henry Plotkin

Published 2004-07-08

I do a lot of batch processing. When a new year begins, I need to reset year-to-date totals in inventory, sales history, customers and vendors. I also import databases, mostly from ASCII, into my customer, vendor and inventory files.

These features present no difficulty in terms of effecting the desired functionality. However, all of them cause issues in the user interface. In the first case, resetting year-to-date totals, multiple Process template procedures, one for each file, can do the work. But the user ends up looking at windows popping up all over the place.

Even if I set each Process' window position as "fixed," one process window must close and, after a short delay, the next one opens. So the user still experiences window popping.

Furthermore, the user has no clue how many files remain to be reset. In other words, the user has no way of telling how far along the process is.

Until taking up this article, I thought that a Process would not work for importing ASCII files because the `Records()` function is not supported by the ASCII or BASIC drivers. I believed that without knowing the total record count, the progress bar in a Process template procedure would not visually indicate progress. I would end

up introducing a visual inconsistency by having to use a window and record counter, as in Figure 1.



Figure 1. Record counter window

A reviewer pointed out that the Process template will use bytes read when the Process's primary table does not use a key. Since ASCII files cannot have keys, they automatically fall into this category. A quick test verified that this is so. Unfortunately, this information does not help achieving my end. Being able to use a Process for processing ASCII files, while nice to know, still requires multiple windows opening and closing, one thing I specifically want to avoid (also, not all of the source files are declared in the dictionary).

What I think I want is a single window, displaying not only a progress bar but a status message. Using a single window, there would be no movement, therefore solving the popping problem. The status message would tell the user how many files have been processed and how many remain to be processed. For example:

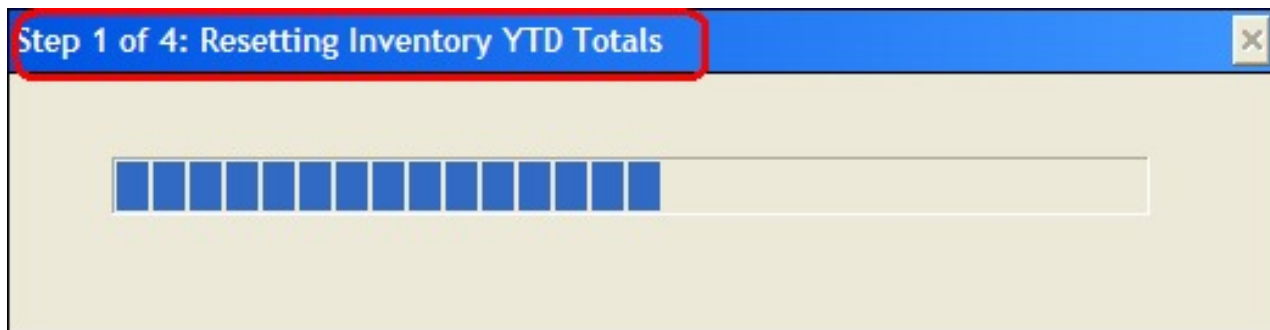


Figure 2. First file

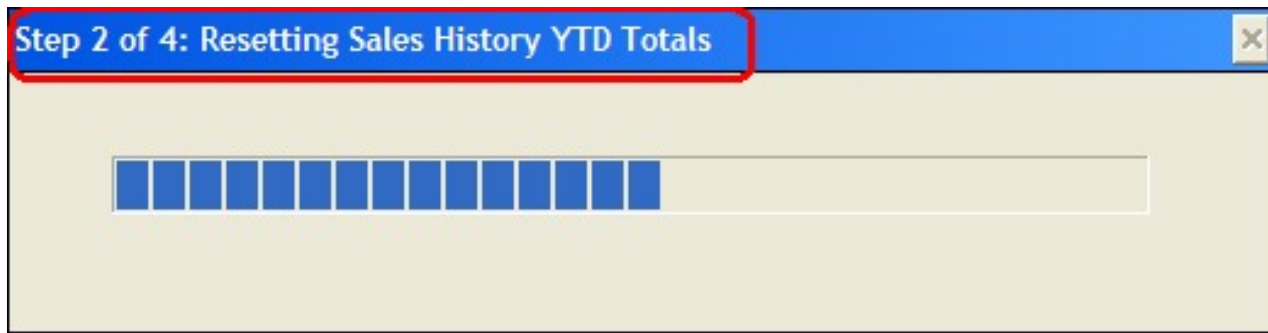


Figure 3. Second file

Etc.

A single window solves the "how far along am I?" issue because I can configure the window's status message dynamically, at runtime as I process each file, e.g.:

```
0{Prop:Text} = 'Step 1 of 4: Resetting Inventory YTD Totals'
0{Prop:Text} = 'Step 2 of 4:Resetting Sales History YTD Totals'
```

However, when using a Window template, I don't have a built in loop to process records like I do in the Process template. Of course, what I need is fairly simple code and I can do it myself. For example:

```
Logout(10,Inventory)
0{Prop:Text} = 'Step 1 of 4: Resetting Inventory YTD Totals'
Clear(INV:Record)
Set(INV:SysIDKey,INV:SysIDKey)
Loop Until Access:Inventory.Next()
  !field assignments here
  If Access:Inventory.Update()
    Stop('Error rewriting ' & Clip(INV:SysID) & ' ' & |
    Clip(INV:Description) )
    Rollback
    Break
  End
End
Commit
```

This block of code is repeated for each file.

Next, where does this code go? I really don't need the window's Accept loop because not only am I providing my own processing, there are no controls on the window for the user to interact with. So, I think the end of window initialization should do. At this point, files are

open, the window is open, relevant classes have been initialized, etc. Figure 4 shows where the code goes.

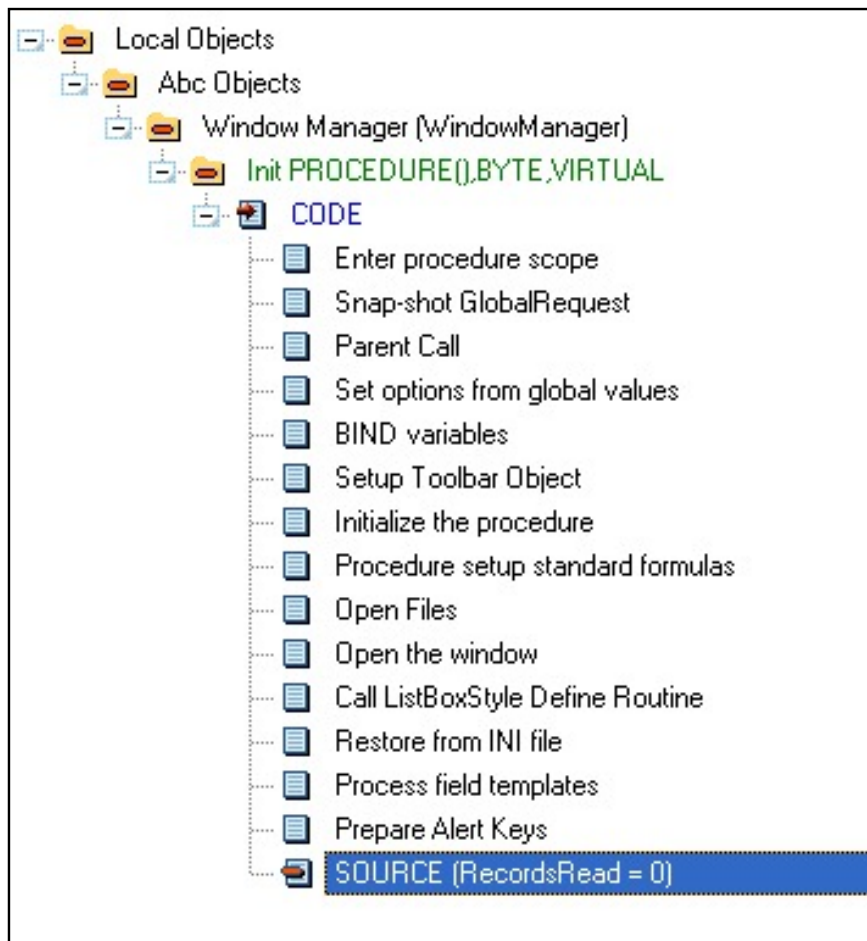


Figure 4. Where the code goes

One thing I must do is add a `Display` statement to my code. This is necessary because `Init` opens the window but `Accept` displays (and updates) it. With what I am doing, I will never get to `Accept`, so I have to force the window to display.

Because I don't use the window's `Accept`, I must also close the window. `Post(Event:CloseWindow)` when all the files are updated handles that.

This leaves one issue: the progress bar never updates. To solve this, I looked at the online help under the heading "PROGRESS (declare a progress control)." There, I see code remarkably similar to my own but with a progress bar implemented (I have highlighted the lines that I need to adapt to my code):


```

CODE
OPEN(Win)
OPEN(File)
?ProgressVariable{PROP:rangehigh} = RECORDS(File)
?ProgressBar{PROP:rangehigh} = RECORDS(File)
SET(File) !Set up a batch process
ACCEPT
CASE EVENT()
OF EVENT:CloseWindow
BREAK
OF EVENT:Timer !Process records when timer allows it
ProgressVariable += 3 !Auto-updates 1st progress bar
LOOP 3 TIMES
NEXT(File)
IF ERRORCODE()
BREAK
END
?ProgressBar{PROP:progress} = ?ProgressBar{PROP:progress} + 1
!Manually update 2nd progress bar
!Perform some batch processing code HERE
END
END
END
CLOSE(File)

```

The help also states:

If a variable is named as the USE attribute, the progress bar is automatically updated whenever the value in that variable changes. If the USE attribute is a field equate label, you must directly update the display by assigning a value (within the range defined by the RANGE attribute) to the control's PROP:progress property (an undeclared property equate -- see Undeclared Properties).

This is important because the default progress control USEs a field equate label and, therefore, I have to assign a value to *?<USE variable>*{Prop:Progress}, as shown in the sample code above.

The example in the online help uses the total number of records, I want to use percent completed. Two local variables and one computation will do it and my code becomes (note, when you populate a progress control on a window, the default use variable is

?Progress1):

```

Logout(10,Inventory)
TotalRecords = Records(Inventory)
RecordsRead = 0
0{Prop:Text} = 'Step 1 of 4: Resetting Inventory YTD Totals'
Clear(INV:Record)
Set(INV:SysIDKey,INV:SysIDKey)
Loop Until Access:Inventory.Next()
  RecordsRead += 1
  ?Progress1{PROP:progress} = (RecordsRead / TotalRecords) * 100
  Display
  !field assignments here
  If Access:Inventory.Update()
    Stop('Error rewriting ' & Clip(INV:SysID) & ' ' & |
Clip(INV:Description) )
    Rollback
    Break
  End
End
Commit

```

Best of all, because I do not have multiple procedures, with multiple windows to open/close/refresh, processing is not only visually smoother, it is noticeably faster.

ASCII files, however, appear to be a different story. With ASCII (and BASIC) files, `Records(file)` doesn't work. So, `TotalRecords = Records(Inventory)` will not return a useable value. Therefore the progress computation will be completely unusable (trying to divide by zero).

Clarion has a statement, `Bytes`, that can make the code I am using easily adapted to ASCII files.

The BYTES procedure returns the size of a FILE in bytes or the number of bytes in the last record successfully accessed. Following an OPEN statement, BYTES returns the size of the file. After the file has been successfully accessed by GET, REGET, NEXT, PREVIOUS, ADD, or PUT, the BYTES procedure returns the number of bytes accessed in the RECORD. The BYTES procedure may be used to return the number of bytes read in a variable length record.

So, Bytes(ImportFile) just after the file is opened but before a record is read gives me the analog of Records(ImportFile). My initial assignment becomes:

```
TotalRecords = Bytes(ImportFile)
```

And inside the loop, I use:

```
RecordsRead += Bytes(ImportFile)
```

No other changes are required.

Implementing this in a Window template procedure, processing ASCII files looks no different than processing any other file. Neat.

Processes are easy to use and do exactly what I need them to do. Now, so do Window template procedures (without that much more work). But, with Window procedures, my users, whose confusion level is often already high enough just showing up at work, are presented with a thoroughly consistent and predictable interface.

"hp" in fact prefers Hewlett-Packard printers but will use whatever is available. Born in New York City, hp is a self-taught Clarion developer doing a substantial amount of work for hospital gift shops.

Reader Comments

[Add a comment](#)

**A nice, simple solution to a problem. thanks for taking...
It seems to you that you could pass in an initial value...
"TotalRecords=Bytes(ImportFile)" can also be done with...**

Copyright © 1999-2004 by [CoveComm Inc.](#) All Rights Reserved. Reproduction in any form without the express written consent of CoveComm Inc., except as described in the [subscription agreement](#), is prohibited.

[Clarion Magazine](#)



Recording Audio: An Introduction To OCXs, Part 1

by Ben E Brady

Published 2004-07-08

Recently I spent a hundred dollars to purchase a small program used to encode audio files into the Macromedia Flash SWF format which allows you to put small buttons on your web site to play the audio file. (see www.firewallreporting.com for an example)

While this program was easy to use and did a reasonably good job, it had a few shortcomings. So I sent an email to the author asking him if he could make a few minor changes to the application to make it just a bit more useful. His reply, which came many days later (after I sent a second message) was basically "The program works as designed."

Given the author's poor attitude toward customer support and satisfaction, I decided I could write my own program to do the same thing, and if I was able to accomplish this feat, I would have yet another product to sell on my web site.

I realized right away that pure Clarion code probably wasn't going to cut it, and since I am not a C++ programmer, interfacing some C++ source to Clarion was out of the question. The only other possible solution was to find a reasonably priced OCX control that I could possibly redistribute without royalties.

This decision started me on a course of events in programming that few in our community ever experience: the use of OCXs in Clarion. I came to

realize I had quite a bit to learn.

Using OCXs in Clarion proved to be no easy task, but it wasn't impossible. I actually found two OCX controls to record and playback audio that were within my meager budget: the [MSSR210 OCX](#) from Product Technology Partners, Ltd. (US\$30) in England, and [DXVU Meter OCX](#) from fxJumpStart (US\$35). I really wanted a display that was similar to that used in [WinAmp](#) but the OCX for that product was several hundred dollars, which was way out of my price range (the audio technology used in WinAmp is owned by AOL).

After looking at the documentation for both MSSR210 and DXVU Meter, I decided the easiest product to use would be the MSSR210 OCX so that is the module I will concentrate on implementing in this article. I will eventually interface the DXVU Meter OCX just to see what I can come up with, because it really does have some very cool features. Perhaps that will be a subject for another Clarion Magazine article.

Since I knew next to nothing about OCXs my first stop was the ubiquitous Clarion documentation. The Language Reference Manual – Appendix A provides documentation for using OCXs, but for some reason the way it was presented just didn't make a lot of sense to me. So my next stop was the Clarion Magazine web site. I found a great article by Morten F. Thomsen entitled [Getting Started with OCXs](#). This really helped get me going, although it was fairly rudimentary and was written for Clarion 4.

One of the key factors in my deciding to use the MSSR210 OCX was the documentation itself. It is very easy to understand, presented well and very comprehensive with respect to the methods and properties for the OCX.

The OCX appeared to have all of the methods I would need to record and playback any type of WAV file format I would need, as the following Visual Basic prototypes show.

Summary of methods

- SHORT errorCode = ChooseAcmFormat([in] LONG hwndOwner,[in] STRING Title)
- SHORT errorCode = ChooseSpecificAcmFormat([in] LONG hwndOwner,[in] STRING Title, [in] SHORT WaveFormat)
- SHORT errorCode = DeleteSamples([in] LONG nSamples)
- SHORT errorCode = DeleteTime([in] DOUBLE time)
- SHORT errorCode = GetAcmFormatAsString([out] STRING Format)
- SHORT errorCode = GetCurrentFormatAsString([out] STRING Format)
- SHORT errorCode = OpenExistingOrNewFile([in] STRING Filename)
- SHORT errorCode = OpenNewFile([in] STRING Filename)
- SHORT errorCode = OpenTemporaryFile()
- SHORT errorCode = Register([in] STRING name, [in] STRING keycode)
- SHORT errorCode = RePack()
- SHORT errorCode = Reset()
- SHORT errorCode = ShowMixer([in] SHORT mixerType)
- SHORT errorCode = StartPlayback()
- SHORT errorCode = StartRecording()
- Stop()

Callback Events

- GetLevels([out] SINGLE leftLevel,[out] SHORT leftOverload, [out] SINGLE rightLevel, [out] SHORT rightOverload)
- NewVoxState([out] SHORT VoxState)
- PlaybackStopped([out] SHORT errorCode)
- RecordingStopped([out] SHORT errorCode)

I was able to extrapolate the following data type cross-reference from the MSSR210 documentation.

VB Data Type	Clarion Data Type
STRING	CSTRING
SHORT	SHORT

LONG	LONG
SINGLE	DECIMAL
DOUBLE	REAL
VARIANT	ANY
COLOR	LONG

Getting my hands dirty

Starting out I created an SDI menu window and populated it with a standard menu bar with choices for File and Help menus. I wouldn't need the Edit, Browse, and Window menus so I scrapped them. I added the Exit menu item to the File menu and the About menu item to the Help menu. Now I could start creating my application to test the MSSR210 OCX.

Next I populated the MSSR210 OCX on the screen with the following properties:

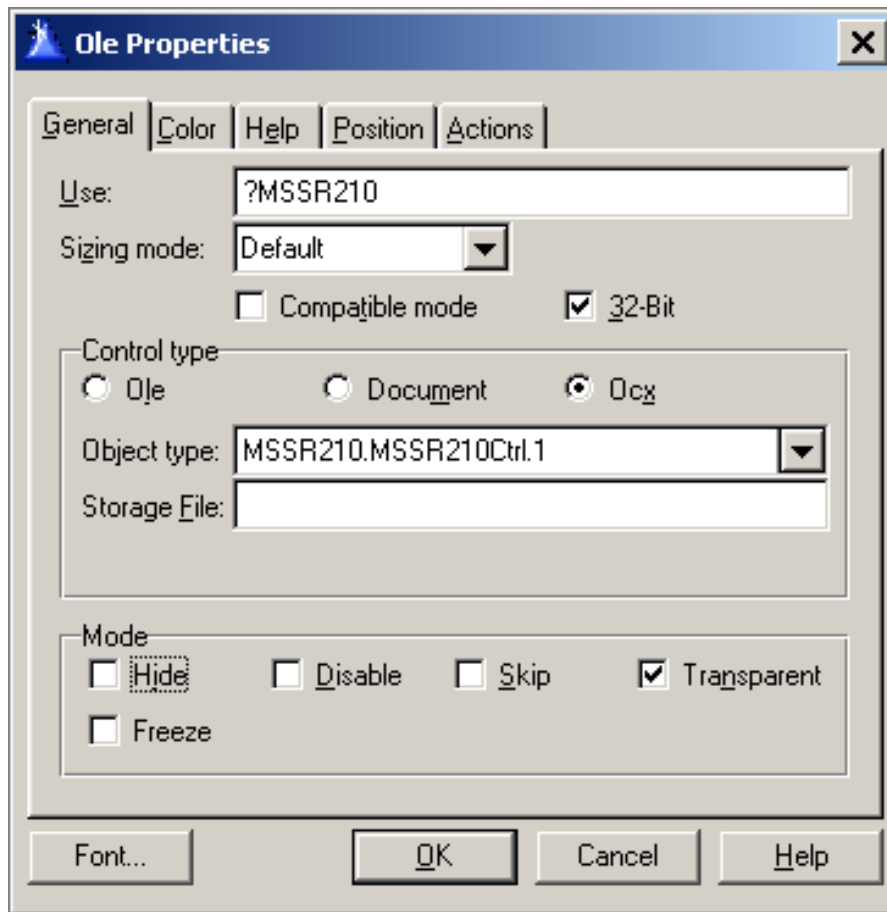


Figure 1. MSSR210 OCX Settings

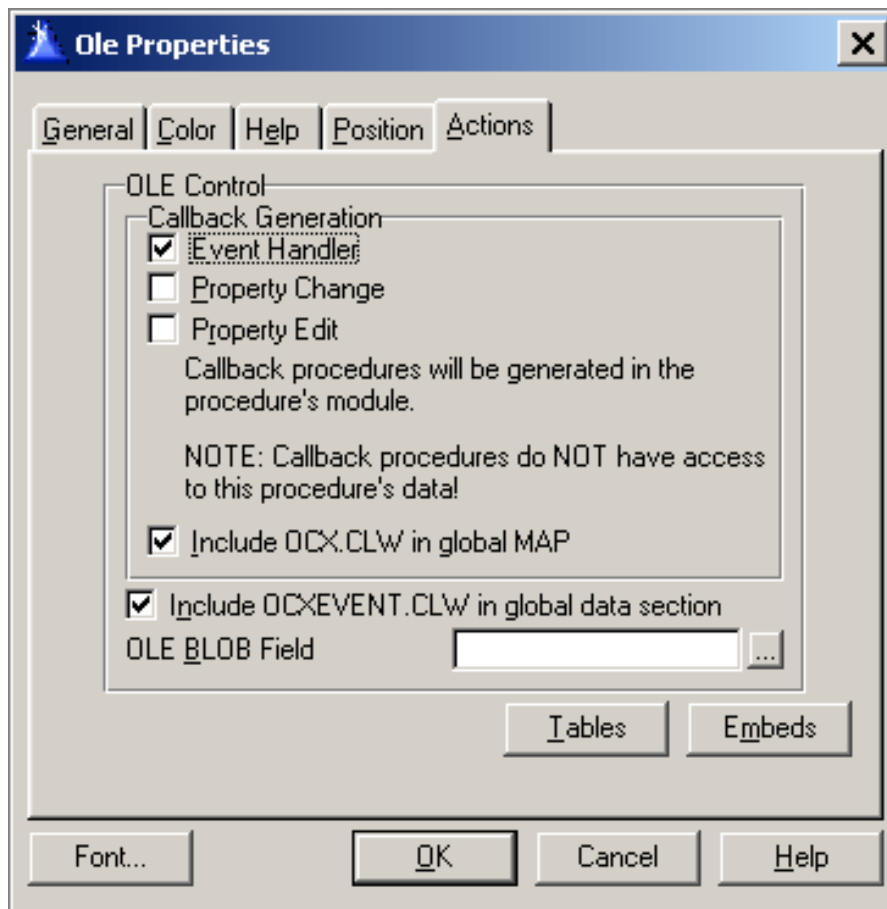


Figure 2. MSSR210 OCX Action Settings

Notice the Event Handler checkbox setting in the Actions tab control template dialog. This will generate an embed point for your code to execute when a callback event is received from the OCX by Windows.

The MSSR210 OCX does not require the use of the callback events, however callbacks do provide a couple of functions that I thought would be useful, such as alerting the user an error occurred during playback or recording and the ability to log the audio levels to a queue or a log file to process into a graph of some sorts. I will expand on this later in the article.

There are also embeds for detecting property changes or property edits (these are not used by this particular OCX).

The MSSR210 control requires that you register the control in your application, but the control does not use an OCX license file. So the first thing you do in the Before Initializing OLE Control (%BeforeOLEInitialization) embed point is to assign your registration name and keycode to the OCX properties using the Register() method, as follows:

In that embed I put in a call to a routine to initialize the OCX; I tend to modularize functionality in routines as much as possible.

```
DO MSSR210_Init
```

Then in the procedure routines embed I coded the following routine:

```
MSSR210_Init          ROUTINE
! Initialize the MSSR210 OCX Control
! Initialize the MSSR210 Error checking procedure
MSSR210_ErrorInit    ! this must be done first
! Set the display properties of the MSSR210 OCX Control
SETPOSITION(?MSSR210,16,55,136,8)
?MSSR210{'BackColor'} = 04E0605H
?MSSR210{'LowColor'}  = COLOR:Lime
?MSSR210{'MidColor'}  = COLOR:Yellow
?MSSR210{'HighColor'} = COLOR:Red
```

The above code is necessary because if you do not explicitly set this OCX's position, it will always display at 0,0. Next, assign the registration values for the OCX:

```
! Set OCX Registration values
MSSR210_name = eMSSR210_name
MSSR210_keycode = eMSSR210_keycode
BIND('MSSR210_name',MSSR210_name)
BIND('MSSR210_keycode',MSSR210_keycode)
! Check registration
MSSR210_Error = ?MSSR210{'Register('& MSSR210_Name |
    & ',' & MSSR210_Keycode & ')'}
MSSR210_ErrorCheck
! Reset the OCX control
MSSR210_Error = ?MSSR210{'Reset()'} ! Reset the OCX
MSSR210_ErrorCheck
EXIT
```

Notice that you must use the Clarion BIND command to allow communication between the values contained in the Clarion local variables and the properties in the OCX.

Whenever you interface with the OCX you must use the following syntax (bold { and } added for clarity) to set an OCX property (note the use of curly braces):

```
?OCX{'property_name'} = variable
```

To get an OCX property:

```
Variable = ?OCX{'property_name'}
```

The other thing to keep in mind is that OCX properties and method names tend to be case sensitive, so it is a good idea to specify the property or method *exactly* as shown in the documentation.

There are actually a couple of different ways to specify the property using Clarion, but I find the above syntax to be the most readable.

Calling a method uses very similar syntax; most of the time methods will return a value but that is not always the case:

```
errorCode = ?OCX{'method_name( ' & parameter1 & ',' & parameter2 & ')'} }
```

For methods that do not return a value or that do not accept parameters you can use this form:

```
?OCX{'method()' }
```

When building your OCX method statements, pay close attention to the use of the single quote character, commas and ampersands. Note that you can also pass a variable by reference to an OCX method.

A brief word about registering this OCX. You have 30 days to evaluate the OCX. If the Register() method fails you will get a 99 value in MSSR210_ErrorCode.

After the Register() method has been called, the next step required by the MSSR210 OCX is that the Reset() method is called to initialize the OCX.

The MSSR210_ErrorCheck procedure is a local procedure used to pop a message to the user based upon the error code returned by the OCX as follows (don't forget to put the procedure prototypes in your Local Data embed) The actual text for the error messages and the error message window title are contained in the APPNAME.EQU file and is initialized by the MSSR210_ErrorInit local procedure prior to initializing the MSSR210 OCX:

```
MSSR210_ErrorCheck Procedure()
I      SHORT

      CODE
      IF MSSR210_Error
        LOOP I = 1 TO RECORDS(ErrorQueue)
          GET(ErrorQueue,I)
          IF ERR:Error_Code = MSSR210_Error
            MESSAGE('Error Code: ' & ERR:Error_Code & ',' |
                    & ERR:Error_Msg, & |
                    ERR:Error_Title, ICON:Exclamation)
          END
        END
      END
      END
      RETURN
```

Error messages and message window titles are contained in the `MSSR210Demo.EQU` file.

I build a global queue called `ErrorQueue` to hold the values initialized below. Note the use of the Clarion `EXECUTE` command to process multiple statements in each line. `EXECUTE` is probably one of the least used, but most useful commands in Clarion.

```
MSSR210_ErrorInit      PROCEDURE()
I      SHORT
CODE
ERR:Error_Code = 99 ; ERR:Error_Msg = ERROR99 ; & |
      ERR:Error_Title = TITLE_ERROR99 ; ADD(ErrorQueue)
LOOP I = 1 TO 17
      EXECUTE(I)
          BEGIN ; ERR:Error_Code = i ; ERR:Error_Msg = ERROR1  ; & |
                ERR:Error_Title = TITLE_ERROR1  ; ADD(ErrorQueue); END
          BEGIN ; ERR:Error_Code = i ; ERR:Error_Msg = ERROR2  ; & |
                ERR:Error_Title = TITLE_ERROR2  ; ADD(ErrorQueue); END
          BEGIN ; ERR:Error_Code = i ; ERR:Error_Msg = ERROR3  ; & |
                ERR:Error_Title = TITLE_ERROR3  ; ADD(ErrorQueue); END
          BEGIN ; ERR:Error_Code = i ; ERR:Error_Msg = ERROR4  ; & |
                ERR:Error_Title = TITLE_ERROR4  ; ADD(ErrorQueue); END
          BEGIN ; ERR:Error_Code = i ; ERR:Error_Msg = ERROR5  ; & |
                ERR:Error_Title = TITLE_ERROR5  ; ADD(ErrorQueue); END
          BEGIN ; ERR:Error_Code = i ; ERR:Error_Msg = ERROR6  ; & |
                ERR:Error_Title = TITLE_ERROR6  ; ADD(ErrorQueue); END
          BEGIN ; ERR:Error_Code = i ; ERR:Error_Msg = ERROR7  ; & |
                ERR:Error_Title = TITLE_ERROR7  ; ADD(ErrorQueue); END
          ! And so forth...

      END
END
RETURN
```

I use a lot of equates in my applications because I can't stand hard-coded references (a throwback from my days as a Z80 assembler programmer), so my name and keycode are contained in my `APPNAME.EQU` file which is INCLUDED in my global embeds. I find it a lot easier to use an external equates file because once I get a sizable list of equates I can't always remember all of them; I keep the equates file open in a text editor while I am programming.

Next in the `After Initializing OLE Control (%AfterOLEInitialized)` embed I placed a call to another routine called `MSSR210_MainEntry`, as follows:

```
! Set the default properties for the MSSR210 OCX
DO MSSR210_MainEntry
```

The MSSR210_MainEntry routine is defined as follows in the Procedure Routines embed:

```
MSSR210_MainEntry          ROUTINE
! Set the properties for the MSSR210 OCX control
DO MSSR210_ReInit          ! Call the routine to set the properties
! Save the OCX properties to a compound storage file.
IF NOT EXISTS('MSSR210.OLR')
    ?MSSR210{PROP:SaveAs} = 'MSSR210.OLR\MSSR210'
END
```

The file MSSR210.OLR compound storage file is actually not used in this application to store the OCX display properties. If you create this file and enter it into the OCX properties on the General tab, and if for some reason the file cannot be found on the user's computer, the OCX does not get displayed and you end up with a "hole" in your window. The code shown above simply provides an example of how to create the compound storage file at runtime, in the event you do decide you want to use one.

[Read Part 2](#)

[Ben E. Brady](#) lives in what he affectionately refers to as the 'No-Tech capital of California', Dinuba, approximately 45 minutes southeast of Fresno in the heart of the Central Valley, with his wife Rita. A programmer for more than 30 years, he discovered Clarion 2.1 in 1989 after having programmed in several xBase dialects, BASIC and Z80 assembler. Ben is currently attending the College of the Sequoias in Visalia California, pursuing a degree in Computer Science and a teaching certificate in order to pass along some of his knowledge. Ben and Rita are also very active with the Tule Fog PC Users group in Visalia California, where they provide instruction to others in need of assistance with technology and give presentations to groups of all types. Ben is also the author of several very popular [firewall reporting](#) tools, written exclusively in Clarion for Windows, for users of personal firewall software such as BlackICE, ZoneAlarm, WinRoute Pro and XP Firewall.

Reader Comments

[Add a comment](#)

Copyright © 1999-2004 by [CoveComm Inc.](#) All Rights Reserved. Reproduction in any form without the express written consent of CoveComm Inc., except as described in the [subscription agreement](#), is prohibited.

Clarion Magazine



Recording Audio: An Introduction To OCXs, Part 2

by **Ben E Brady**

Published 2004-07-08

[Read Part 1](#)

The compound storage file is used to store OCX properties (presumably between user sessions). So far, each time I have saved the properties to the file, it does not contain a full set of the properties for the OCX. You can also create a compound storage file by right clicking on the OCX control in the window formatter and select 'Custom...' from the context menu. You will be shown a set of properties for the OCX and you can set them to the values you would like to store in the compound storage file. The first call in the `MSSR210_MainEntry` routine calls the `MSSR210_ReInit` routine which was coded so that I could re-initialize the properties at any time without worrying about other logic getting in the way. If you can isolate the OCX properties in this way, you can save yourself a lot of grief later on when it comes time to re-initialize them.

The `MSSR210_MainEntry` routine also contains some logic used to open the file and set some other properties that would only needed to be done once the OCX control was created:

```
! Display the selected ACM format
! Get the active ACM format if applicable
  DO MSSR210_ShowACMStatus
                                     ! but it can actually use any .ext
! Open a file for storage
  DO MSSR210_OpenWavFile
```

```
! Display the current file parameters
DO MSSR210_ShowFileStatus
! Set the current position to the beginning of the file for playback
POST(EVENT:Accepted,?BeginButton)
EXIT
```

The MSSR210_ReInit is as follows in the Procedure Routines embed:

```
MSSR210_ReInit          ROUTINE
! Set the properties for the MSSR210 OCX Control
! Set the style of the display
! 0 (MSSR_RECTANGLE) or 1 (MSSR_ELLIPSE)
?MSSR210{'Style'} = MSSR_Rectangle
Style = ?MSSR210{'Style'}
DISPLAY(Style)
! Set the number of segments for display
! 1 - 20, 8 (default)
?MSSR210{'NumSegments'} = MSSR_Segments
NumSegments = ?MSSR210{'NumSegments'}
DISPLAY(NumSegments)
! Set the orientation of the control
! MSSR_LeftRight (default), MSSR_BottomTop
! MSSR_RightLeft, MSSR_TopBottom
?MSSR210{'Orientation'} = MSSR_LeftRight
Orientation = ?MSSR210{'Orientation'}
DISPLAY(Orientation)
```

The code segments above determine how the OCX is going to be displayed once it has been created at runtime. You can display rectangular LEDs or oval LEDs, control the display of the number of LED segments and determine the orientation of the segments, either portrait or landscape, upside-down or right-side up.

```
! Set Sample Rate
! 6000, 8000, 11025 (default), 16000
! 22050, 32000, 44100, 48000, 64000
! 88200, 96000, or 192000
?MSSR210{'SampleRate'} = MSSR_AM_Radio_High
SampleRate = ?MSSR210{'SampleRate'}
SelectSampleRate = SampleRate
DISPLAY(SampleRate)

! Set Bits Per Sample
! 8 (MSSR_PCM8), 16 (MSSR_PCM16)
! 100 (MSSR_ACM)
! 106 (MSSR_ALAW - 8-bit A-Law)
! 107 (MSSR_MULAW - 8-bit μ-Law)
?MSSR210{'BitsPerSample'} = MSSR_PCM16
```

```

BitsPerSample = ?MSSR210{'BitsPerSample'}
SelectBitsPerSample = BitsPerSample
DISPLAY(BitsPerSample)

```

The MSSR210 OCX can record many different file formats, as long as they are supported by your sound card recording and playback devices. For my purposes I could use either 8 or 16 bit PCM or MP3. Since I didn't want to deal with the licensing issues of using an MP3 encoder, I opted for the tried and true Microsoft-sanctioned PCM. If you decide to use MP3 you will need to make sure your user has an MP3 codec installed on their computer. The FHG Radium codec is free for personal use and is highly recommended (do a Google search to find a download site). Other encoders, using more open licensing terms, are also available.

```

! Limit the recording time
! Set the value from equate default
RecordTimeLimit = MSSR_RECORDTIMELIMIT
RecordLimit      = MSSR_RECORDLIMIT
RecordLimit      = RecordTimeLimit * SampleRate
?MSSR210{'RecordTimeLimit'} = RecordTimeLimit
?MSSR210{'RecordLimit'}     = RecordLimit

```

The MSSR210 OCX also provides the ability to limit the length of the recording based upon the number of samples in the file, or the amount of time recorded. Since it is difficult to know every time (unless you specifically code it) what your `SampleRate` is going to be. I found it to be much easier to calculate the `RecordLimit` number of samples by always specifying the time and multiplying the `SampleRate` by the number of seconds.

Unfortunately, the author of the OCX used a `LONG` to describe the `RecordLimit` when he actually should have used a `REAL`. This presents a problem if you want to set up very long recording sessions, such as more than a couple of days at a time.

```

! Set Number of Channels
?MSSR210{'NumChannels'} = MSSR_STEREO      ! 2 (MSSR_STEREO), 1 (MSSR_MONO)
NumChannels = ?MSSR210{'NumChannels'}
SelectNumChannels = NumChannels
DISPLAY(NumChannels)

! Set the TemporaryFile flag to permanent
?MSSR210{'TemporaryFile'} = MSSR_PERMANENT

```

```

! 1 (MSSR_PERMANENT) or 0 (MSSR_TEMPORARY)

! Set Data Only
?MSSR210{'DataOnly'} = MSSR_FILE_WAV

! Enumerate the recording and playback devices to a queue
CLEAR(RecordingDevices)
FREE(RecordingDevices)
NRecordDevices = ?MSSR210{'NRecordDevices'}
LOOP i# = -1 TO NRecordDevices - 1
    REC:DeviceNumber = I#
    REC:DeviceName = ?MSSR210{'RecordDeviceName(' & i# & ')'}
    ADD(RecordingDevices)
END

! Set Recording Device to the Windows default
?MSSR210{'RecordDevice'} = MSSR_DEFAULT_RECORDING_DEVICE
RecordDeviceName = ?MSSR210{'RecordDeviceName(' |
    & ?MSSR210{'RecordDevice'} & ')'}
SelectRecordingDevice = RecordDeviceName
CLEAR(PlaybackDevices)
FREE(PlaybackDevices)
NPlaybackDevices = ?MSSR210{'NPlaybackDevices'}
LOOP i# = -1 TO NPlaybackDevices - 1
    PLY:DeviceNumber = I#
    PLY:DeviceName = ?MSSR210{'PlaybackDeviceName(' & i# & ')'}
    ADD(PlaybackDevices)
END

! Set Playback Device to the Windows default
?MSSR210{'PlaybackDevice'} = MSSR_DEFAULT_PLAYBACK_DEVICE
PlaybackDeviceName = ?MSSR210{'PlaybackDeviceName(' |
    & ?MSSR210{'PlaybackDevice'} & ')'}
SelectPlaybackDevice = PlaybackDeviceName

```

The MSSR210 OCX has the ability to use a recording device other than your sound card. This is very handy when you use a USB headset microphone to capture audio. You can easily switch the input to your audio capture device, or you can select one of the line in inputs from your sound card to capture music to a file as well. The code listed above shows how to load the various recording and playback device names into a queue for the user to select.

Next I have to set up the filename and the initial state of the PAUSE flag to display to the user.

```

! Set the default filename
Filename = MSSR_DEFAULT_FILENAME

```

```

?MSSR210{'Filename'} = Filename

! Set the Paused flag
?MSSR210{'Paused'} = MSSR_RUNNING
Paused = ?MSSR210{'Paused'}

! Set the VOX parameters to defaults
?MSSR210{'VoxChannel'}          = MSSR_VOXOFF
VoxChannel = ?MSSR210{'VoxChannel'}
VoxFlag = FALSE
?MSSR210{'VoxStartLevel'}      = MSSR_VoxStartLevel
VoxStartLevel = ?MSSR210{'VoxStartLevel'}
?MSSR210{'VoxStopLevel'}      = MSSR_VoxStopLevel
VoxStopLevel = ?MSSR210{'VoxStopLevel'}
?MSSR210{'VoxTime'}           = MSSR_VoxTime
VoxTime = ?MSSR210{'VoxTime'}
EXIT

```

VOX (voice activation) is also supported by the MSSR210 OCX to keep from recording long periods of silence while the sound level at the input source is below a specified threshold. The code above displays the property settings for VOX in a disabled state. To enable VOX you would simply change the `VoxChannel` from `MSSR_VOXOFF` to `MSSR_VOXBOTH` to monitor both channels of a stereo input or the left channel for a mono input.

3... 2... 1... Ignition?

All of the properties have been set and the OCX has been positioned on the screen where I would like to have it appear, in the appropriate orientation and style. All that is left is to code the logic for the various controls used to manipulate the OCX to handle audio files.

The MSSR210 OCX has several very interesting features that made it ideal for my purpose, the most interesting being the ability to insert audio into, and delete audio from, an already recorded audio file. The original audio application that I purchased would not allow me to do this. As a result, if I were recording an audio file and I screwed up, I would have to re-record the entire file. Because of this lack of functionality in the original program, it took me more than an hour and a half to record a five minute message that is up on my web site. (This is the downside of having three Dachshunds and trying to record audio during the day with the telephone ringing all afternoon and the UPS man knocking at the door. I now do any

major voicing at night after everyone, including the dogs, have gone to bed.)

Since I knew I wanted to be able to use the Insert() method to 'punch in' audio at a specific location in a file, I needed a control to trigger the method as well as Record and Play buttons. Naturally, I also needed a Stop button and perhaps a Pause button.

Ultimately, I decided to code a whole set of buttons, including a few that the QuickRecord2 program (which comes with the MSSR210 OCX) didn't have. So I ended up with Insert, Record, Stop, Beginning, Rewind, Play, Fast Forward, Ending, Pause, VOX and Delete.

Two other buttons that I thought might be useful would be used to trigger the Windows ACM dialog (Audio Compression Manager) and the Windows playback and recording mixers to make it handy to select recording inputs or setting volume levels while recording or playing back.

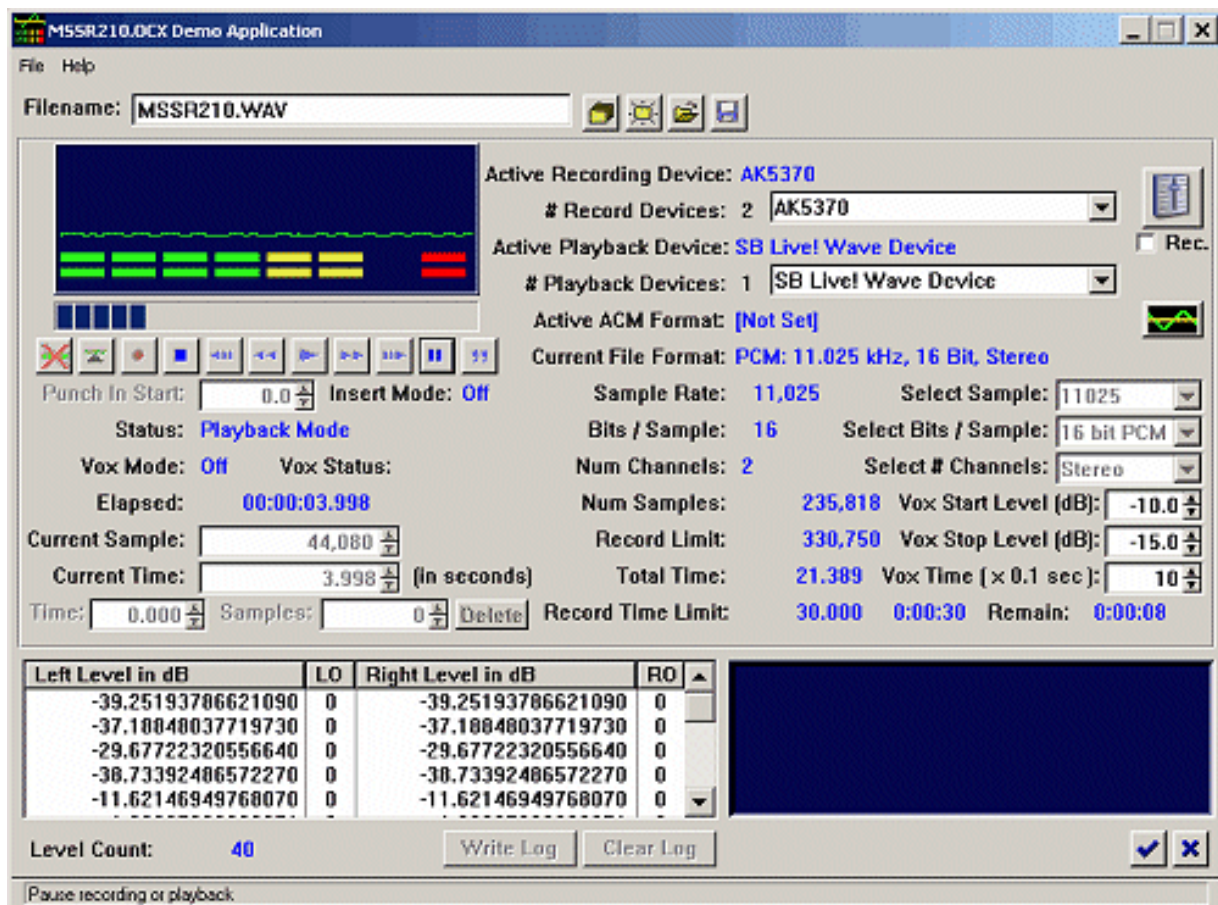


Figure 3. MSSR210 OCX playing a WAV file ([full sized image](#))

[Next week](#) I'll cover some of the more detailed logic involved in actually recording an audio file using the control.

[Download the APP & TXA](#)

[Download the source install package \(TXA, OCXs, no .APP\)](#)

[Ben E. Brady](#) lives in what he affectionately refers to as the 'No-Tech capital of California', Dinuba, approximately 45 minutes southeast of Fresno in the heart of the Central Valley, with his wife Rita. A programmer for more than 30 years, he discovered Clarion 2.1 in 1989 after having programmed in several xBase dialects, BASIC and Z80 assembler. Ben is currently attending the College of the Sequoias in Visalia California, pursuing a degree in Computer Science and a teaching certificate in order to pass along some of his knowledge. Ben and Rita are also very active with the Tule Fog PC Users group in Visalia California, where they provide instruction to others in need of assistance with technology and give presentations to groups of all types. Ben is also the author of several very popular [firewall reporting](#) tools, written exclusively in Clarion for Windows, for users of personal firewall software such as BlackICE, ZoneAlarm, WinRoute Pro and XP Firewall.

Reader Comments

[Add a comment](#)

Copyright © 1999-2004 by [CoveComm Inc.](#) All Rights Reserved. Reproduction in any form without the express written consent of CoveComm Inc., except as described in the [subscription agreement](#), is prohibited.

[Clarion Magazine](#)



Clarion News

[Search the news archive](#)

[Rights to Use and Redistribute MSDE 2000](#)

Johan van Zyl points out this link to information on using/redistributing MSDE 2000.

Posted Saturday, July 31, 2004

[No JAGUAR At DevCon](#)

SoftMasters regrets that it will be unable to attend DevCon 2004 to demonstrate JAGUAR. However, there will be a series of online presentations of JAGUAR starting on August 4th. If you are interested, please fill in the form indicating which days/hour range best suit your needs.

Posted Saturday, July 31, 2004

[xSearch C6.1 Compatible](#)

SealSoft's xSearch is now compatible with Clarion 6.1.

Posted Saturday, July 31, 2004

[Clarion 6.1 Production Release](#)

Version 6.1 Production Release is available for download. Build 9025 is the official production release of Clarion 6.1. Note that this installs onto C6 Gold.

Posted Saturday, July 31, 2004

["Powered by Clarion" Image](#)

Leroy Schultz has created a "powered by Clarion" animated GIFimage, which is freely available.

Posted Saturday, July 31, 2004

Fomin Report Builder 2.88

Fomin Report Builder 2.88, for Clarion 6.1, is now available.

Posted Saturday, July 31, 2004

DLL Tutorial

Dan Gorrell has posted a Clarion DLL tutorial (PowerPoint and Example apps). This tutorial is more about building function libraries than splitting up existing apps, but it does cover the basics.

Posted Saturday, July 31, 2004

vuFileTools Available At Clarionshop

vuFileTools is a collection of 38 functions that will help you manage files and control the environment in which your program is running. vuFileTools ships with full documentation and a sample application (written in C5.5).

Posted Saturday, July 31, 2004

Simplified Software Templates C6.1 Compatible

All Simplified Software templates are now C6.1 compatible. Only the Simshape templates need updating. A new full installation and a partial installation (for those who already have the templates) are now available for download.

Posted Saturday, July 31, 2004

Lodestar Software C6.1 Installs Available

All Lodestar Software C6.1 products are now available for download: RPM 6.1, AFE 6.1 & PNET 6.1.

Posted Saturday, July 31, 2004

UltraTree And Clarion 6.1

UltraTree Version 8 supports Clarion 6.1. No update is necessary for existing version 8 users.

Posted Saturday, July 31, 2004

50% Off Firebird Lessons

Robert Meyer is offering a 50% discount on a set of lessons on using Firebird with Clarion. There are currently six lessons with many more on the way. These lessons are accessed by a one time subscription which will be discounted by fifty percent (50%) until Robert has fifteen (15) lessons online or one-hundred (100) new subscribers, whichever ever happens first.

Posted Saturday, July 31, 2004

[BST Ver 3.61 Patch](#)

Bug fix in BST 3.61: An "Insert Record" by popup menu or doubleclick on the BST single resource scheduler for Week or Month Grid could load an incorrect date when using the "Split Time Field" option. The main installers and dem's have been updated as well. Install the patch, reload your app, compile.

Posted Saturday, July 31, 2004

[New Passwords For All Super Templates](#)

BoxSoft has released a 6.15 version for all the templates, with new passwords for each. Some of the templates contain new features and bug fixes, while others are basically unchanged. Read the revision history for each product (see the Product pages on our site) to learn what has changed, so you can determine whether you need to upgrade. If you upgraded after December 1, 2003, then there is no charge. Otherwise, the upgrade price varies between 15% and 35% of the regular price. For passwords and further upgrade information, please email mitten@mittensoftware.com or call (952)745-4941.

Posted Saturday, July 31, 2004

[Problem With BoxSoft Super Stuff 6.15 Installation](#)

The install for the recent Super Stuff 6.15 unintentionally contained some new code, resulting in compile errors when you make your base APP. Delete STABMHTBF.INC from your LIBSRC directory to solve this. For now BoxSoft is recalling 6.15, and returning to 6.12 (with the original password).

Posted Saturday, July 31, 2004

[ClarionMantis Update](#)

If you have reported bugs on clarionmantis.com, please check them out on 6.1 and mark them as dead or alive or whatever.

Posted Saturday, July 31, 2004

PlugIT 3.0 Delayed

The expected new version of PlugIT 3.0 has been delayed due to a death in Andy's family. He does expect to get something out next week (pending testing).

Posted Tuesday, July 20, 2004

Icons Sale Ends Soon

The 1st Icon Design icon sale ends soon. Included: XP Collection; XP People Collection; Red Theme Collection; Yellow Theme Collection; Blue Theme Collection; Green Theme Collection; Silver Theme Collection. 871 icons, 36,582 files, five sizes, four formats (ICO, PNG, BMP, GIF). Bundle valued at over \$752, now \$299. Includes bonus wallpapers and icons.

Posted Tuesday, July 20, 2004

Gitano \$50 Summer Sale Continues

Gitano's summer sale continues, with the following utilities available at \$50 each: gCal; gCalc; gNotes; gQ; gSec; gFileFind; Look Good Package. Or get the bundle for \$500 - reg. \$659 (gCal, gCalc, gNotes, gQ, gSec, gFileFind, gReg).

Posted Tuesday, July 20, 2004

AdiTech Holiday Schedule

Aditech staff will most likely be very hard to reach until about the end of July.

Posted Tuesday, July 20, 2004

PDF-XChange/Tools V3 New Build

A new build of PDF-XChange/Tools V3 is now available. There are some improvements in terms of speed/compression and formatting in the new V3 release which is now about 80 % complete. This release includes scanning now and many new apps and extended docs with a new utility added to convert PDF-Tools V2.5 to V3 compliant.

C5/C55/C6 compatible.

Posted Tuesday, July 20, 2004

GWBSQLAutoInc Updated

The GWBSQLAutoInc template has been updated as follows:

Self.Request is now only changed to ChangeRecord just before TakeCompleted is called, for better compatibility with other third party tools; The template can now be used with an ALIAS file, but the parent to the alias must be added to the list of files in the calling procedure.

Posted Tuesday, July 20, 2004

BST 3.6

Big Schedule Tamer Ver 3.6 is now available.

Posted Tuesday, July 20, 2004

Clarion Release Candidate (RC6)

Version 6.1 Release Candidate (RC6) is available for download.

Posted Tuesday, July 20, 2004

BG XML DICO New Release

A new version of BG XML DICO for Clarion 6.1 RC5 is now available.

Posted Tuesday, July 20, 2004

In-Memory Database Driver Patch

A patch for the In-Memory Database Driver is now available. This provides RC5 compatibility.

Posted Thursday, July 08, 2004

CPCS Support

CPCS support will be unavailable from 7/8/2004 through 7/10/2004.

All requests for support will be handled asap after 7/10/2004.

Posted Thursday, July 08, 2004

Gitano Software Updates

New installations for all Gitano utilities are now available for the latest C6.1 build. gNotes and gCal have also been updated and have had

some bugs fixed.

Posted Thursday, July 08, 2004

[Fenix ASP.NET Generator At UK Meeting](#)

Sebastian Talamoni will be holding another Fenix Presentation for the UK Clarion User Group on Monday, July 12.

Posted Thursday, July 08, 2004

[Linking MySQL and SQL Server](#)

From Timo Lahtinen comes a link to an article describing how to link Microsoft SQL Server and MySQL server by using the 'linked server' feature in SQL Server.

Posted Thursday, July 08, 2004

[EasyCOM2INC 1.12](#)

EasyCOM2INC 1.12 is now available. New features include: To avoid "Warning: Redefining system intrinsic" compiler warning the reserved words list was extended; There is a Clarion-IDL data types substitution list; Tree view has enum-type and enum-constant; Tree view has Copy to clipboard in the popup menu; Four new example apps (ICalendar, IShockwaveFlash, Windows Media Player, and XpdfViewer ActiveX control). The EasyCOM2INC utility is used to automatically creating Clarion include files with the definitions of COM-interfaces from IDL file. The Microsoft® Interface Definition Language (MIDL) defines interfaces between client and server programs. You can create a IDL file using MS Oleview.exe - OLE/COM Object Viewer. This administration and testing tool browses in a structured way, configures, activates, and tests all Microsoft Component Object Model (COM) classes installed on your computer. Requires Clarion 5.0, 5.5 or 6.x. Trial version available (10Kb limitation on IDL file size).

Posted Thursday, July 08, 2004

[Clarion 6.1 Release Candidate Build 5](#)

Version 6.1 Release Candidate 5 (RC5) is available for download.

Posted Thursday, July 08, 2004

[Short Takes by Robert Zaubere](#)

Roger Due has noted that Robert Zaunere is now writing a "Short Takes" column on the SV home page. The issue for the week of June 28 covers the new In-Memory Database Driver, some overlooked Clarion 6 features, and .NET topics. SoftVelocity is looking for feedback on a Clarion .Net compiler that would execute under Linux and/or as an option target the Mono runtime.

Posted Thursday, July 08, 2004

[xDataBackup Manager Lite, xNotes CR4 Compatibility](#)

SealSoft's xDataBackup Manager Lite (1.9) is now compatible with Clarion 6.1 (RC4), as is xNotes (1.6). Updated Demonstration program and Installation Kit for Clarion 6.1 (RC4), Clarion 5.5 and 5 are available.

Posted Thursday, July 08, 2004

[ImageEx 3 Beta](#)

The first beta of ImageEx 3 is now available. New features include: ; Read and write support for JPEG 2000 images; Write support for TIFF images (single- and multipage); Write support for PDF and PostScript files; Lossless jpeg transformations; New TWAIN engine; more than 100 different transition effects; A new color picker control similar to MS Office; Ability to include mouse cursor with screen capture functions; scrollbars for the viewer control, plus lots of other additions, improvements and fixes, and updated examples, docs etc. Existing ImageEx customers will be notified about the upgrade. Both the upgrade (US\$ 59) and full version (US\$ 249) as well as the updated ImageBrowsingBundle (US\$ 369) and competitive upgrade (US\$ 179) are available at ClarionShop now.

Posted Thursday, July 08, 2004

[SQL Server 2005 Express Edition](#)

A beta version of SQL Server Express Edition is now available. This product is the upcoming replacement for MSDE.

Posted Thursday, July 08, 2004

[UK Clarion User Group All-Day Meeting July 12](#)

If your planning on coming to the next UK clarion user group meeting

(12th July), visit the web site first. As with all meetings, delegates will be given a CD containing example apps and tools related with the days topic to refresh your memory when you get home plus special offers/discounts on third party products exclusive to delegates attending the meeting. The day's events begin at 9:00 a.m., and speakers include Simon Burrows, Veronica Chapman, Richard Rose, Sebastian Talamoni, and Andy Watts. The day wraps up at 5:30 p.m.

Posted Thursday, July 08, 2004

[46 Best-Ever Freeware Utilities](#)

Leroy Schulz points out this list of popular free utilities.

Posted Thursday, July 08, 2004

[Developers Logo Special - Correction](#)

In a June 23 news item we incorrectly reported the price on a 1stLogoDesign special. The correct price, including a logo, application icon, splash screen, a product box image and wallpaper is \$299, \$150 off the regular price. We apologize for any inconvenience. The original news item has also been corrected.

Posted Thursday, July 08, 2004

Copyright © 1999-2003 by [CoveComm Inc.](#) All Rights Reserved. Reproduction in any form without the express written consent of CoveComm Inc., except as described in the [subscription agreement](#), is prohibited.

[Clarion Magazine](#)



Recording Audio: An Introduction To OCXs, Part 3

by Ben E Brady

Published 2004-07-14

[Last week](#) I looked at some of the basic information you need to know to use an OCX in Clarion, as well as some specific information on initializing the MSSR210 OCX to get it ready for recording audio. Now it's time to jump right into the logic involved with creating an application that emulates the familiar "cassette recorder" interface used by many audio editing programs.

Showing Progress

I was rapidly showing progress and was quite pleased with the way things were turning out. I would have loved to have had a slider control that would allow the user to set the current sample or time position in the WAV file but unfortunately, Clarion doesn't have a slider control, unless you use a third party product. I looked at what was available from several third party vendors and could not find one that would support the granularity of control that I felt would be necessary to have the user control the specific placement of the `CurrentSamplePosition` in the file. (Look for an example application on my web site at www.clariondeveloper.com regarding the use of the `MSCOMCTL.OCX`, which contains a slider control.)

The next best thing was a progress bar so I could at least let the user know the relative position of the current sample or time position. I populated the progress control and then had to figure out how to make it work. It wasn't hard, but I had simply never used a progress control without using a Process Template procedure.

```
IF OCXStatus >= MSSR_Ready
  CurrentSamplePosition = ?MSSR210{'CurrentSamplePosition'}
  DISPLAY(?CurrentSamplePosition)
  ?AudioProgress{PROP:Progress} = CurrentSamplePosition / NumSamples * 100
  CurrentTimePosition = ?MSSR210{'CurrentTimePosition'}
```

```
DISPLAY(?CurrentTimePosition)
```

I placed the call to the `MSSR210_Progress` routine into the window timer embed and set the timer to 2 in the window properties dialog. The progress bar worked like a champ!

I later removed the window timer and called the progress bar update routine from my user-defined event handler; it just seemed like a better place to do it once I had a trigger for an event that would only update the progress bar when the user was actually recording or playing back audio.

I used standard buttons for each user interface function except for the `PunchInFlag`, `VOX` and `DeleteFlag` buttons, which use a `Checkbox` control with an icon specified to create a latching button.

For details regarding the embeds for the checkboxes, and all of the buttons used to control the logic of the recording, playback and positioning of the `CurrentSamplePosition`, refer to the source code accompanying this article.

I typically put button functions in routines as well, unless there is some reason why I cannot do it. This makes it easy to access in the Procedure Routines embed point in the main window of the application generator, instead of digging through each button embed point.

Processing Callback Events

One of the issues that really hung me up was the processing of the callback event functions for the OCX.

As I stated last week, the `MSSR210` OCX will operate just fine if you don't implement these callback events, but I thought they would be nice to have. Besides, I really wanted to learn how to do it. I read everything I could find about Clarion and OCX callback events and the more I read, the more I got confused.

I didn't quite understand that you couldn't process the callback events quite the same way as Clarion window events. Thanks to Sebastian Streiger from Argentina, I was able to finally get my mind around it – Sebastian provided me with an example to actually see how it is done. The confusion was primarily caused by the 'hand coded' example provided in the `C5.5` help file and the differences coded by the OCX callback event extension template as produced by the application generator. Hopefully, these discrepancies have been addressed in the help files for `C6.x`.

The key point to remember is, the callback event code does not have access to variables that are defined locally in the procedure where the OCX resides. You must use global variables to allow access to data elements. The other issue is that callback events must be handled very quickly, otherwise an important event might be missed by your event handler.

If you need to execute code based upon the values returned from the callback event, the best way to handle this is to create a user-defined event and `POST(EVENT:User, , Thread#, TRUE)` to the parent thread. You will also notice that I explicitly post the user defined events to the top of the event queue so they get processed as quickly as possible, just in case there are other events firing that would get in the way of the callback event processing.

Use the `EXECUTE` command to process your parameters if at all possible.

In the OLE Event Handler, Declaration Section (`%EventHandlerDeclaration`) embed for the OCX I declared the following variables:

```
Count          LONG
Res            CSTRING(256)
Parm          DECIMAL(7,2) ! All of the params for this OCX are numeric
```

In the OLE Event Handler, Code Section (`%EventHandlerCode`) embed for the OCX I added the following code:

```
! MSSR210 OCX Event Handler
IF OleEvent <> OCXEVENT:MouseMove ! Eliminate mouse move events
  ErrorFlag = FALSE
  Res = OleControlFeq{PROP:LastEventName} & ':'
```

The first thing I had to do is eliminate the events that would only get in the way. Mouse events are ignored by this event handler because the OCX doesn't process any callbacks based upon mouse events.

Next, I retrieved the event name of the event for the OCX by interrogating the `LastEventName` property.

```
IF OCXStatus > MSSR_Ready ! If OCX is recording or playing
  LOOP Count = 1 TO OCXGETPARAMCOUNT(Ref) ! Cycle through all parameters
    Parm = OCXGETPARAM(Ref,Count) ! getting each parameter name
    Res = CLIP(Res) & ' ' & Parm ! and concatenate them together
```

Then I had to determine whether or not I actually wanted to continue processing events based upon the status of the OCX. If it was recording or playing back audio

then I wanted to process the callback events.

Callback events usually have one or more data parameters used to communicate back to the main procedure. These parameters can be strings (typically CSTRING) or numeric data (usually LONGS). In this case, I had to deal with DECIMALS because the OCX is returning the levels of audio in dB (decibels) and SHORTS to return error codes and status flags. Thankfully, Clarion is forgiving in that you can store a SHORT into a DECIMAL without the compiler coughing up a hairball.

The OCXGETPARAMCOUNT provides a way to get the number of parameters that are sitting in the event queue waiting to be processed. One easy way to keep track of parameters is to concatenate them to the event name, as shown above.

The event name can be interrogated to determine where to post the data contained in the parameter. A simple LOOP structure makes it short work to assign the parameters to variables as shown below. In this case there is only a single parameter and based upon the value returned a user-defined event is posted back to the main accept loop event handler:

```
IF PARM <> 0 AND INSTRING('PlaybackStopped:',Res,1,1) OR |
  INSTRING('RecordingStopped:',Res,1,1)
  ErrorMessage = CLIP(Res)
  POST(EVENT:Playback_Record_Error,,MSSR210_Thread,TRUE)
ELSIF PARM = 0 AND INSTRING('PlaybackStopped:',Res,1,1) OR |
  INSTRING('RecordingStopped:',Res,1,1)
  ErrorMessage = Error0
  POST(EVENT:NormalTermination,,MSSR210_Thread,TRUE)
END
```

Some callback events are only applicable while certain OCX functionalities are active. Using a global variable to hold the OCXStatus makes it easy to conditionally process specific callback events. In the following case, I can set the display status to show the user if a VOX pause is active during recording.

```
IF OCXStatus = MSSR_Recording
  IF INSTRING('NewVoxState:',Res,1,1)
    NewVoxStateFlag = Parm      ! Update the global variable
    POST(EVENT:NewVoxState,,MSSR210_Thread,TRUE) ! Post the user event
  END
END
```

Multiple returned parameters are processed the same way. The code below processes a group of four parameters, returned as DECIMALS and SHORTS.

```
IF INSTRING('NewLevelValues:',Res,1,1)
  EXECUTE(Count)
```

```

        BEGIN          ! Add 6 dB to the measurements to allow for headroom
            IF Parm <> -9999.0
                LVL:LeftLevel = Parm + MSSR_MaxHeadroom
            ELSE
                LVL:LeftLevel = Parm
            END
        END
    END
BEGIN
    LVL:LeftOverload = Parm
END
BEGIN          ! Add 6 dB to the measurements to allow for headroom
    IF Parm <> -9999.0
        LVL:RightLevel = Parm + MSSR_MaxHeadroom
    ELSE
        LVL:RightLevel = Parm
    END
END
BEGIN
    LVL:RightOverload = Parm
END
END
END
END
LVL:Count = RECORDS(LevelQueue) + 1
ADD(LevelQueue)
LevelCount = RECORDS(LevelQueue)

```

Since I wanted to log the levels to a log file, I found it useful to store them into a global queue for processing after the recording or playback session had terminated. It would not be a good idea to write this data directly to a file during the time the callback event handler is processing as OCX events can fire very rapidly, and you could miss an important one if your computer is busy doing something else. It is a very good idea to optimize the event handler code as much as you can.

Warning!: Do *not* call a Clarion MESSAGE() box in your event handling code or in the main accept loop event processor used to process your user-defined events. This can cause unpredictable results, and can hang the system.

I also determined that callbacks are the best place to post user-defined events to update the progress bar and the level graph, since these events would only be applicable during record and playback. In this case it is a much better trigger than using a window timer, because the callback is only active when it is absolutely necessary.

```

! Post the events to draw the graph and display the progress bar
! at the top of the event queue
    POST(EVENT:GraphLog, ,MSSR210_Thread,TRUE)
    POST(EVENT:ProgressBar, ,MSSR210_Thread,TRUE)
END

```

END

Handling the User-defined Event

In the Take Event embed for the main window I placed the following code:

```
! Handle User Defined Events Posted by OCX Event Handler
CASE EVENT()
OF EVENT:NormalTermination      ! display message to user
  DISPLAY(?ErrorMessage)
  POST(EVENT:Accepted,?StopButton) ! Set the OCX to MSSR_READY
OF EVENT:GraphLog
  DO MSSR210_GraphLog
OF EVENT:ProgressBar
  DO MSSR210_Progress
OF EVENT:Playback_Record_Error
  DISPLAY(?ErrorMessage)      ! display error message to user
  POST(EVENT:Accepted,?StopButton) ! Set the OCX to MSSR_READY
OF EVENT:NewVoxState           ! Display the Vox status to user
  IF OCXStatus = MSSR_Recording
    CASE NewVoxStateFlag
    OF TRUE
      VoxState = MSSR_Vox_Pause_On
      ?VoxState{PROP:FontColor} = COLOR:Red
    ELSE
      VoxState = MSSR_Vox_Pause_Off
      ?VoxState{PROP:FontColor} = COLOR:None
    END
  END
END
END
```

Dressing Up the Screen

Once I got the OCX going, I realized that I needed something more. Yes, the "LED" style peak level meter was working great but it looked kind of drab on a gray background. Besides, most audio enabled programs provide some sort of waveform feedback to the user. Unfortunately, the MSSR210 OCX doesn't provide this type of display, although the DXVU Meter OCX does.

Once again I fired up Google and started poring over web pages, and found a freeware OCX for an oscilloscope type waveform display called [XCap](#) at www.programmersheaven.com, a clearinghouse for all kinds of OCX components.

Since this OCX was free, and it would apparently do what I needed to do, I decided to put it to the test.

There are only three methods associated with the use of XCap.


```
?XCap{'Start()'}      ! Start the display
?XCap{'Stop()'}       ! Stop the display
?XCap{"AboutBox()"}   ! Display the about box
```

I installed the downloaded file and found there was a problem with the install. The install program had an older version that included references to a debugging DLL that was not included in the install.

I contacted Alex, the author and he sent me back the corrected OCX to register using the REGSVR32.EXE program that comes with Windows. (If you install the demo source code included with the article, you get the correct version of XCap and my installation program correctly registers the OCX.)

I fired up the demo app and started a recording session. The waveform trace appeared on the screen, but it was not in the correct place. Once again I had to specifically position the XCap display with the following code in the XCap_Init routine which was placed in the Before Initializing OLE Control (%BeforeOLEInitialization) embed for the ?XCap OLE control.

```
! Initialize the XCap control - Before XCap OCX init
  DO XCap_Init
```

And once again in the Procedure Routines embed I created the following routines. Notice the HIDE and UNHIDE commands. These keep the XCap control from displaying an ugly white box in the middle of the panel when it is not active.

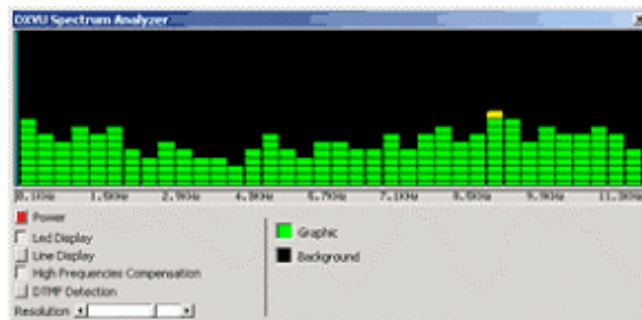
```
XCap_Init          ROUTINE
! Initialize the XCap OCX for screen display
  SETPOSITION(?XCap,16,22,136.32)
  IF NOT EXISTS('XCAP.OLR')
    ?XCAP{PROP:SaveAs} = 'XCAP.OLR\!XCAP'
  END
  EXIT
XCap_About         ROUTINE
! Display the about box for the XCap control
  ?XCap{'AboutBox'}
  EXIT
XCap_Start         ROUTINE
! Stop the XCap display
  UNHIDE(?XCap)
  XCapStart = ?XCap{'Start()'}
  EXIT
XCap_Stop         ROUTINE
! Stop the XCap display
  HIDE(?XCap)
  XCapStop = ?XCap{'Stop()'}
  EXIT
```

Using [WinSpector](#), a tool no Clarion developer should be without, I fired up the program and captured the RGB color settings for the background of the XCap control while it was active and then I populated a panel control on the window and placed the MSSR210 OCX and the XCap OCX on top of it. I then set the background color for the MSSR210 OCX to match. When I was done, I had a reasonable looking facsimile of a waveform display.

By now you should have a very good idea (and some very well tested logic) as to how to implement the MSSR210 OCX and alternatively, the XCap OCX as well. If you do not need the XCap trace display, you can simply comment out the calls to the XCap routines and the MSSR210 OCX will work just fine.

One thing that I would have liked to do is create a display based upon the data in the global `LevelQueue`.

Each entry in the queue represents the values, in decibels, received every 1/10 of a second by the callback event `NewLevelValues()`. I wanted something that would display the data in a manner similar to the image below, which can be done using the frequency analyzer display in the DXVU Meter OCX.



DXVU Meter Frequency Analyzer

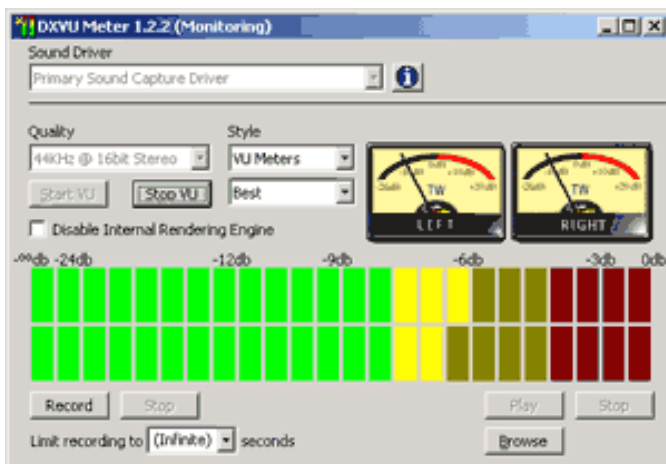
I initially thought about using Insight Graphing for this display but unfortunately, it is only really suited for the graphing of static data and not something that would stand up to constant refreshing and displaying of dynamic data.

I figured I would have to roll up my sleeves and crawl into the Clarion screen drawing commands to try to come up with some kind of solution. Unfortunately, I am not a math wizard by any stretch of the imagination and even after some pretty intense scrutiny and conversation with Tom Ruby (who is a pretty smart cookie when it comes to mathematics) via Yahoo Messenger I had to come to the realization that I am nothing more than a database programmer who is very good at logic, but lacking in the math skills I needed to pull this off.

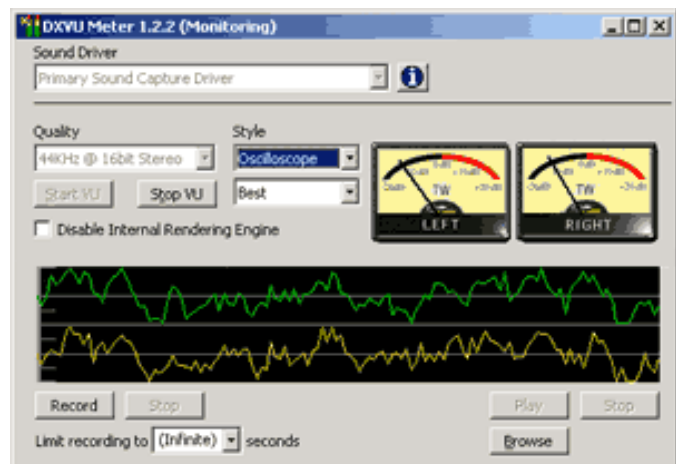
I have left the routine in the Procedure Routines embed but I took out all of the code I was trying out. I will leave it up to the reader to come up with a solution and hope that at some point I will be able to understand the offset and scaling that Tom tried to explain to me. All of the logic is in place to call the routine to draw the graph based upon data returned to the user-defined event. All that needs to be done is to put the code into the MSSR210_GraphLog routine.

If there is sufficient interest, I will try to explain in a follow up article how to convert a WAV file that you have recorded with the MSSR210 OCX into a Macromedia Flash SWF file. I have the Flash OCX documentation and I'm fixing to become neck deep in it soon.

I'll also be playing around with the DXVU Meter OCX to have another tool in my arsenal of audio tools and I hope to update you in another article about using that OCX as well.



DXVU Meter – VU Meter Display



DXVU Meter – Oscilloscope Display

I hope in some small way I have pried the lid off of Pandora's OCX box for those of you who have never used an OCX in Clarion.

[Download the source](#)

Ben E. Brady lives in what he affectionately refers to as the 'No-Tech capital of California', Dinuba, approximately 45 minutes southeast of Fresno in the heart of the Central Valley, with his wife Rita. A programmer for more than 30 years, he discovered Clarion 2.1 in 1989 after having programmed in several xBase dialects, BASIC and Z80 assembler. Ben is currently attending the College of the Sequoias

in Visalia California, pursuing a degree in Computer Science and a teaching certificate in order to pass along some of his knowledge. Ben and Rita are also very active with the Tule Fog PC Users group in Visalia California, where they provide instruction to others in need of assistance with technology and give presentations to groups of all types. Ben is also the author of several very popular [firewall reporting tools](#), written exclusively in Clarion for Windows, for users of personal firewall software such as BlackICE, ZoneAlarm, WinRoute Pro and XP Firewall.

Reader Comments

[Add a comment](#)

Copyright © 1999-2004 by [CoveComm Inc.](#) All Rights Reserved. Reproduction in any form without the express written consent of CoveComm Inc., except as described in the [subscription agreement](#), is prohibited.

[Clarion Magazine](#)

Reborn Free

CLARION
online

Bio: Andrew Guidroz II

Published 2004-07-16

Andrew Guidroz II is famous in the Clarion community for his Cajun Cookouts at the ETC conferences (and elsewhere), but he's also a skilled Clarion developer and a regular in the SoftVelocity newsgroups. He also hosts his own newsgroup, called Andrews.Kitchen, on the ClarionMag news server at news.clarionmag.com (use cmsub3 as the user id and the password).

Who do you work for? (Do you own your own company?)

For 16 years, I was the Vice President of IT in an insurance company specializing in high risk automobile insurance. On July 1, 2003 I became an independent consultant. I do everything from programming to general business consulting.

What's interesting/what do you like best about what you do now?

When I had a "real" job, there were many responsibilities with both the organization and the many employees I managed. Now, my work is more wide open. I've done work for attorneys, propane distributors, offshore oil platform builders, and political candidates. My days aren't measured by 24/7 uptime as much anymore. Every day is a totally different challenge. I sat with six candidates for the governorship of Louisiana just a few months ago, and many days I work with a nonprofit organization specializing in educational television programming. In the

last few months, I've met with an association of car dealers to discuss new methods to measure the credit worthiness of buyers, along with demonstrating to them the "Teames Technique" of margarita making.



That's a young alligator eating a cracklin. You've got to fatten them up somehow.

Describe one of your biggest challenges in using Clarion.

Challenge, to me, is finding a way around obstacles. Clarion is the opposite. It is a tool I use to work around obstacles.

Describe one of your biggest challenges in business?

So far, [business] has been like falling off a log<bg>. The toughest part of the new work is getting people to understand the wide variety of services I can provide them with. I heard someone ask one of my clients, "What exactly is it that Andrew does for you? Is he your technology guru?" He said, "He's my all around guru."

Do you use any computer languages besides Clarion? If so, why?

I use PHP for server side web programming. It is just another weapon in the arsenal. I have done work in the past in C++, but most of my Windows work now is in Clarion.

When did you start using



Pots cooking a full meal using coals. From the bottom is, chicken stew, smothered potatoes and sausage, corn maque choux, and a peach cobbler on top.

Clarion? Why?

The owners at my previous organization decided to contract an independent developer for a very large project. He began writing it in Clarion Personal Developer for DOS. For many reasons, it wouldn't scale very well and his background really was not well suited for writing in the PC market space. My job was to monitor his progress, and soon I had the unpleasant task of putting the project out of its misery.

About a week later, a local television station contacted me about writing a simple database to track their advertising rotations. I thought, "That Clarion thing might work." I wrote the whole app in the space of a couple of hours and charged a few grand for the project. The check cleared, and about three days later I figured out that there was a language underneath the IDE. I have been hooked ever since.

What's the coolest project you've worked on using Clarion?

I wrote some touch tone aware apps that did everything from sell insurance to reporting drug testing results. I always recorded sample scripts for their apps with the understanding that they would hire a professional voice to rerecord them later. Years later, I was contacted to do voice work by one of the companies <g>. And at ETC, one of the guys on one of the projects claims they still use my voice in the original app.

I also wrote software that computes live election return information and

writes all those cool graphics onto your television screen that you see on election night.

Have you done anything for a living other than software development? What was it?

I grew up on a 2,000 acre grain farm – soybeans, corn, and rice. I sold PCs for a while in college. Back then, I was contacted by a family in Arkansas with the last name of Walton who wanted to open a chain of computer stores. Luckily for both of us, they concentrated on their core business, and Wal-Mart survived.

Much of the work at my previous job was in areas outside of programming – CFO type areas. I've also done political consulting for state wide and national elections. I even serve in an advisory capacity for Clarion Magazine.

What are your hobbies/what do you like to do when you're not using Clarion?

I enjoy reading, cooking, and eating. My newest hobby is digital photography. From time to time, I've cooked for large groups of Clarion programmers.

Married, children, grandchildren, other close family you want to mention?

I've been married to Sabrina for just over ten years. We have five children ranging from seven to 23 years old: Lauren, Joseph, Ian, Austin, and Andrew. And we even have a grandbaby who is three, Spencer.

When Sabrina and I married, I don't think I realized how important and deep-running my Cajun roots were. Sabrina believes my culture and heritage is important, and has participated in sharing it from day one. She's also made sure our children understand that they are Cajun, and what exactly that means to them. The people who are closest to us and who we love the most help shine a mirror as to who we are. My relationship with Sabrina has driven me to work harder, to be a better person, a better father, a better husband, and a better representative of the entire Cajun culture.



Where were you born?

I was born in Palmetto, Louisiana, which is a town of about 200 people.

Those are the two youngest, Austin and Andrew. They are standing on the porch of an old Acadian home in Grand Coteau that is nearly 200 years old.

Where do you live now?

I live in Opelousas, Louisiana which is only about 20 miles away from where I was born.

What do you like or think is interesting about where you live?

What isn't interesting? It's the heart of Cajun country. Cajuns are

descendents of a group of French settlers who were exiled from Canada about 250 years ago. We bring a weird mixture of independence, music, and food and share it with the world.

Opelousas is also the birthplace of Zydeco music, a sort of blues/Cajun mix and the inventor of blackened redfish.

Have you lived any other interesting places?

I've always lived close to this area. Grand Coteau, where I lived for five years, is also a very interesting tiny town. I think there are more priests and nuns per capita there than any other town outside of Vatican City. My children could walk to church and school there. We also could walk in the local cemetery and see markers for family members dating back over 250 years.



Rayne, Louisiana is the Frog Capital of the World. The frogs there grow so large that the city made a lot of money exporting the frogs worldwide including France.

Which person, from past or present, do you most admire and why?

How about two people?

First, I admire Leonardo da Vinci for his engineering, art, writing, and massive array of talents.

Second, any Cajun who goes out into the world and succeeds while teaching and getting people to appreciate his/her culture. That includes folks like Dudley Leblanc, Paul Prudhomme, Clifton Chenier, George Rodrigue, and Bobby Dupre.



Tabasco Sauce is made in Louisiana. These white oak barrels are used to age the sauce for years, just like fine wine.

What is your favorite food?

Anything from a hog! I really like all food. I'm a 500 lb man trapped in a skinny boy's body.

What is your favorite drink?

Newcastle Ale, a fine English drink. Like food, I like all drinks.

What is your favorite type of music?

I like blues, rock, alternative, country, Cajun, Zydeco ... I guess I'm just a glutton for life.

What is your favorite book? Movie?

I try to read a few books every week from every topic imaginable. "The Once and Future King" by T.H. White stands out as a favorite as well as "American Psycho" by Bret Easton Ellis. I've read everything Stephen King and William Gibson have ever written. I'm reading all of Neal Stephenson's works and am currently on Cryptonomicon.

I enjoy movies too, although I tend to watch them on DVD with the surround sound rather than going to theatres. I like all the superhero movies. "Citizen Kane" stands out in my mind. The Monty Python movies are all excellent, with the "Holy Grail" as my favorite.



This is a pirogue, which is a flat bottom canoe.

If Clarion never existed, what do you think you would be doing at this time?

My work with others in the Clarion community certainly gave me more confidence to go out and try new things. A lot of my cooking and business dealings are directly related to that new found confidence. I don't think the tool changed my life directly as much as it opened me up to a new community of thinkers. I probably wouldn't be as conscious of the rest of the world without discovering it.

Is there anything else you want to mention?

Just that everyone should spend more time learning about Cajun culture

and how important communities of immigrants have been to this country.



We use sugarcane syrup on our pancakes in Louisiana and everyone grew up with a little can of it in the pantry. These cans are larger replicas (read the size on the front).



Reader Comments

[Add a comment](#)

Hi Andrew, finally some recognition as a "guru"!! I...

Copyright © 1999-2004 by [CoveComm Inc.](#) All Rights Reserved. Reproduction in any form without the express written consent of CoveComm Inc., except as described in the [subscription agreement](#), is prohibited.

Reborn Free

CLARION
online

A Function to Return Clarion Standard Time

by **Steven Parker**

Published 2004-07-16

Mark Riffey made the request: "I want to do `TIME('12', '24', '06')` and get a Clarion [standard] time just like I would do `DATE('12', '24', '2004')` and get a Clarion date."

A function to construct a Clarion standard time nicely complements previous articles on time functions. In [Marking Time, Part 1](#), functions are presented to extract the number of hours, minutes and seconds between two times (similar to the `Day()`, `Month()` and `Year()` functions for dates), as well as a function to return an elapsed time (time math). [Marking Time: Round 2](#) shows how to set wait periods, timeouts and a count down timer.

Creating Mark's function, like the functions in the Marking Time articles, requires only understanding Clarion standard times. The online help states:

A Clarion standard time is the number of hundredths of a second that have elapsed since midnight, plus one (1). The valid range is from 1 (defined as midnight) to 8,640,000 (defined as 11:59:59.99 PM). A standard time of one is exactly equal to midnight to allow a zero value to be used to detect no time entered. Although time is expressed to the nearest hundredth of a second, the system clock is only updated 18.2 times a second (approximately every 5.5

hundredths of a second).

The key facts are that midnight is 1 and that a Clarion clock operates in one hundredths of a second.

Because the numbers involved can be easily mistyped, it is a good idea to create Equates and embed them in the Global area:

```
eSecond    EQUATE(100)      ! one second
eMinute    EQUATE(6000)    ! 60 * 100
eHour      EQUATE(360000)  ! 60 seconds * 60 minutes
eDay       EQUATE(8640000) ! 11:59:59.99 pm
```

Now I can create the required function. Here's the data declaration:

```
RetVal Long
```

And here's the code:

```
RetVal = pSeconds * eSecond
RetVal = RetVal + (pMinutes * eMinute)
RetVal = RetVal + (pHours * eHour)
Return RetVal
```

Carlos Guitierrez immediately pointed out that I forgot to add back the 1 (for "midnight" per the documentation quoted above).

His solution is:

```
If pSeconds or pMinutes or pHours
  Return (pHour * ehour + pMinutes * eMinute + |
    pSeconds * eSecond + 1)
Else
  Return (0)
End
```

And his solution is also more elegant than mine.

However, the parameters in Carlos's code are not optional (though you can make them optional). This means that Carlos' "If" check will always succeed. So, I made a final modification:

```
RetVal = pHours * eHour + pMinutes * eMinute + pSeconds * eSecond
```

```
If RetVal  
    Return (RetVal + 1)  
Else  
    Return (0)  
End
```

Whichever flavor of the function you use, you can now take a string, parse it using string slicing (or SUB). You can extract values corresponding to seconds, minutes and hours and get a Clarion standard time. Then, you can use the return value to do time math.

For the time being, you would implement this function by creating a Source procedure and copying/pasting. For more general usage, I would add it to the functions in the Marking Time articles and (probably) make a class out of all the functions.

[Steve Parker](#) started his professional life as a Philosopher but now tries to imitate a Clarion developer. He has been attempting to subdue Clarion since 2007 (DOS, that is). He reports that, so far, Clarion is winning. Steve has been writing about Clarion since 1993.

Reader Comments

[Add a comment](#)

Below is a version of the HMSH2Time() function that works...

**Hi Steven Thanks for mentioning my name in your...
forgot to add: But I like your solution better, it's...
Your solution was much more elegant than mine. I...**

Thanks Steve for the article and also thanks Carl for the...

Copyright © 1999-2004 by [CoveComm Inc.](#) All Rights Reserved. Reproduction in any form without the express written consent of CoveComm Inc., except as described in the [subscription agreement](#), is prohibited.

[Clarion Magazine](#)



Generating A Unique Registration Code

by **Vince Du Beau**

Published 2004-07-26

While browsing some of the Mac developer's websites, I came across a scheme for generating unique (hopefully) registration codes. I thought it might be a good Clarion project.

Registration schemes can take many forms. Some generate a seemingly random series of numbers. Others add in parts of the product name and/or the user name. Of course, the end user's application can employ various validation schemes depending on the method used to generate the registration code.

The registration code generator

This will be a simple scheme based on the user name. The code that it generates will look like this:

CGL100-1234-5678

The first character is the license type, S for single, G for group, and C for corporate. The next two characters are the application code followed by a three digit version number. The numbers 1234 and 5678 will be the actual generated registration number.

The code generator window should look like Figure 1.

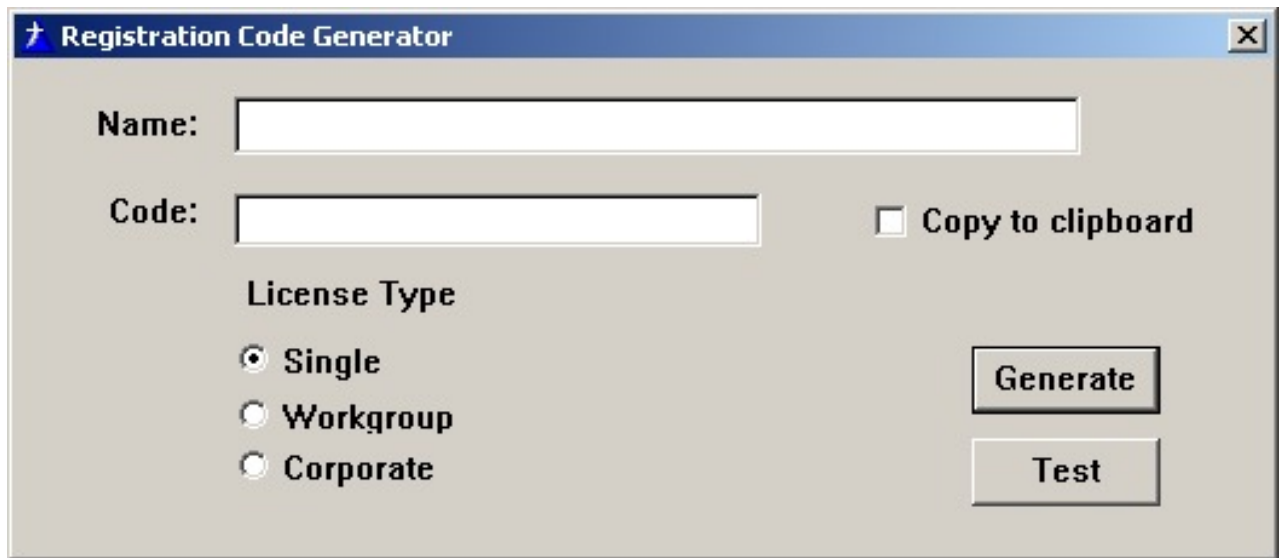


Figure 1. Registration Code Generator

Create the following global variables in your application. They will be used by the various procedures.

```
GLO:License           STRING('S')
GLO:NameGenerate     STRING(50)
GLO:CodeGenerate     STRING(16)
GLO:NameValidate     STRING(50)
GLO:CodeValidate     STRING(20)
GLO:CopyToClipboard  BYTE
```

Setting the GLO:License global to 'S' will set the default for the option box.

Set the USE and VALUE fields according to the following window structure.

```
Window WINDOW('Registration Code Generator')
    , AT( , , 260, 102), IMM, SYSTEM, GRAY
    ENTRY(@s50), AT(46, 8, 174, 11), USE(GLO:NameGenerate)
    STRING('Code:'), AT(11, 27, 27, 11), USE(?String2), RIGHT
    ENTRY(@s16), AT(46, 28, 108, 10), USE(GLO:CodeGenerate)
    CHECK('Copy to clipboard'), AT(178, 29, 72, 9), USE(GLO:CopyToClipboard)
    STRING('License Type'), AT(48, 44, 53, 10), USE(?String3)
    OPTION('Option 1'), AT(42, 53, 60, 43), USE(GLO:License)
        RADIO('Single'), AT(46, 58, 41, 8), USE(?Option1:Radio1), VALUE('S')
        RADIO('Workgroup'), AT(46, 70, 53, 7), USE(?Option1:Radio2), VALUE('G')
        RADIO('Corporate'), AT(46, 80, 76, 8), USE(?Option1:Radio3), VALUE('C')
    END
    BUTTON('Test'), AT(198, 78, 39, 14), USE(?Button1:2)
    BUTTON('Generate'), AT(198, 59, 39, 14), USE(?Button1), DEFAULT
    STRING('Name:'), AT(10, 9, 28, 10), USE(?String1), RIGHT
END
```

In the `Init` procedure for the window, put this code snippet:

```
GLO:NameGenerate = CLIPBOARD(1)
```

This will automatically copy any text in the clipboard to the `GLO:NameGenerate` global when the application starts. This is handy if you wanted to copy the name from an email registration. Just be sure not to copy the text of War and Peace into the clipboard before running the application! The '1' in the function equates to the `CF_TEXT` constant in the Windows API. If the content of the clipboard is not text, null will be returned.

In the Action Tab for the "Test" button, set the button to call a procedure. The procedure should be named `TestWindow`.

Do the same for the Generate button naming the procedure to be called `GenerateSerial`.

After setting up the main window, double-click on the `GenerateSerial` procedure in the IDE and select source as the type.

Create a string variable for the procedure called `SerialNumber` and give it a length of 16. Next create two longs named `Num1` and `Num2`.

Type the following code in the editor.

```
SerialNumber = 'SGL100-abcd-efgh'           ! Default values
SerialNumber[1] = GLO:License
IF LEN(CLIP(GLO:NameGenerate)) < 10 THEN
  Num1 = MESSAGE(CLIP(GLO:NameGenerate) |
    & ' must be at least 10 | characters', 'Invalid |
  Name', ICON:Exclamation)
ELSE
! Generate value for 'abcd'
  Num1= ((VAL(GLO:NameGenerate[10])+ |
  VAL(GLO:NameGenerate[8])*2)*VAL(GLO:NameGenerate[5])+3)* |
  VAL(GLO:NameGenerate[2])
  Num1= ((Num1 % 10000) + 3749) % 10000
  SerialNumber[8:11] = FORMAT(Num1, @P####P)
! Generate value for 'efgh'
  Num2 = ((VAL(GLO:NameGenerate[9]) + VAL(GLO:NameGenerate[7]))* 2* |
  VAL(GLO:NameGenerate[6]) + 3) * VAL(GLO:NameGenerate[3])
  Num2 = Num1 + Num2
```

```
Num2 = (( Num2 % 10000) + 7564) % 10000
SerialNumber[13:16] = FORMAT(Num2, @P####P)
GLO:CodeGenerate = SerialNumber
IF GLO:CopyToClipboard THEN SETCLIPBOARD(GLO:CodeGenerate) .
END
```

While it's not necessary to assign a value to `SerialNumber`, it helps to visualize where the different values go. The first value is the license type which will be replaced by the value of the option box. The next two characters represent the application code (GL = General Ledger?). Characters four through six represent the serial number. The values 'abcd' and 'efgh' will be replaced by the registration code.

Since string slicing does no bounds checking, a check needs to be included to make sure there are at least ten characters in the name; otherwise, for demonstration purposes only, the registration code will be generated with arbitrary values from memory.

As you can see, the code is replacing 'abcd' with the tenth, eighth, fifth, and second characters and 'efgh' with the ninth, seventh, sixth, and third characters. Some random math is performed on the ASCII value of the characters to arrive at the registration code. The two numbers 3749 and 7564 are added solely as an attempt to avoid a code that begins with a zero.

The character positions were picked with no particular logic behind them. The basic concept is to attempt to make it more difficult for a cracker to build a code generator. Feel free to use any character sequences you desire.

The last thing the code does is to copy the registration code to the Windows clipboard if the checkbox was checked.

The registration code validator

The code validation window should look like Figure 2.

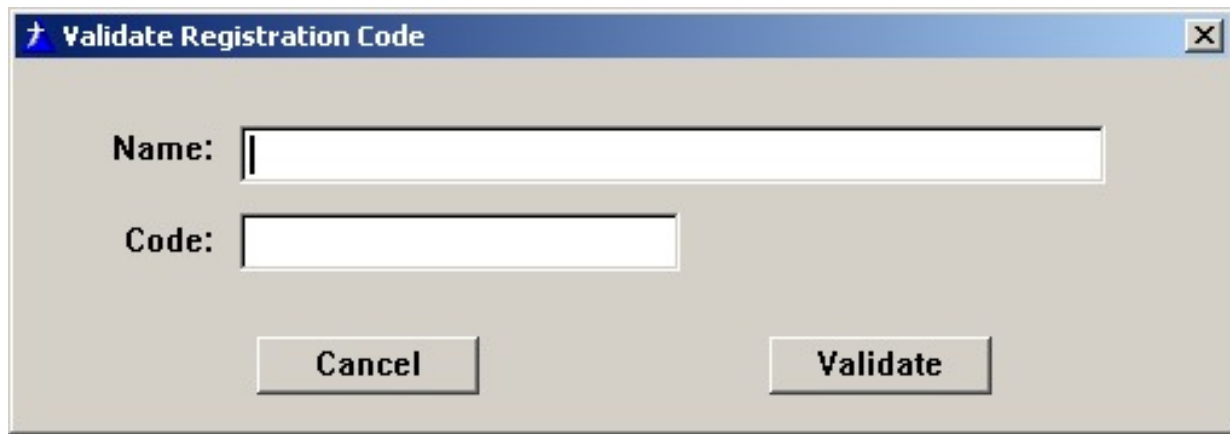


Figure 2. Validate Registration Code

Again, set the USE and VALUE fields according to the window structure.

```
Window WINDOW('Validate Registration Code'),AT(, ,251,76),IMM,SYSTEM,GRAY
  STRING('Name:'),AT(10,14,31,11),USE(?String1),RIGHT
  ENTRY(@s50),AT(47,14,178,11),USE(GLO:NameValidate)
  STRING('Code:'),AT(10,33,31,11),USE(?String2),RIGHT
  ENTRY(@s20),AT(47,32,90,11),USE(GLO:CodeValidate),UPR
  BUTTON('Validate'),AT(156,57,46,12),USE(?Button1)
  BUTTON('Cancel'),AT(50,57,46,12),USE(?Button2),STD(STD:Close)
END
```

In the Action Tab for the Validate button, set the button to call a procedure named ValidateSerial.

After creating the Validate Window, double-click on the ValidateSerial procedure and select source as the type. Create the SerialNumber, Num1, and Num2 variables for this procedure as in the GenerateSerial procedure.

Add another long named MsgReturn for the return value of the MESSAGE function.

In the editor type the following code.

```
SerialNumber = 'CGL100-abcd-efgh' ! Default values
Num1 = ((VAL(GLO:NameValidate[10]) + VAL(GLO:NameValidate[8]) * 2) * |
  VAL(GLO:NameValidate[5]) + 3) * VAL(GLO:NameValidate[2])
Num1 = ((Num1 % 10000) + 3749) % 10000
SerialNumber[8:11] = FORMAT(Num1, @P####P)
Num2 = ((VAL(GLO:NameValidate[9]) + VAL(GLO:NameValidate[7]) * 2) * |
  VAL(GLO:NameValidate[6]) + 3) * VAL(GLO:NameValidate[3])
Num2 = Num1 + Num2
Num2 = (( Num2 % 10000) + 7564) % 10000
```



```
SerialNumber[13:16] = FORMAT(Num2, @P####P)
IF GLO:CodeValidate[2:16] NOT = SerialNumber[2:16] THEN
    MsgReturn = MESSAGE('Invalid Serial Number. Try Again', |
        'Validation',ICON:Exclamation)
ELSE
    MsgReturn = MESSAGE('Valid Serial Number', 'Validation', ICON:Asterisk)
END
```

When a name and serial number are entered and the Validate button is clicked, this code will go through the same procedure used to generate the registration code. It then compares the entered code with the one it generates, and displays a message whether the code is correct or not.

Summary

This is just a simple application to demonstrate one method of generating a registration code. Naturally, the code to do the validation would have to be incorporated into your application.

More work would have to be put into the code if you were actually going to validate the license, application code, and version number. The code presented here should at least give you a good starting point in designing a registration scheme to suit your needs.

[Download the source](#)

[Vince Du Beau](#) is an independent consultant writing custom applications for OS X and Windows using REALbasic and Clarion.

Reader Comments

[Add a comment](#)

Copyright © 1999-2004 by [CoveComm Inc.](#) All Rights Reserved. Reproduction in any form without the express written consent of CoveComm Inc., except as described in the [subscription agreement](#), is prohibited.

Clarion Magazine



Reading XML With The CenterPoint Classes

by David Harms

Published 2004-07-30

A year ago (how time flies) I wrote an [article](#) on how to create an XML file with the Clarion 6 CenterPoint wrapper classes. In this article I'll show you one way to read an XML file with those same classes. But first, by way of recap, a word or two about doing XML in Clarion.

Clarion 6 takes a multi-pronged approach to XML development. The easiest way to get started is to use the new templates that let you output reports and other data as XML. If you feel you need more control, you can get your hands a bit dirtier with the XMLGenerator, an all-Clarion class that is used by the XML templates. And at the far end of the spectrum, if you want the greatest power and flexibility and you don't mind the extra effort, you can use COM and the MS XML parsers.

The Clarion CenterPoint classes lie somewhere between the XMLGenerator class and the COM option, at least in terms of power and flexibility. These provide a wrapper around the open source [CenterPoint C++ XML class library](#), which includes SAX and DOM parsers, and which ships with C6. As Clarion's documentation on the CP classes is a bit sketchy, you may want to [download](#) the CenterPoint C++ examples, or browse the [CenterPoint discussion forums](#).

Most Clarion developers who read/write XML use one of the following technologies, in descending order of popularity (according to a recent ClarionMag poll):

- custom XML code
- COM and the Microsoft parsers
- Clarion XML classes
- C6 XML templates
- CenterPoint wrapper classes.

Why are the CP wrapper classes at the bottom of this list? In part, it's because of the lack of documentation. I believe it's also because these classes occupy a niche that's getting crowded from both ends. The all-Clarion XML code is getting better and better, and at the same time it's becoming easier to use COM and therefore the MS XML parsers. So why

bother with the CP classes at all? If you're comfortable with COM, there probably isn't a compelling reason. But if you're not ready to move to COM, and you need more power than the all-Clarion solution, then they are worth a look.

As with my previous article, I'm going to look at the CP DOM parser, not the SAX parser. Increasingly, DOM is the tool of choice for XML developers. As I explained in [XML for Clarion Developers](#), a SAX parser processes each line (or more accurately, each element) as it is read, much like Clarion processes ASCII files. As each element is read, you, the programmer, decide what to do with that element by means of callback functions. DOM parsers, on the other hand, read the complete XML file before presenting you with a tree-structured document based on the file.

Although DOM parsers are more memory-intensive, than SAX parsers, memory isn't as big a problem as it once was, and for many if not most XML tasks DOM parsers get the nod these days.

Writing XML

To use the CP classes in your app, you'll need the following global include:

```
include('cpxml.inc'),once
```

That ensures that you have not only the CenterPoint wrapper classes available, but also some library functions that are used to create and manage XML document instances.

There isn't all that much code to this procedure. The data declarations are as follows:

```
pDoc          &Document
pRootElement  &Element

types         GROUP
              STRING( ' ELEMENT                ' )
              STRING( ' ATTRIBUTE              ' )
              STRING( ' TEXT                    ' )
              STRING( ' CDATA_SECTION           ' )
              STRING( ' ENTITY_REFERENCE        ' )
              STRING( ' ENTITY                  ' )
              STRING( ' PROCESSING_INSTRUCTION ' )
              STRING( ' COMMENT                 ' )
              STRING( ' DOCUMENT                ' )
              STRING( ' DOCUMENT_TYPE           ' )
              STRING( ' DOCUMENT_FRAGMENT      ' )
              STRING( ' NOTATION                 ' )
              END
typename      STRING( 25 ), DIM( 12 ), OVER( types )
```

`pDoc` is a reference to an instance of the XML Document class, and `pRootElement` is a reference to an instance of the XML Element class. The combination of the types GROUP and the typenames STRING, OVER the types declaration, creates a preloaded array of string values. These will be used a little later to decode node types.

The code to read the XML document is fairly straightforward, for the most part:

```
pdoc &= XMLFileToDOM('xmlgen.xml')
IF (pdoc &= null)
    MESSAGE('Unable to process the input XML document')
ELSE
    pRootElement &= pDoc.GetDocumentElement()
    IF MESSAGE('root element ' & pRootElement.getTagname(), 'Reading with DOM', |
        icon:exclamation,button:abort+button:ok,button:ok) <> button:abort
        Traverse(pRootElement,1)
    END
END
END
```

The `XMLFileToDOM` function handles all the tasks associated with loading and parsing the XML document, and returns an instance of the `Document` class. Since this is a DOM document, that means that all of the information contained in the XML file is now contained in memory, in a tree structure.

The error checking here is rudimentary; you'd probably want to code something a little more robust in a production environment.

The code displays the name of the root node, obtained with the `GetDocumentElement` method, and then calls the recursive `Traverse` function. `Traverse` is a local function, and is prototyped this way in the procedure's Data Section embed:

```
MAP
    Traverse(Node n,long level),BYTE,PROC
END
```

The procedure is implemented in the Procedure Routines embed, as follows:

```
nl      &NodeList
i      LONG
CODE
IF MESSAGE('type ' & typename[n.getNodeType()] |
    & '|name ' & n.getNodeName() |
    & '|value ' & n.getNodeValue() |
    & '|Level ' & level |
    , 'Reading with DOM', |
    icon:exclamation,button:abort+button:ok,button:ok) = BUTTON:Abort THEN RETURN FALSE.
nl &= n.GetChildNodes()
ASSERT(NOT nl &= NULL)
LOOP i = 0 TO nl.getLength() - 1
    level += 1
    IF NOT Traverse(nl.item(i),level) then return false.
    level -= 1
END
nl.release()
RETURN TRUE
```

`Traverse`'s job is twofold. First, it reports on the type, name, and value of each of the node it received (for test purposes, it also tracks a level number). `Traverse` also get a list of child nodes for the passed node, and if there are any child nodes it calls itself for each of

those nodes. In this way `Traverse` walks the entire document tree.

It can be tedious examining an entire document to find just one node, and for this reason several standards have evolved for querying XML documents. XPath is supported by CenterPoint, although I haven't attempted to use it, and I can't guarantee it's usable through the CP wrapper classes. I leave this as an exercise for the reader. XQuery is, as far as I know, not available with the CenterPoint library.

Summary

The Clarion 6 CenterPoint XML wrapper classes give Clarion developers access to a robust open source XML implementation. Although not as fully functional as, say, the COM-based MS parsers, this library does fill a mid-level niche, and is worth a closer look if you're not ready to implement a COM solution, and not satisfied with the all-Clarion XML code in C6.

[Download the source \(C6 only\)](#)

*[David Harms](#) is an independent software developer and the editor and publisher of Clarion Magazine. He is also co-author with Ross Santos of *Developing Clarion for Windows Applications*, published by SAMS (1995). His most recent book is [JSP, Servlets, and MySQL](#), published by HungryMinds Inc. (2001).*

Reader Comments

[Add a comment](#)



Using Virtual PC 2004

by **Michael Lawson**

Published 2004-07-30

If you're like me you usually have a couple of older PCs lying around to either help test compatibility, or to keep older projects and development environments on. After a while this gets to be a burden, first in keeping track of them, and second in keeping them running, not to mention the space each computer takes up. When I went to a recent Microsoft event the presenter demonstrated Virtual PC 2004, which MS had just started shipping. Virtual PC comes as part of MSDN and has been added to the more affordable Action Pack bundle. Links to both these programs are available from the [Microsoft Partner Site](#). You can also download a free [45 day trial](#) from Microsoft.

Virtual PC enables your computer to run more than one operating system at once. The original operating system on your computer is the host, and all the others are "guest" operating systems. Virtual PC differs from boot manager products in that each OS can work concurrently in its own window. If you have ever used a Terminal Service Client, Virtual PC will look familiar to you. Not only can you load older operating systems, but you can also use a newer operating system, i.e. the host could be Windows 2000, with a Virtual PC running Windows XP. Virtual PC 2004 currently supports guest operating systems from DOS 6.22 on up. The host operating system must be Windows XP Professional, Windows 2000 Professional or Windows XP Tablet PC Edition.

I have used this multi-OS capability to test the latest Clarion 6.x development environments on my applications. I no longer have to worry about removing patches, restoring applications or corrupting various registry entries. I load a new Virtual PC with the XP operating system, and I load the base Clarion 6.0 IDE. I then apply whatever patch for the current release candidate or EA version. Third party templates are applied last after I first check that the supplied examples compile and run.

I have also used this methodology to verify the builds of demonstration software that

I distribute. This helps test such things as built-in time bombs and security. You could also build configurations that would match a customer's default system image, without requiring a separate PC. It is also useful to test Windows, SQL Server and Office patches and the effect that they may have on your applications.

In order to get started it is a good idea to get either an original ISO image (which is a CD image) of the guest operating system, or an install CD and bootable floppy with drivers to support your CD drive. Your PC should also have adequate RAM and hard disk space available to support the Virtual PC. Theoretically you could have as many Virtual PCs running as you have physical memory and hard disk space to support. The major bottleneck in response would depend on processor utilization and disk swapping. Virtual PC supports multiple processors and many options are available for optimizing the use of multiple Virtual PCs at once.

Once the Virtual PC application is loaded on your computer it is just a matter of running the Virtual PC Console.

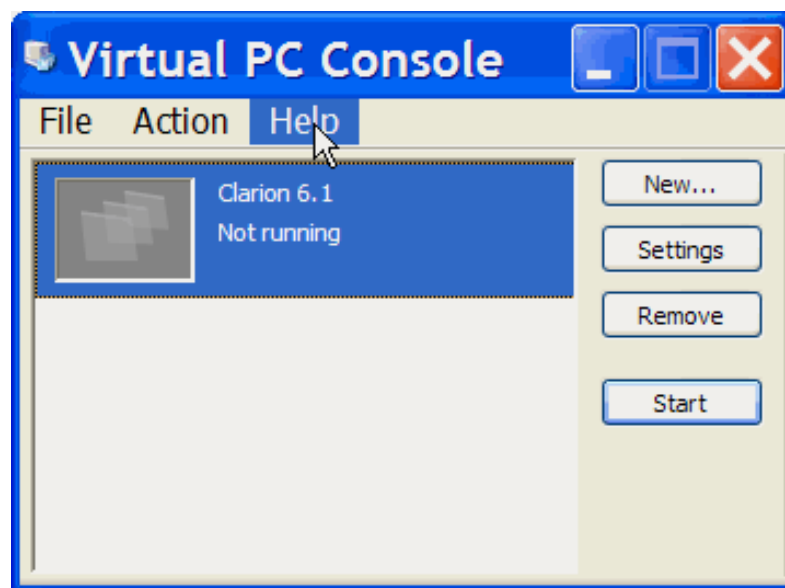


Figure 1

In the Virtual PC Console pictured above I currently have one Virtual PC installed and it is not running. The menu options to the right show the most common functions that are used by the console. In order to create a new Virtual PC I would select the New option which would then launch a wizard to create a new Virtual PC. I will now go through this task and explain what is actually being done at each step in the process.

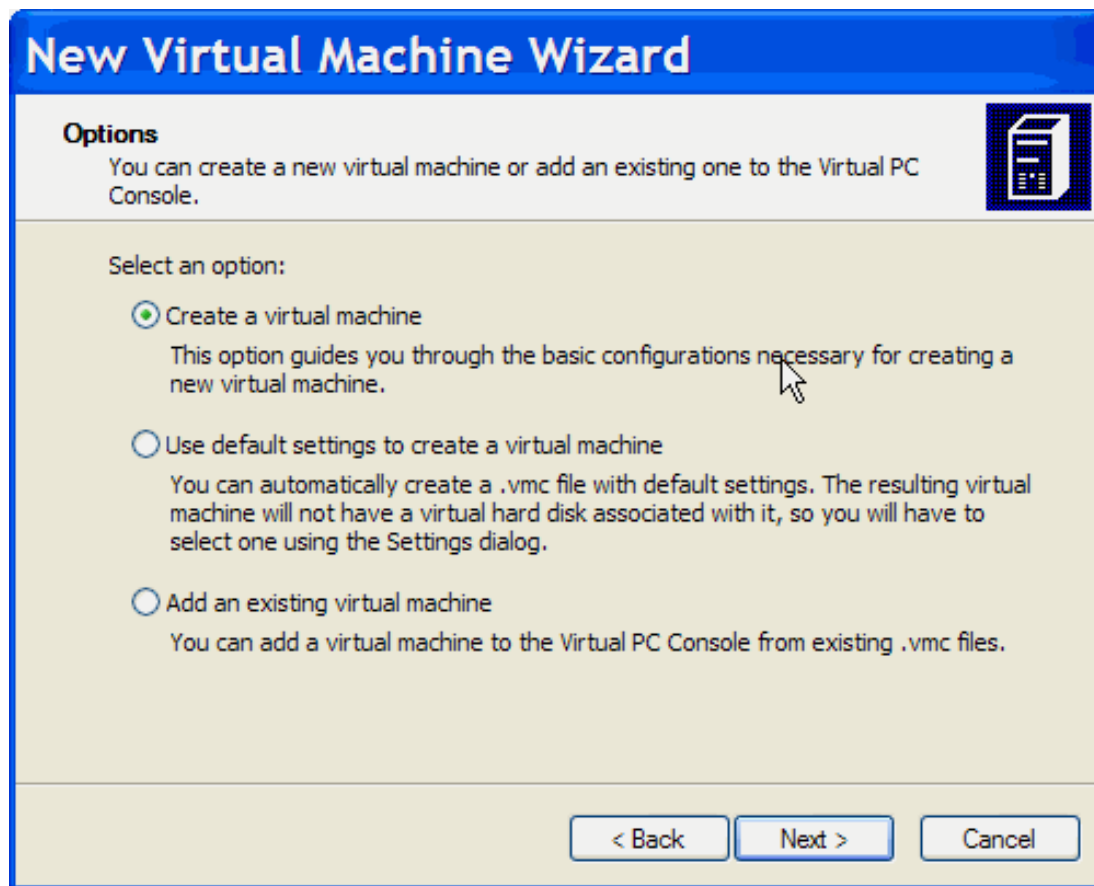


Figure 2

In Figure 2 you have three options to choose from; if you want to get used to using Virtual PC the easiest is to have the wizard create a virtual machine. This option allows you the least amount of flexibility and has the fewest prompts. The second and third options are useful if you already have existing Virtual PCs configured.

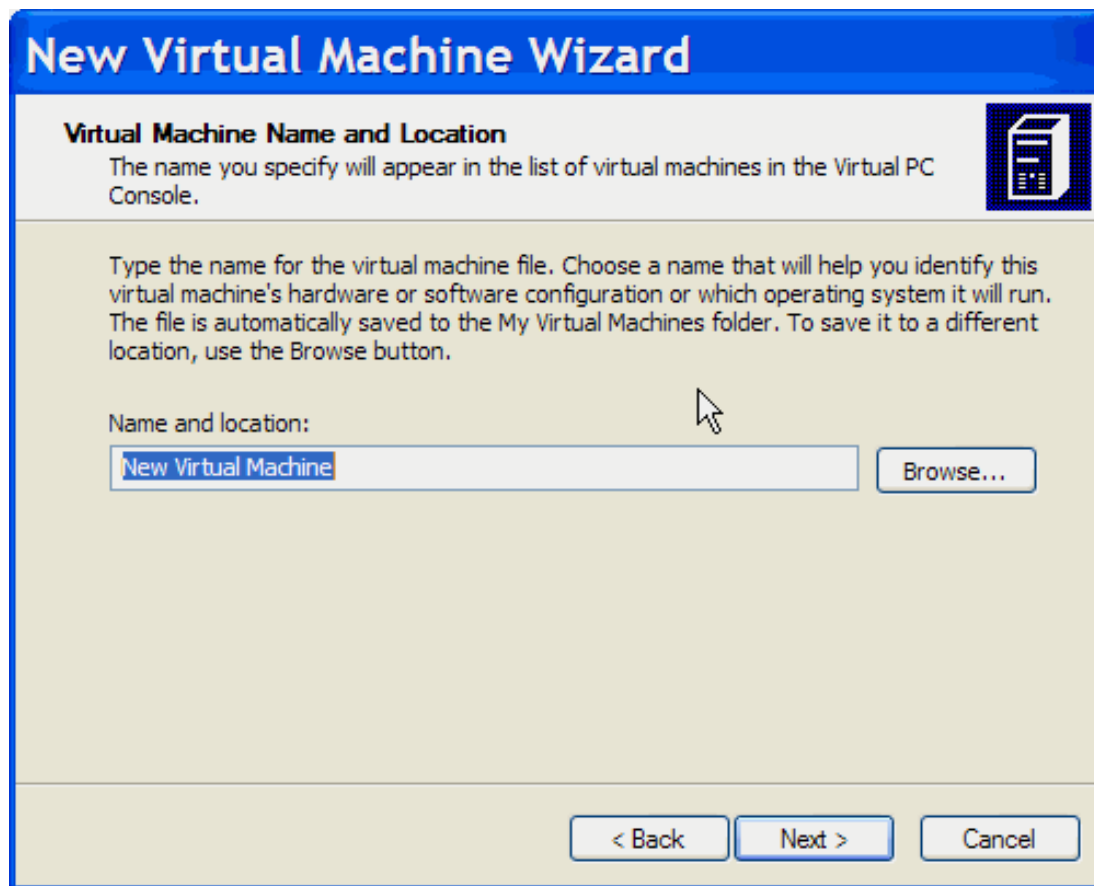


Figure 3

The next screen in the wizard allows you to specify a name for your Virtual PC. This name will correspond to a file ending in a .vmc extension. This is the virtual machine configuration file which will contain all of the information about a specific Virtual PC. A hard disk file is also created later in the process which will actually contain the operating system and applications just like a regular PC.

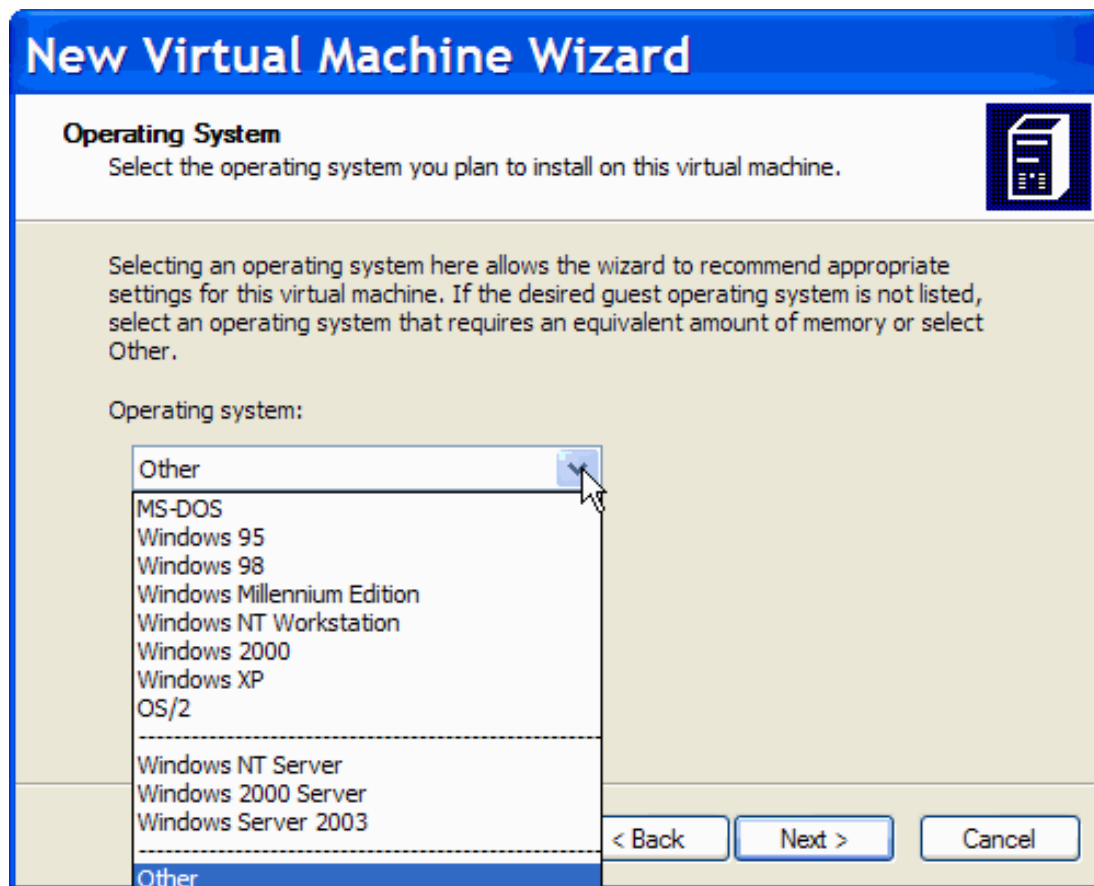


Figure 4

As you can see in Figure 4 above Virtual PC directly supports most Microsoft operating systems including the infamous Millennium Edition. While I haven't tried it, supposedly certain versions of Linux may also run with Virtual PC (I would check on-line in the Microsoft newsgroups first to see which versions others have successfully loaded). Microsoft confirms that Windows 3.x will also run as a Virtual PC.

It is important to note that the newer operating systems images which ship with MSDN and Action Pak subscriptions are easier to install than the ones that come with your PC or as upgrades. In order to load older operating systems with non-bootable CDs, you will need a boot floppy with CD drivers, or you will have to create an ISO image from the original media, and stored it on the host PC. A great freeware product for recording ISO images from CDs is available but it only works with XP; this utility plugs into Explorer and can be downloaded from [Alex Feinman's web site](#).

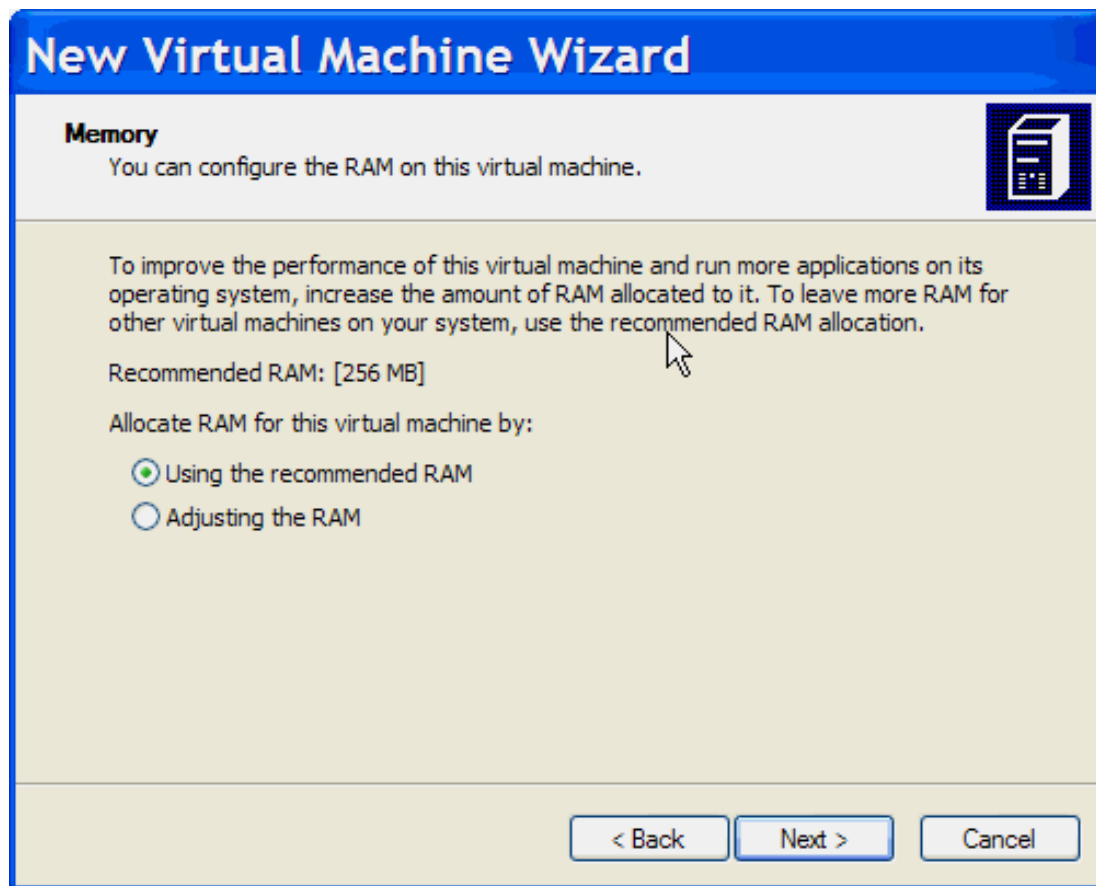


Figure 5

Unless you are going to use this Virtual PC as an alternative configuration, a testing platform, or in production, I suggest you use the default settings. The operating system will automatically size and manage its own paging file based on the Virtual PCs RAM size. This can be changed at any time so it really is a good idea just to take the recommended defaults. Most modern Microsoft operating systems have a limit of 4GB of physical memory, and this selection will take the selected amount of RAM from your host OS when the Virtual PC is running.

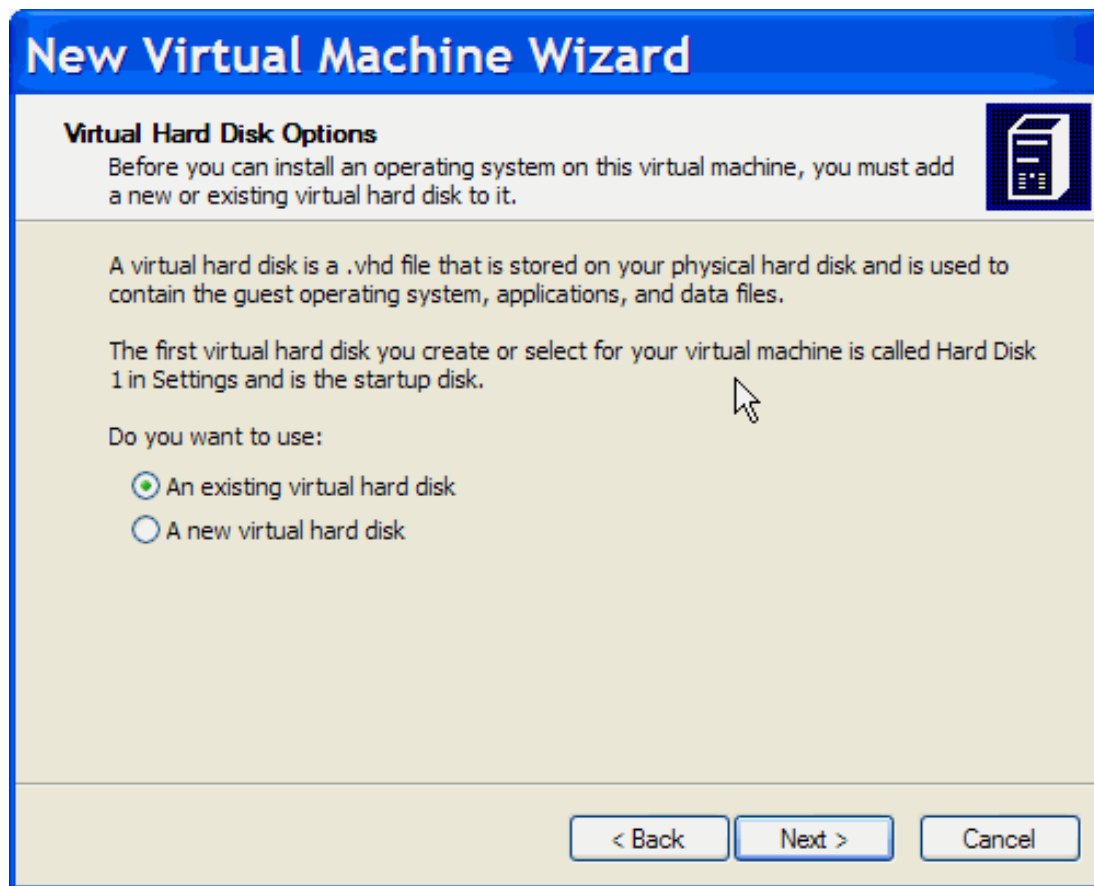


Figure 6

Along with the .vmc file mentioned above a .vhd file is needed in order to store the guest operating system and applications. What is good about this configuration is that you only need to deal with the two associated .vmc and .vhd files in order to move the Virtual PC, or to make a backup. While you could use an existing .vhd when creating a new Virtual PC, I would recommend creating a new one. The hard drive space is by default allocated dynamically and is normally defaulted to 16GB on Windows 2000 and XP.

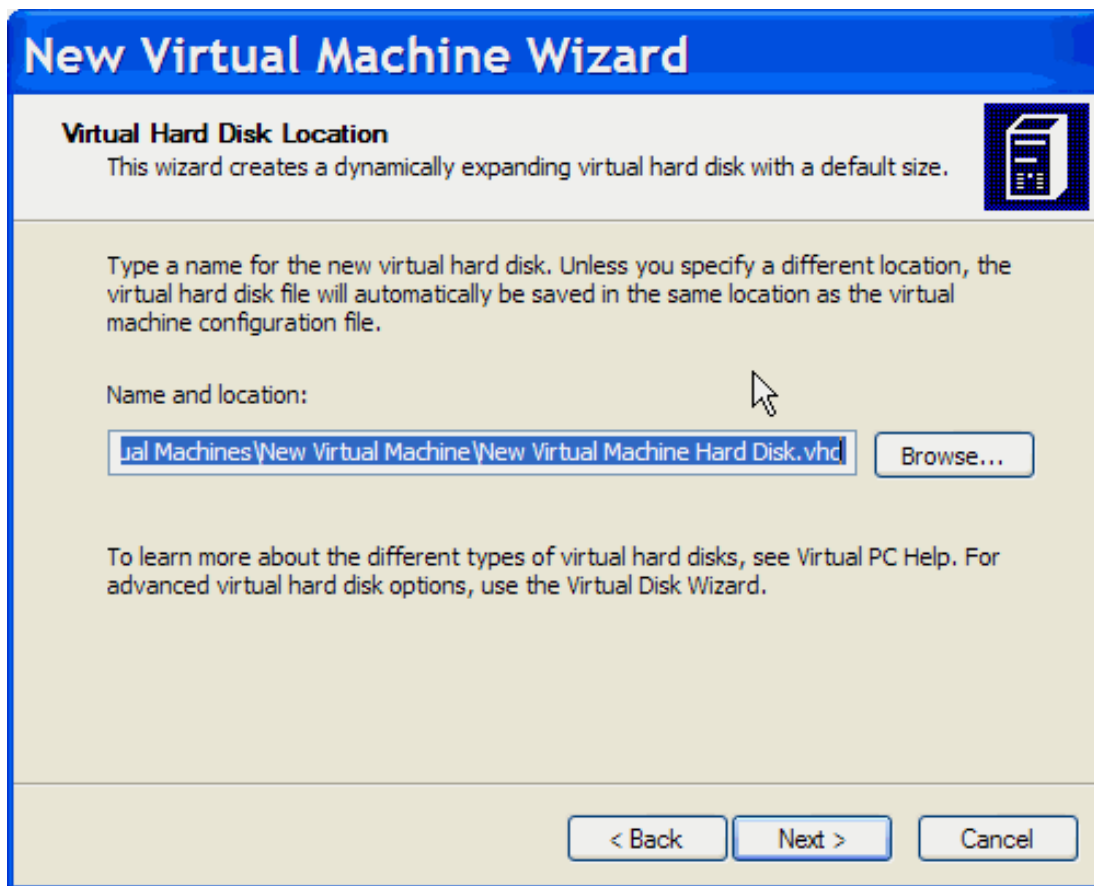


Figure 7

You may also select the location of the .vhd file, in normal circumstances this should be located in the same subdirectory as the .vmc file.

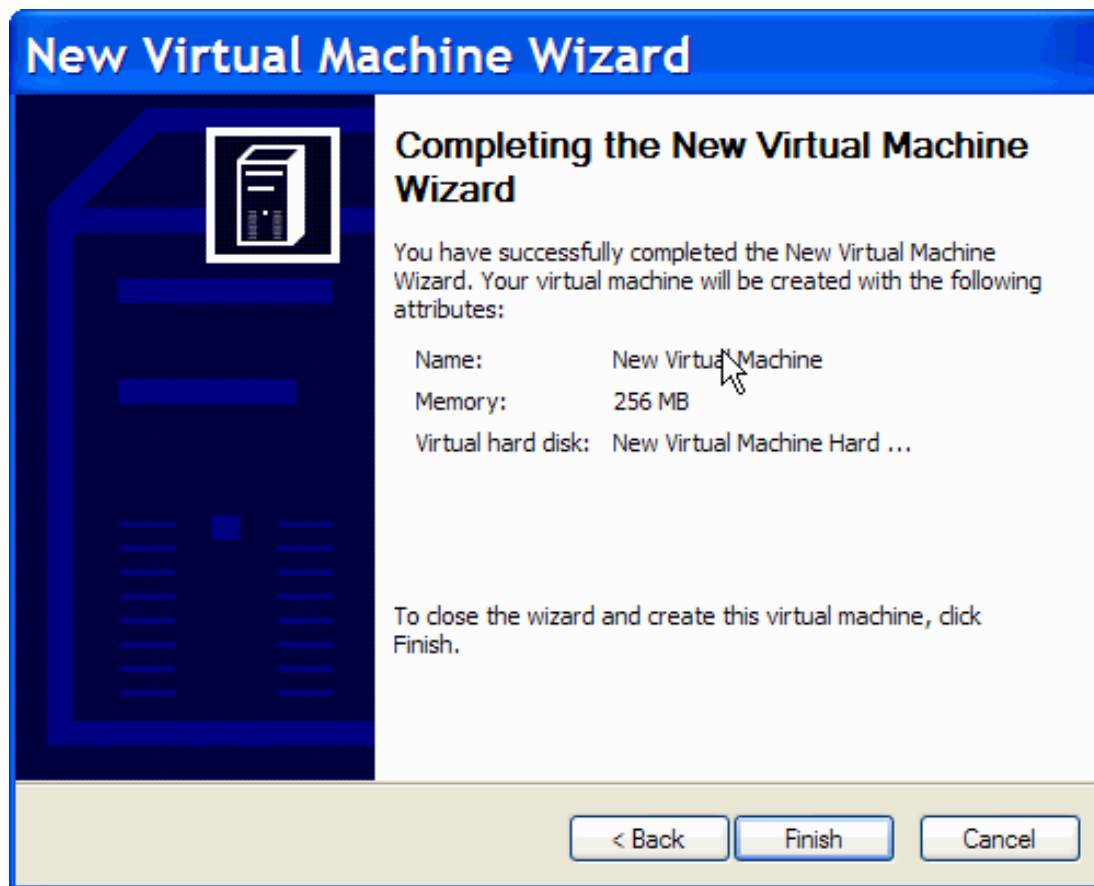


Figure 8

A confirmation page appears that you have now created a Virtual PC which now will appear in the Virtual PC Console. The confirmation pages displays the name of the virtual machine, memory allocated and the virtual hard disk file name.

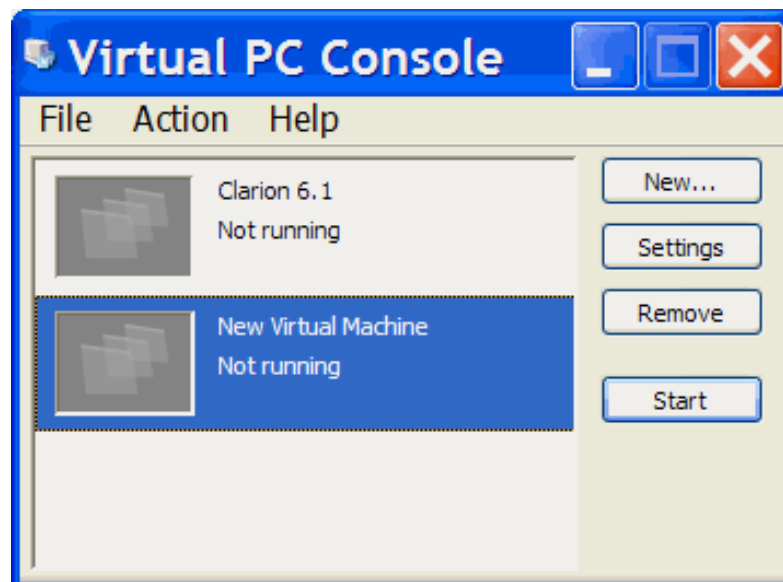


Figure 9

Now that you have created the Virtual PC you will need to load an operating system. This process is similar to loading the OS on a regular PC. Click on the Start button to

boot the Virtual PC, and it will display the amount of RAM, and hard drive information. It will then prompt you for a boot device or an operating system to be installed.



Figure 10

If a bootable CD is inserted in the host PC's drive the normal installation process should begin for the selected OS. Otherwise if you load an OS DVD from the MSDN library you will be presented with a selection of the operating systems contained on that DVD as shown in Figure 11 below. If you are using a DOS boot disk make sure it is in the drive when you start the Virtual Machine and that it has the CD drivers installed correctly. The ISO image option is a little more complicated, but step by step directions are available from the Help File.



Figure 11

If the boot screen still keeps asking for a boot image or boot device select CD|Use Physical Drive X: from the Virtual PC's menu. A few caveats to note is that installation of an operating system may take longer than on a real machine, especially when installing XP. XP actually has post installation additions that make using it more efficient. I would also turn off any virus scanning software while installing an OS on a Virtual PC. Advanced adjustments are available to speed things up for different operating systems and operating system loads; please consult the Virtual PC's documentation for more information. A few issues also are documented in the included documentation that note conflict resolution issues between running specific Virtual PCs together.

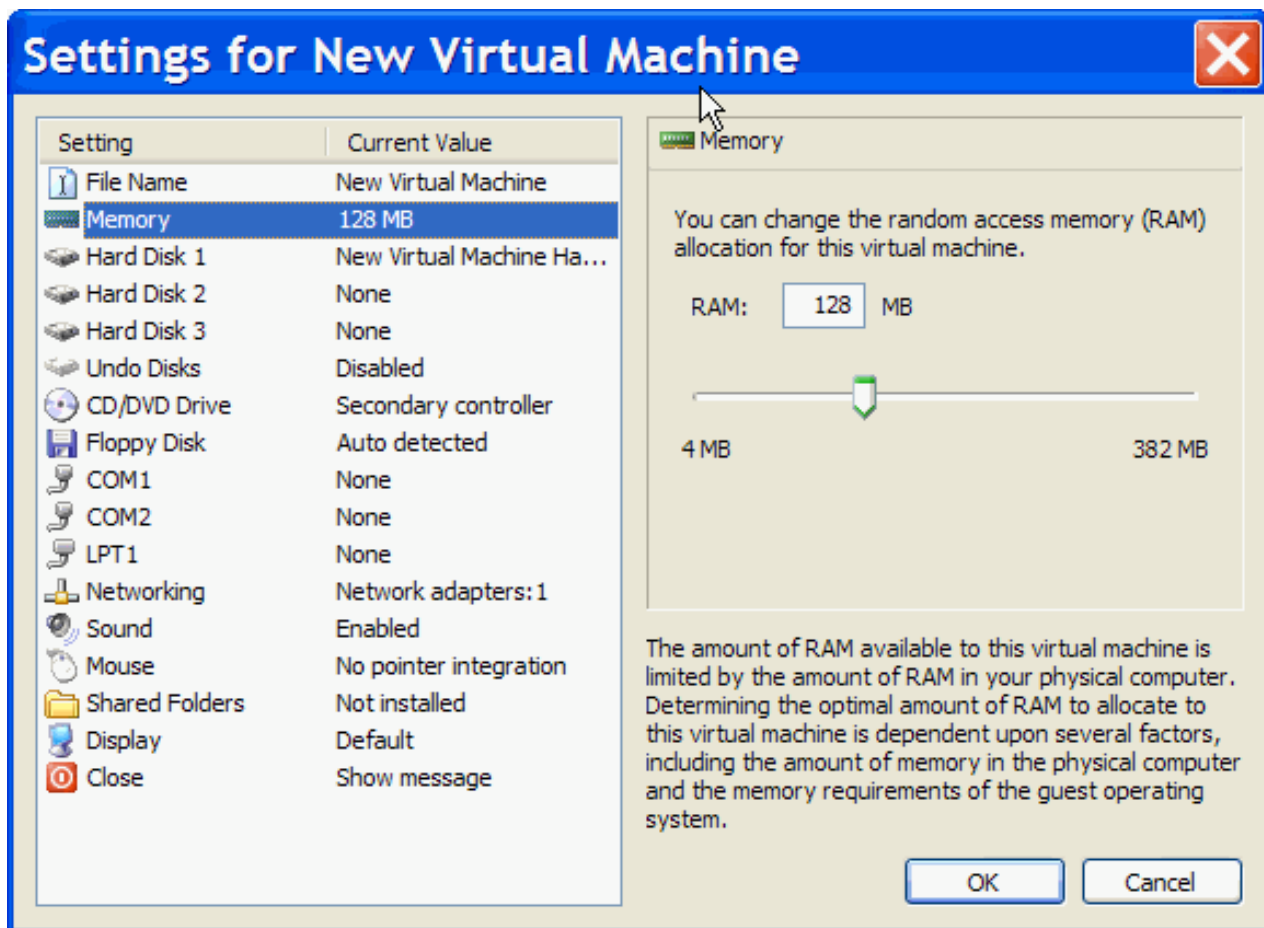


Figure 12

Basic adjustments may also be made to the Virtual PC by selecting the Settings button. Additional drives, memory adjustments and access to network adapters may be selected from the list on the left.

A Virtual PC is identical to a normal PC and may even be run in full screen mode. I would suggest selecting a different wallpaper or theme to denote the active Virtual PCs.

Since you can now run multiple Virtual PCs only limited by available disk space and RAM, you may be wondering if your PC could host an entire network. Well, the answer is yes, and this opens up a whole new area for testing and debugging your applications with just one PC.

Reader Comments

[Add a comment](#)

Although since VPC has been acquired by MS, it no longer...

Copyright © 1999-2004 by [CoveComm Inc.](#) All Rights Reserved. Reproduction in any form without the express written consent of CoveComm Inc., except as described in the [subscription agreement](#), is prohibited.