SPREZZATURA

45 St Mary's Road • Ealing • LONDON W5 5RQ
Tel: 020 8912 1010 Fax: 020 8912 1011
info@sprezzatura.com

SENL

DECEMBER 2006

Sprezzatura's Electronic Newsletter

For Revelation developers by Revelation developers

MAKING DATABASES HAPPEN

SPREZZATURA

Welcome to this special bumper end of year edition of SENL - the journal that just wouldn't die despite trying very very hard. If I had a pound for every time we've said "Must put that in the next SENL" over the past 24 months I'd be considerably richer! The problem is – as most of you will know – getting the work/life balance is tough and sitting down to write when paying clients are knocking on your door is a tough one. Even now our clients may be reading this asking "how can you work on this when you've got deliverables due"? If it's any consolation guys and gals it's 20:10 on a Monday evening and I'm on a train returning home from a client!

Since we last wrote the product has come in leaps and bounds and there's even more to come. The management team at Revelation have concentrated on ensuring that destructive bugs in the product are tracked down and squished – witness some particularly elusive memory leaks which have created problems both here and abroad. Very recently the Rev web site listed some of the goodies to come in version 8.0 and to say that the list is extensive would be an understatement.

It has been a bit strange for all of us at Sprezz over the past few years as our relationship with Revelation has become stronger we've felt a little uncomfortable in how we represent ourselves to the community so we thought it'd be a nice idea to put to rest a few shibboleths.

Firstly Revelation Software Ltd is NOT Sprezzatura under a different name. Revelation Software is a separate company; set up specifically to distribute Revelation in the EMEA (Europe, Middle East and Africa). It has its own staff, a separate bank account etc. As a courtesy to the Revelation community Sprezzatura make technical resource available to Revelation, sometimes on a commercial basis and other times just in the interests of the community. An example of the former is that Carl Pates (whom many of you will know) was actively involved in the release of 7.2 – specifically in relation to internal issues on UI and editable functionality. This does NOT mean that Sprezz have Revelation Source Code. Revelation US are fiercely protective of their IP and access is granted remotely on a "needs must" basis. Following on from the last conference at Vegas, Sprezz were also tasked with making Carl's "Drag and Drop" proof of concept a reality for version 8.0 – another task that has made sure our feet don't hit the ground!

Add to this some of the other exciting projects that we've been involved in (a National Web Enabled Secure system for UK Homelands Security springs to mind – enter data at any one of a number of regional centres and as it updates the local database almost immediately you can create web queries against the same data copied to a centralised database) and a huge amount of skunk works projects and utilities and before you know it another year has passed!

In this issue, we've tried to cram in as much as possible without being overwhelming. We've combined lighter material (a reprint of an article written for another non-Rev journal on outsourcing) with the highly technical (an article on Windows API tricks). We've also included the first part of a 2 part article on using OLE in OpenInsight. We're hoping that'll give us the motivation to ensure another SENL comes out soon! We've also been playing with a less formal writing style pioneered by our very own Mr Kaplan – blog like, if you will – and we're going to intersperse a number of these items throughout this SENL.

As the year draws to a close we've got an exciting conference to look forward to in Seattle. The Sprezz Team will be represented by our usual lead three of Aaron Andrew and Carl and we look forward to seeing you there! We'll be arriving the Saturday before conference and leaving the Saturday after conference. It's amazing what a difference those Saturday night stays make to flights from Europe! As it is we'll be flying for around 11 hours so the getting there early should see us in good stead for getting onto time zone!

Hopefully we'll see y'all in Seattle!

SPREZZATURA

Contents

Debugger of it all – Aaron Kaplan

The other day at a client site, we came upon a line of code similar to the following:

- write record to handle, id else debug

There are many reasons we can think of for putting debug as part of a failure, but none of them are really any good. In fact, there's never any reason to have DEBUG as statement on it's own. Whenever you want to use DEBUG, you should always think of it in one of these ways:

a) if "My @STATION ID" = @STATION then DEBUG

b) if "My USERNAME" = @USERNAME then DEBUG

This way, if the DEBUG does somehow make it into your production systems, the debugger will not be invoked. Yes, we do know that we can intercept the debugger and display, but this is not the same thing. Breaking to the debugger for a program error is one thing. Breaking to the debugger as a specified command and program feature is quite a different matter.

In general, we prefer to have all of our error handling in common routines. Well before Revelation placed FSMSG back into OI, we created our own routines that read in from REVERROR.DAT and displayed the appropriate error text.

The important thing is that at no time should an end-user be deliberately dropped into the debugger. As programmers, part of our job is handling all possible outcomes of a code statement, prompting and informing the end user appropriately.

SPREZZATURA

An Introduction to the Use of OLE controls from OpenEngine – Part 1 - Andrew McAuley

Introduction

For several releases now, OpenInsight has had the ability to utilize OLE controls from within OpenInsight forms. This implementation called for the use of Presentation Server (a component of OINSIGHT.EXE) to work. Whilst this was effective, it meant that if OINSIGHT was not running then OLE controls could not be used. This introduced a problem for people developing Web applications as the sole mechanism available for the production of complex reports (OIPI – an OLE control) could not be used as OINSIGHT is not running.

Realizing that this was a significant disadvantage, Revelation moved quickly to address the issue and as of 7.1, OLE controls can be run in an engine context. Making it possible to use OIPI in an engine context opens the door on a whole host of opportunities. This article will show both how to use the routines provided in OpenInsight to achieve the use of OLE controls, and some sample examples of it in use to perform mail merges and spell checks.

OLE controls are essential Windows controls that permit conversations between themselves and other applications as is obvious from the expansion of the acronym – Object Linking and Embedding. The technology behind OLE has undergone many changes since it began with DDE but we're now at a stage where the underlying technology is mature, robust and easily accessible from OE. Basically the idea is that if an application is sufficiently modular that it might be useful in another application it can be exposed as an OLE control – an example would be the Microsoft Calendar control.

BASIC+ Routines

The Basic+ routines provided for manipulating OLE controls are OLECreateInstance, OLEGetProperty, OLEPutProperty and OLECallMethod. These allow the user to do exactly what they say. They are documented well in the online help but for the purposes of ease of use of this article I'll redocument them here.

Note that all of these functions have been created as opcodes – what this means is that you don't need to declare them, nor call you CALL them. If you attempt to call the routines, RTP27 will look for them and fail to find them and generate an error.

OLECreateInstance

The OLECreateInstance is used to instantiate an OLE control so that we may communicate with it. The syntax is very straightforward. When we create an OLE instance, we receive a handle which we then use to communicate with the object in subsequent OLE routine calls. This handle is referred to as an "IDispatch" pointer and if you examine it in the debugger you will just see the word "Idispatch" – this is because it is a structure and as such is of little use to access directly. This does lead to slight confusion however if you create multiple handles to different processes as they will all appear as Idispatch in the debugger!

The syntax is

```
        objectHandle = OLECreateInstance(OLEControlProgId)
```

Note that the OLEControlProgId is frequently a long confusing string – we'll deal with finding this later but for now we'll just use the online help example of VSPRINTER7.VSPRINTER.1 – the OIPI Printer OLE object.

To DELETE an OLE instance one simply sets the objectHandle to null, eg

```
        objectHandle = ""
```

OLEGetProperty

The OLEGetProperty function returns the property of the OLE control that has been requested. In this it is identical to the OI GET_PROPERTY. Note however that at the time of writing there is a bug extant related to getting and using properties that become IDispatch pointers in themselves. This can GPF the machine – you'll know soon enough if you encounter this bug ☺.

The syntax is

```
objectProperty = OLEGetProperty(objectHandle, objectPropertyName)
```

OLEPutProperty

The OLEPutProperty is functionally equivalent to the OI SET_PROPERTY function. It permits the SETTING (or PUTTING) of a property of an OLE control.

The syntax is

```
void = OLEPutProperty(objectHandle, objectPropertyName, Value)
```

Note that the objectPropertyName can be modified to point to array elements if necessary – see the online help for details.

OLECallMethod

The OLECallMethod permits the running of methods within the OLE control.

The syntax is

```
ReturnValue = OLECallMethod(objectHandle, objectMethodName, args)
```

Where args can be multiple comma delimited arguments.

OLEStatus

After any OLE function it is a good idea to check if it worked. This is achieved by calling the function OLEStatus which returns a result indicating the status of the operation. The result (if unsuccessful) is returned as a negative integer which it is the programmer's responsibility to convert into meaningful text.

The syntax is

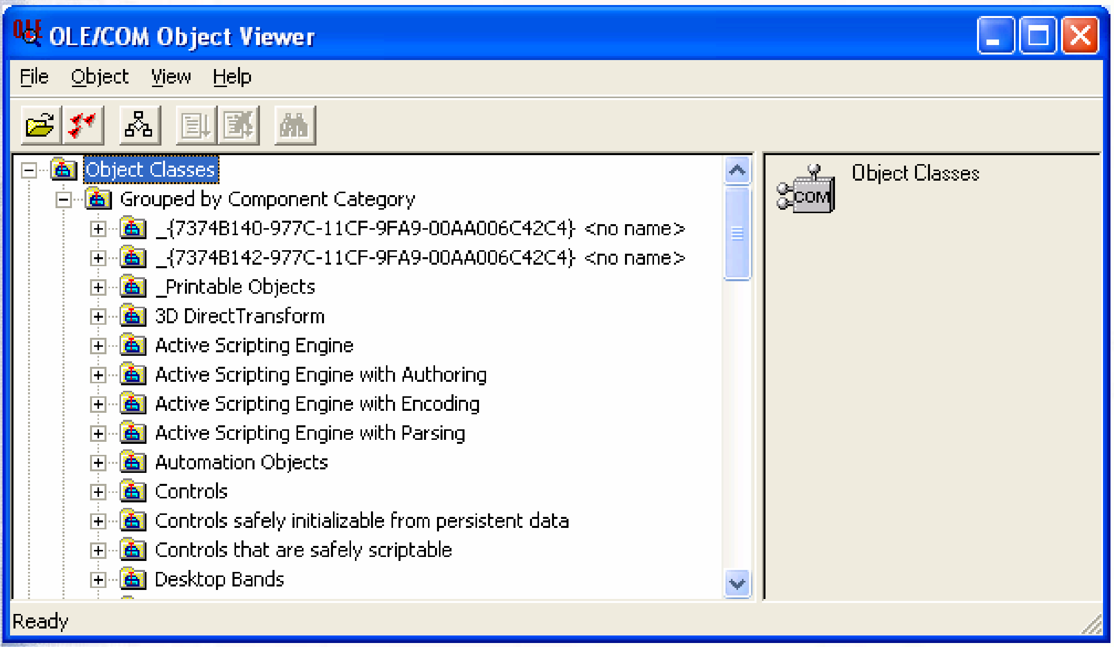
```
OLEStatus = OLEStatus()
```

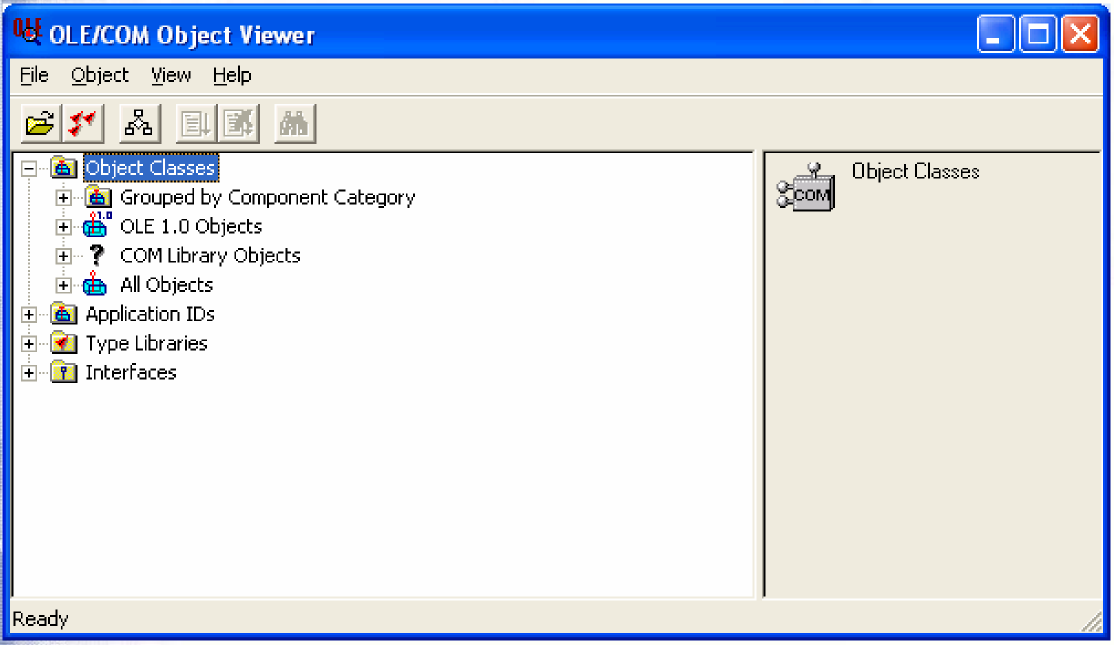
Exploring OLE Controls

How to find OLE Controls

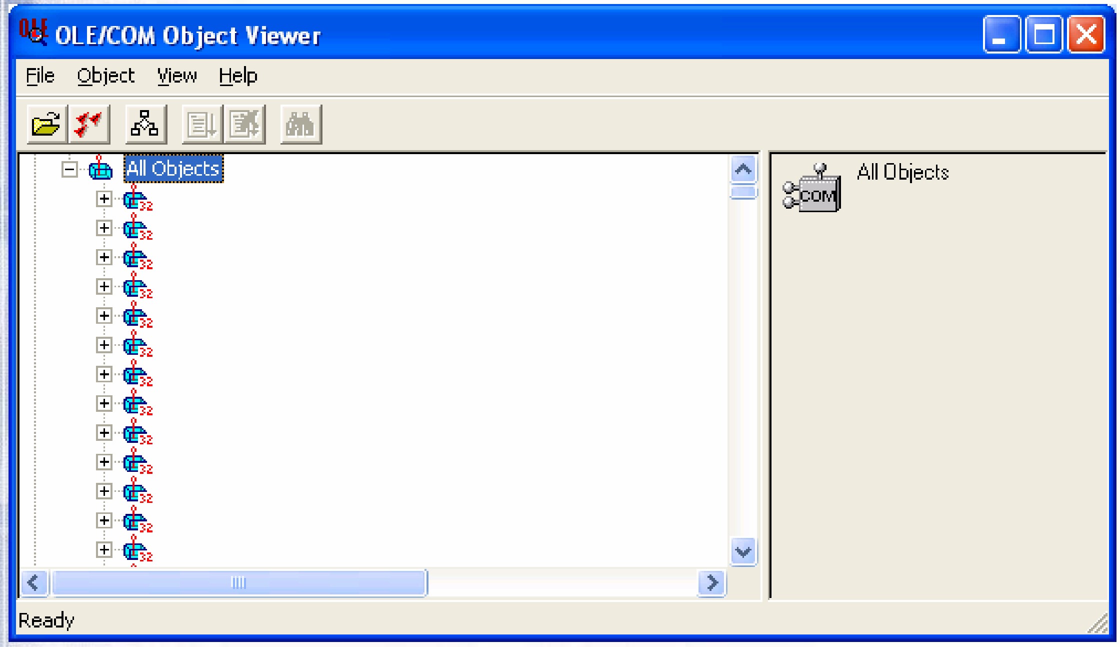
With the wealth of OLE controls out there it can be difficult to know what's on your own machine. There are several ways of getting at this information but the way we're recommending is the use of a Microsoft utility called OLEVIEW. This ships with various Microsoft development environments but not with the operating systems. So to use it you must first download it. At the time of writing this was available from http://support.microsoft.com/default.aspx?scid=kb;en-us;122244

This program locates all of the OLE controls on your machine and provides an interface to get at the information. Running it produces the following screen :-

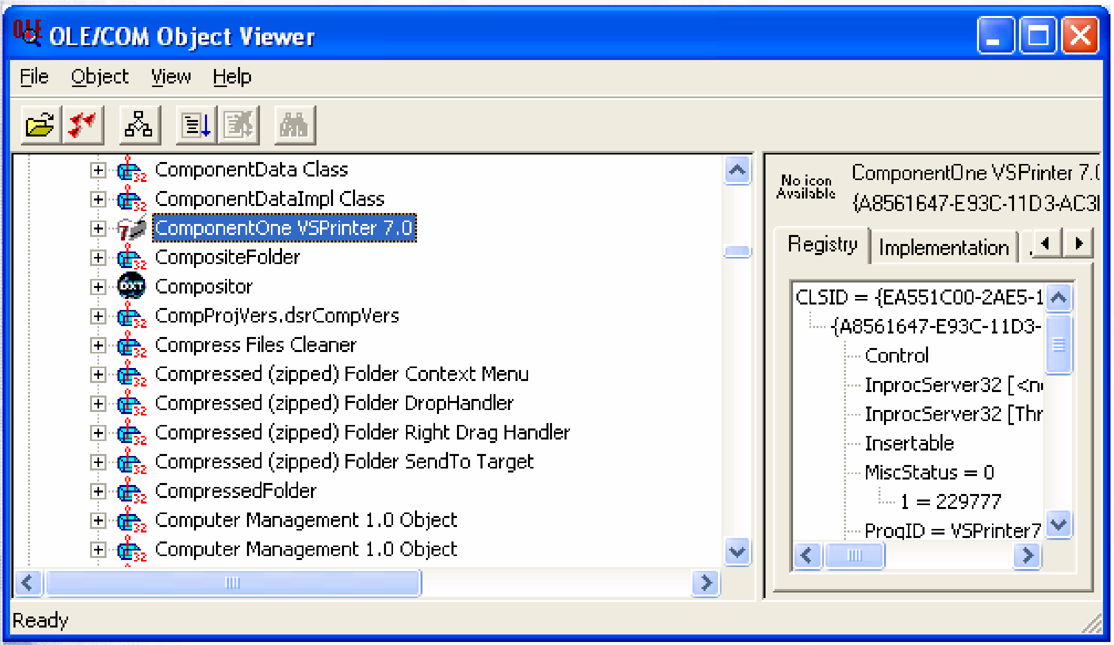

OLE/COM Object Viewer
File Object View Help
Object Classes
Grouped by Component Category
_{7374B140-977C-11CF-9FA9-00AA006C42C4} <no name>
_{7374B142-977C-11CF-9FA9-00AA006C42C4} <no name>
_Printable Objects
3D DirectTransform
Active Scripting Engine
Active Scripting Engine with Authoring
Active Scripting Engine with Encoding
Active Scripting Engine with Parsing
Automation Objects
Controls
Controls safely initializable from persistent data
Controls that are safely scriptable
Desktop Bands
Object Classes
Ready



We'll leave you free to explore the various options vis-à-vis viewing the Object Classes and go straight to the All Objects view… double clicking this results in



So let's scroll down to our VSPRINT object – as this is provided by ComponentOne we'll look under this in the listing…



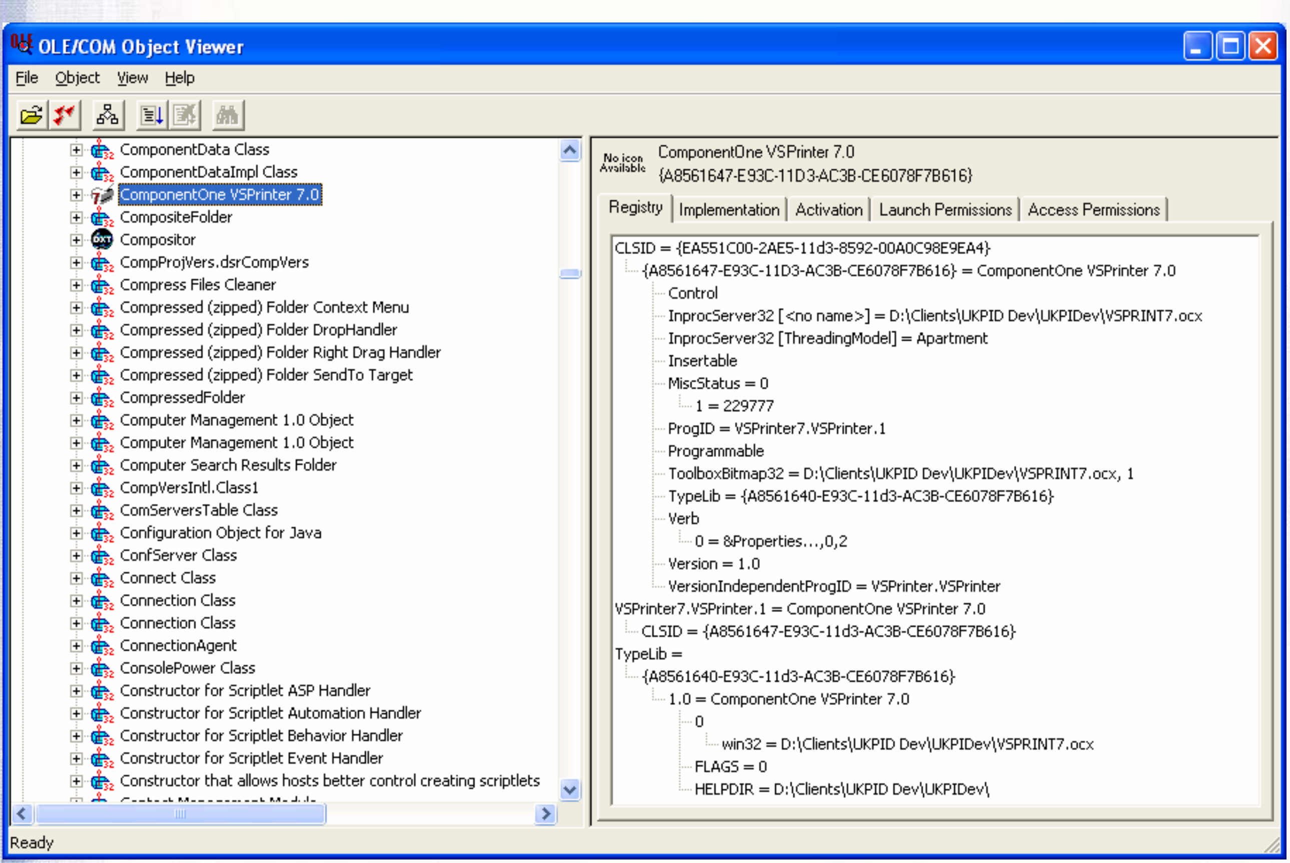
```
ComponentData Class
ComponentDataImpl Class
ComponentOne VSPrinter 7.0
CompositeFolder
Compositor
CompProjVers.dsrCompVers
Compress Files Cleaner
Compressed (zipped) Folder Context Menu
Compressed (zipped) Folder DropHandler
Compressed (zipped) Folder Right Drag Handler
Compressed (zipped) Folder SendTo Target
CompressedFolder
Computer Management 1.0 Object
Computer Management 1.0 Object
```

```
ComponentOne VSPrinter 7.(
{A8561647-E93C-11D3-AC3I

CLSID = {EA551C00-2AE5-1
  {A8561647-E93C-11D3-
    Control
    InprocServer32 [<n
    InprocServer32 [Thr
    Insertable
    MiscStatus = 0
      1 = 229777
    ProgID = VSPrinter7
```

Now we're there what can we find out?

The *great* thing about OLE controls is that they're self documenting – not to the level that replaces proper documentation but at least to the level where we can get an idea about what's available to us!

So for starters let's expand the above screen a little…


OLE/COM Object Viewer
File Object View Help
No icon Available
Registry Implementation Activation Launch Permissions Access Permissions
Ready

ComponentData Class
ComponentDataImpl Class
ComponentOne VSPrinter 7.0
CompositeFolder
Compositor
CompProjVers.dsrCompVers
Compress Files Cleaner
Compressed (zipped) Folder Context Menu
Compressed (zipped) Folder DropHandler
Compressed (zipped) Folder Right Drag Handler
Compressed (zipped) Folder SendTo Target
CompressedFolder
Computer Management 1.0 Object
Computer Management 1.0 Object
Computer Search Results Folder
CompVersIntl.Class1
ComServersTable Class
Configuration Object for Java
ConfServer Class
Connect Class
Connection Class
Connection Class
ConnectionAgent
ConsolePower Class
Constructor for Scriptlet ASP Handler
Constructor for Scriptlet Automation Handler
Constructor for Scriptlet Behavior Handler
Constructor for Scriptlet Event Handler
Constructor that allows hosts better control creating scriptlets

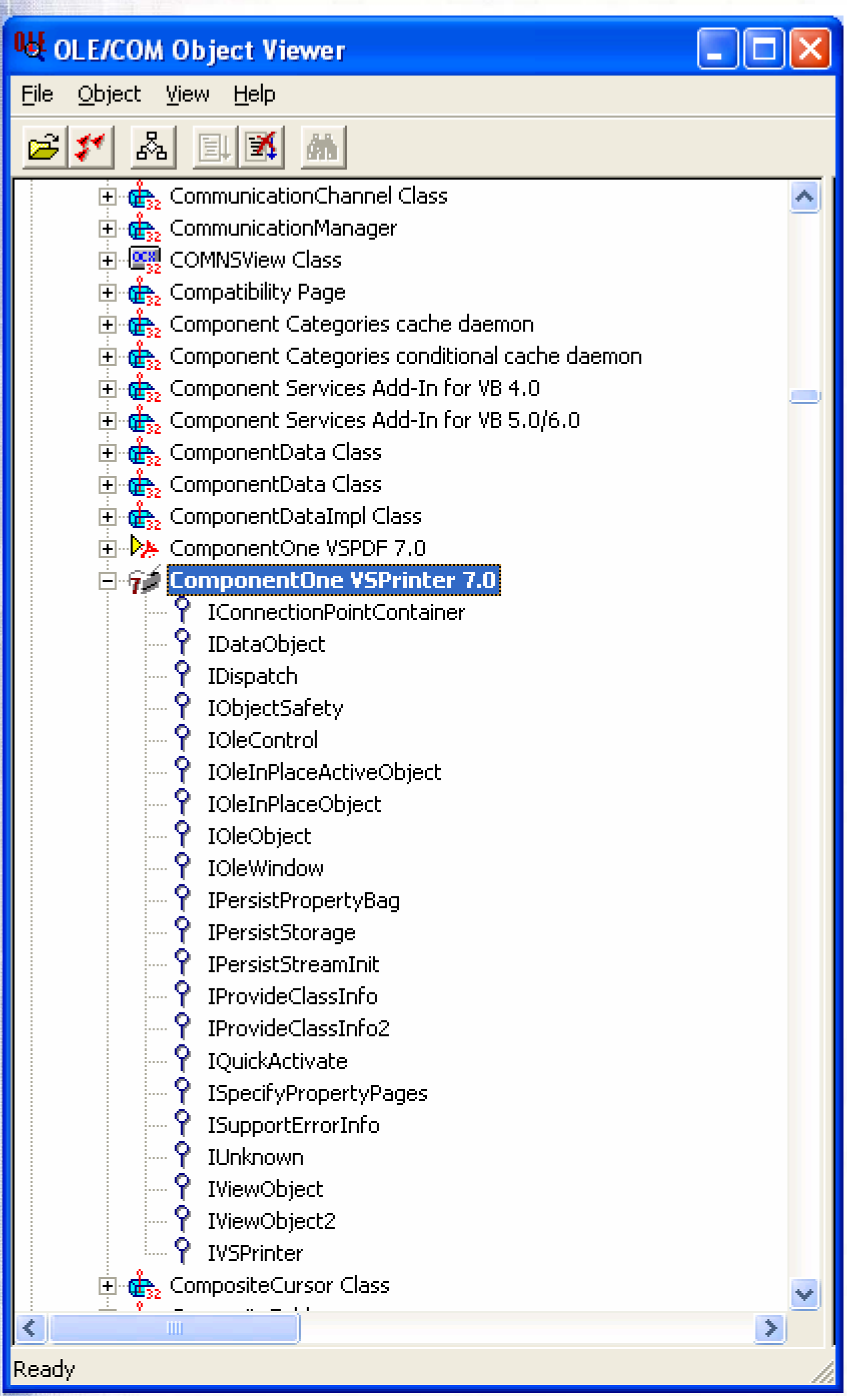
ComponentOne VSPrinter 7.0
{A8561647-E93C-11D3-AC3B-CE6078F7B616}

```
CLSID = {EA551C00-2AE5-11d3-8592-00A0C98E9EA4}
    {A8561647-E93C-11D3-AC3B-CE6078F7B616} = ComponentOne VSPrinter 7.0
        Control
        InprocServer32 [<no name>] = D:\Clients\UKPID Dev\UKPIDev\VSPRINT7.ocx
        InprocServer32 [ThreadingModel] = Apartment
        Insertable
        MiscStatus = 0
            1 = 229777
        ProgID = VSPrinter7.VSPrinter.1
        Programmable
        ToolboxBitmap32 = D:\Clients\UKPID Dev\UKPIDev\VSPRINT7.ocx, 1
        TypeLib = {A8561640-E93C-11d3-AC3B-CE6078F7B616}
        Verb
            0 = &Properties...,0,2
        Version = 1.0
        VersionIndependentProgID = VSPrinter.VSPrinter
VSPrinter7.VSPrinter.1 = ComponentOne VSPrinter 7.0
    CLSID = {A8561647-E93C-11d3-AC3B-CE6078F7B616}
TypeLib =
    {A8561640-E93C-11d3-AC3B-CE6078F7B616}
        1.0 = ComponentOne VSPrinter 7.0
            0
                win32 = D:\Clients\UKPID Dev\UKPIDev\VSPRINT7.ocx
            FLAGS = 0
            HELPDIR = D:\Clients\UKPID Dev\UKPIDev\
```

The first bit of information that is immediately useful is the "ProgID" – you'll recognize this (VSPrinter7.VSPrinter.1) as the OLEControlProgID from earlier. But a quick tip – if all you're doing is using standard non version specific commands then a better name to use is the VersionIndependentProgID as this will work with all versions of the VSPrinter control. In this case this is VSPrinter.VSPrinter. A further tip – if you have multiple OI installations on your machine and one is at a different level to another, the installs from Rev don't REREGISTER the controls. By which I mean I'm currently doing beta testing on my machine in a subdirectory named appropriately. But in preparing this document I've noticed that regardless if I'm running OIPI I'm running the version I installed for a client a while ago!

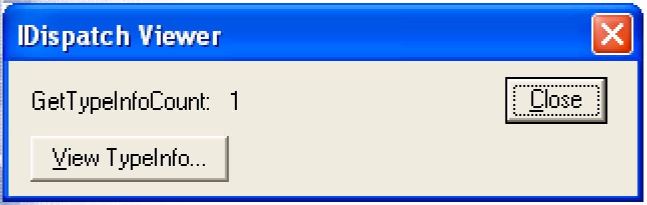
The second thing to note is what happens when we double click the control in the left list box to expand it.

SPREZZATURA



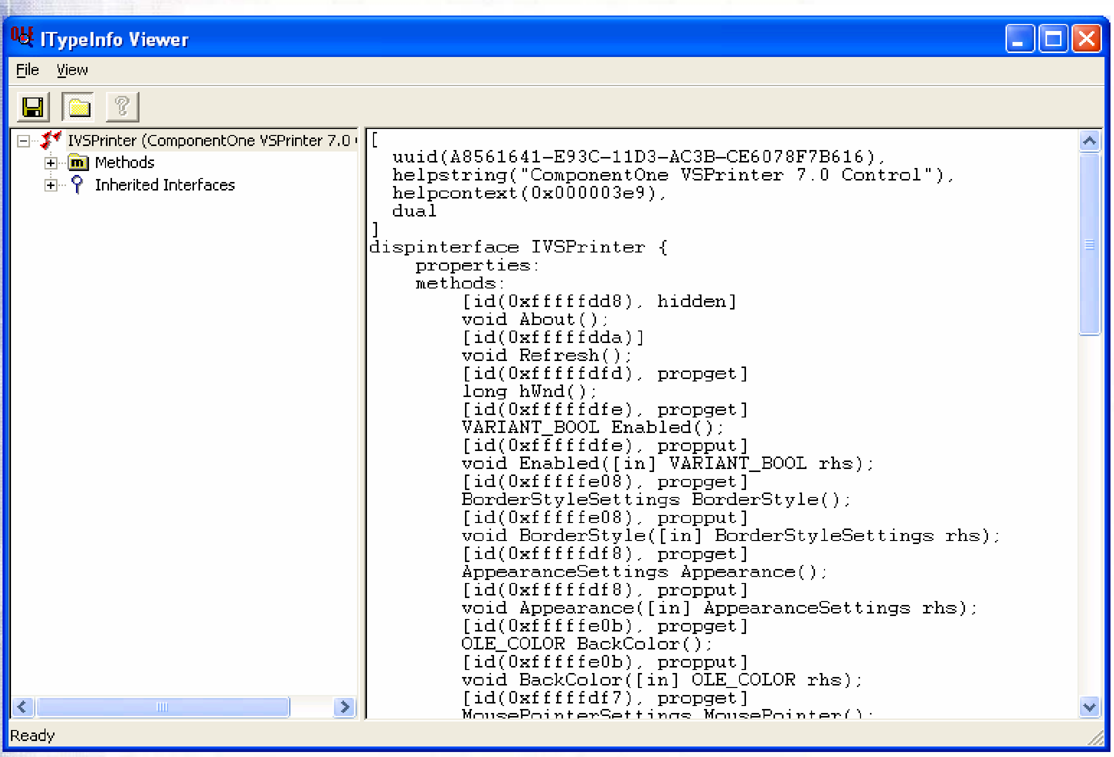
```
CommunicationChannel Class
CommunicationManager
COMNSView Class
Compatibility Page
Component Categories cache daemon
Component Categories conditional cache daemon
Component Services Add-In for VB 4.0
Component Services Add-In for VB 5.0/6.0
ComponentData Class
ComponentData Class
ComponentDataImpl Class
ComponentOne VSPDF 7.0
ComponentOne VSPrinter 7.0
    IConnectionPointContainer
    IDataObject
    IDispatch
    IObjectSafety
    IOleControl
    IOleInPlaceActiveObject
    IOleInPlaceObject
    IOleObject
    IOleWindow
    IPersistPropertyBag
    IPersistStorage
    IPersistStreamInit
    IProvideClassInfo
    IProvideClassInfo2
    IQuickActivate
    ISpecifyPropertyPages
    ISupportErrorInfo
    IUnknown
    IViewObject
    IViewObject2
    IVSPrinter
CompositeCursor Class
Ready
```

We see a list of entry points – but due to the way in which OpenEngine has implemented access all we are interested in is the IDispatch entry point so let's double click that…



GetTypeInfoCount: 1

And click View TypeInfo…

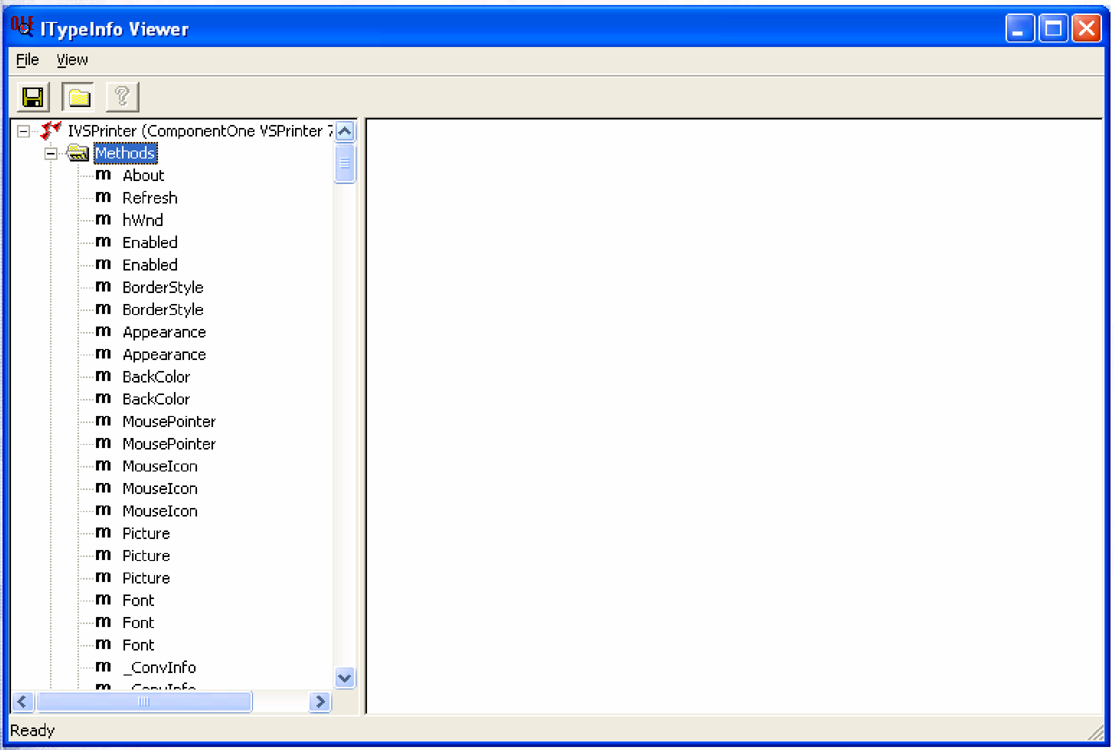


```
IVSPrinter (ComponentOne VSPrinter 7.0
    Methods
    Inherited Interfaces
```

```
[
  uuid(A8561641-E93C-11D3-AC3B-CE6078F7B616),
  helpstring("ComponentOne VSPrinter 7.0 Control"),
  helpcontext(0x000003e9),
  dual
]
dispinterface IVSPrinter {
    properties:
    methods:
        [id(0xfffffdd8), hidden]
        void About();
        [id(0xfffffdda)]
        void Refresh();
        [id(0xfffffdfd), propget]
        long hWnd();
        [id(0xfffffdfe), propget]
        VARIANT_BOOL Enabled();
        [id(0xfffffdfe), propput]
        void Enabled([in] VARIANT_BOOL rhs);
        [id(0xfffffe08), propget]
        BorderStyleSettings BorderStyle();
        [id(0xfffffe08), propput]
        void BorderStyle([in] BorderStyleSettings rhs);
        [id(0xfffffdf8), propget]
        AppearanceSettings Appearance();
        [id(0xfffffdf8), propput]
        void Appearance([in] AppearanceSettings rhs);
        [id(0xfffffe0b), propget]
        OLE_COLOR BackColor();
        [id(0xfffffe0b), propput]
        void BackColor([in] OLE_COLOR rhs);
        [id(0xfffffdf7), propget]
        MousePointerSettings MousePointer();
```

Ready

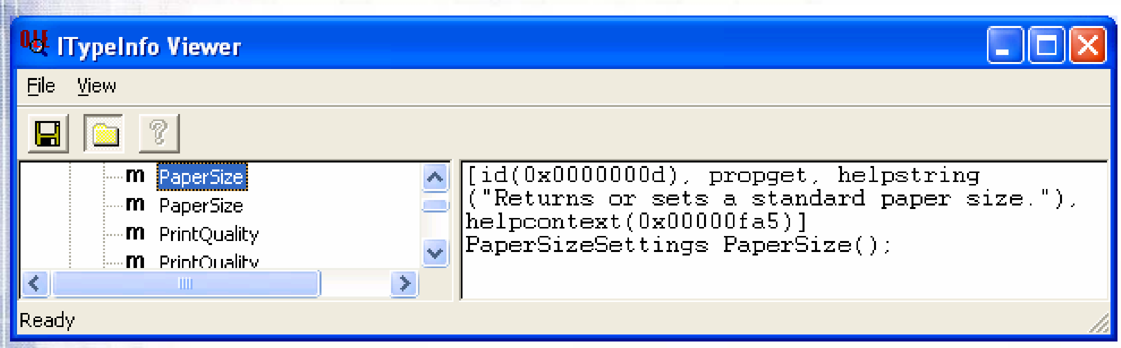
Double clicking Methods produces a listing of all of the methods that VSPrint exposes



```
IVSPrinter (ComponentOne VSPrinter 7
    Methods
        m About
        m Refresh
        m hWnd
        m Enabled
        m Enabled
        m BorderStyle
        m BorderStyle
        m Appearance
        m Appearance
        m BackColor
        m BackColor
        m MousePointer
        m MousePointer
        m MouseIcon
        m MouseIcon
        m MouseIcon
        m Picture
        m Picture
        m Picture
        m Font
        m Font
        m Font
        m _ConvInfo
```

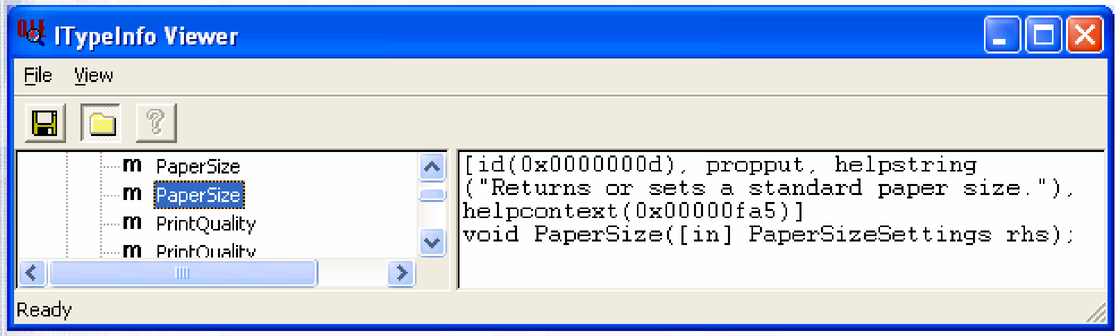
Ready

And as you can see the list is comprehensive. But the confusing thing is that some methods are repeated one or more times. As an example take PaperSize. The reason for this is simple – OLE controls expose properties via Methods. So we'll have a PaperSize Method for getting the PaperSize property and a Papersize Method for setting (or in OLE terms putting) the PaperSize property. Double clicking on the two methods illustrates this. In the first Method we see



```
[id(0x0000000d), propget, helpstring
("Returns or sets a standard paper size."),
helpcontext(0x00000fa5)]
PaperSizeSettings PaperSize();
```

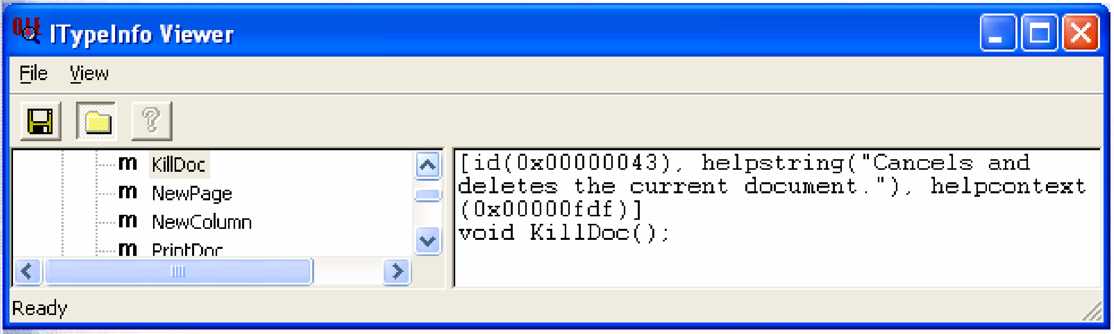
The propGET says that this is the method to get the property. It even contains a help string and a pointer to the help file. In the second we see



```
[id(0x0000000d), propput, helpstring
("Returns or sets a standard paper size."),
helpcontext(0x00000fa5)]
void PaperSize([in] PaperSizeSettings rhs);
```

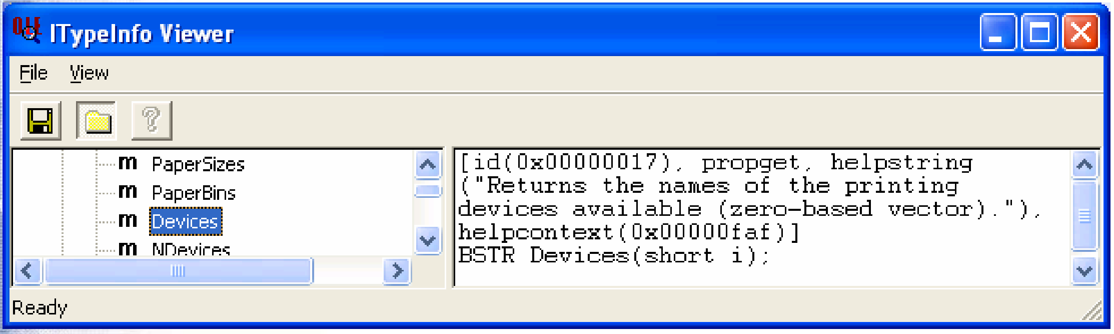
Where the propPUT is setting the property.

If we're actually looking for Methods in the classical sense we'd want to look at those methods which only occur once. Normally these will be callable methods BUT remember that some properties such as the Devices present in the system can only be got so this isn't an infallible rule of thumb. So for example, KillDoc is a Method which can be called



```
[id(0x00000043), helpstring("Cancels and
deletes the current document."), helpcontext
(0x00000fdf)]
void KillDoc();
```

But Devices is not a method, rather the exposing of a get property.



```
[id(0x00000017), propget, helpstring
("Returns the names of the printing
devices available (zero-based vector)."),
helpcontext(0x00000faf)]
BSTR Devices(short i);
```

Naturally we wouldn't suggest that you use this self documentation as the sole source of information about an OLE control but it does provide pointers to allow you to go off and search the internet for additional help using the keyword that you've now found.

SPREZZATURA

Bugs in the system? How about hobgoblins? – Aaron Kaplan

Ralph Waldo Emerson is generally misquoted as "Consistency is the hobgoblin of little minds." In actuality, the quote is "A foolish consistency is the hobgoblin of little minds, adored by little statesmen and philosophers and divines". Some people will take the misquote as license to never do anything the same, since, after all, why would they want to have a little mind? However, there are times when consistency is important, and that's in how you structure your data.

In the Revelation world, we can easily format our data with ready to use ICONVs and OCONVs. Most people automatically put MD2 formatting into all monetary fields, and MD3 in to financial percentages (interest and growth rates, for example). When doing this it should be clear that you'd want all interaction with these data fields to be handled using the proper and consistent conversion. This isn't an Emersonion hobgoblin at all, since the consistency is not foolish, but maintaining data integrity.

We mention this because a client recently suggested that it might be OK to store some information as varying MD3 and MD4, depending on context. Eventually, reason won and an unfoolish consistency was sustained.

BTREE.EXTRACT Gotchas – Andrew McAuley

We were vexed recently by searches that just wouldn't behave no matter how hard we tried. The results we got back were just not what we were expecting. Time after time we pored over the suspect code but it was no good – the code looked fine. We were getting close to tearing our hair out by this time when we remembered that old programmer's maxim

"I don't write buggy code – there must be a problem with the data"

So we took a closer look at what it was we were looking for and noticed that it contained the string "(BLACKBERRY)". This triggered something in the almost vestigial parts of our brain that deal with seriously deprecated commands within Revelation products and sure enough we were eventually able to establish that in addition to the DOCUMENTED Btree.Extract operators there are other operators that are just sitting there waiting to take the unwary coder by surprise. What was actually going on here was that it wasn't looking for the string "(BLACKBERRY)" it was looking for something that MATCHED "BLACKBERRY".

To try to help you avoid falling into the same trap we present below a table of the deprecated operators along with their modern equivalent and some explanatory comments.

(VALUE)	%VALUE	Matches operator. Double up to escape out – in other words to look for (VALUE) enter ((VALUE))
VALUEFROM^VALUETO	VALUEFROM…VALUETO	No way to escape out. If you search for A^B it won't find things containing the string "A^B" it will find things between A and B.

ListBox Control Extras – "Select All" – Carl Pates

One interesting and an undocumented ListBox feature we found recently applies to the SELPOS property.

A simple way to select all the rows in a multi-select ListBox is to set the SELPOS property to "*"

e.g

```
Call Set_Property( @window : ".LIST_1", "SELPOS", "*" )
```

SPREZZATURA

DEP Revisited – Andrew McAuley

Back in March of this year we started to see an increasing number of installation problems with people running Windows 2003 Server or XP Service Pack 2. OpenInsight just wouldn't start on these boxes – it would abend out with a number of different error messages. We did some initial investigations and documented our findings for the community in a White Paper available at http://www.sprezzatura.com/downloads/dep.pdf

As the year wore on we saw increasingly strange occurrences – Citrix farms where OpenInsight applications would just disappear from memory, AREV applications which would just randomly disappear from memory. We weren't able to fully grasp what was going on – all we knew was that disabling DEP on the server seemed to be a workaround IF that was an acceptable workaround to the IT Department. Increasingly though this was not an acceptable work around – DEP was put there for a security based reason and most IT literate people weren't happy with it being disabled.

Finally in a recent installation we came across the proverbial last straw. We were rolling out a local site in what was to be a nationwide rollout of a highly secure "homelands security" type application. This application is the one mentioned earlier in the editorial. At one specific site every time we ran our app it would just crash and burn. At this particular site the IT Department just laughed at our request to disable DEP. They were running other applications on this server – "did we think they were mad?".

It's funny how panic can stoke the fires of creativity! Casting our mind back we were able to remember that when we originally reported the DEP problem we believed that for various reasons we'd tied it into the RCL4.DLL Universal Driver client. Quickly we replaced the UD client with the 2.1 client and the problem disappeared.

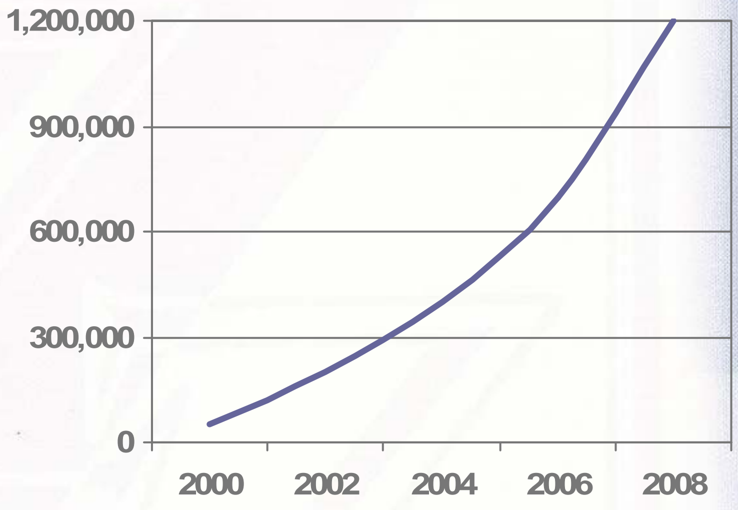
Having learnt the "trick" we started applying it to other scenarios were we were experiencing failures :- Windows 2000 Citrix Server Farms, AREV databases and sure enough the problems started to disappear.

Please note that we are not definitively saying that we have identified where the problem lies – just identified a work around that helps us out on a number of sites. Of course this then leaves you with the problem of having UD3 files which will no longer be accessible if you switch back to the 2.1 Service. We expect to have utilities to help cope with this in due course.

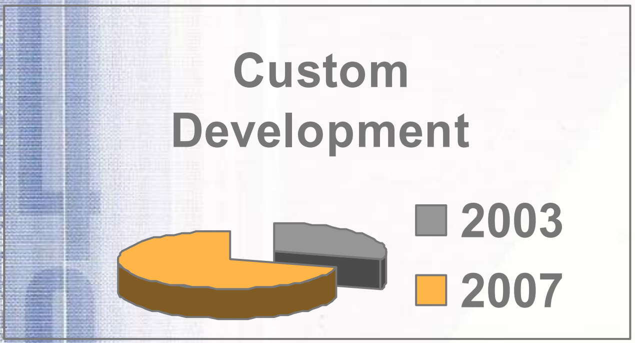
Funny Talking Foreigners Try To Destroy US Economy! – Really? Andrew McAuley

A lot has been published recently about outsourcing. The very issue of who outsources, and why? - is such a political hot potato that it can form the basis of entire governmental campaigns. It seems that you can't open a magazine these days without some attention grabbing headline (such as above) drawing your attention to the fact that your job is under threat from some ill defined foreign enemy who wishes to steal the bread from your table and destroy your well earned way of life. You may even have recently received an email in which you are urged (somewhat xenophobically) to hang up the phone if the person at the other end "sounds foreign". There goes the entire US call industry from a European perspective.

The reasons behind such fear can be found by studying the following diagrams – firstly let's look at the projected growth of new white collar jobs in India over the next few years. Note that the growth rate is pretty much exponential! All of these figures and graphs are borrowed from recent articles in Fortune and Wired magazines – lest you think I'm inventing them!



In 2003, 18% of Software Maintenance jobs were sent abroad from the US. It is projected that by 2007 this figure will be 47%. In 2003, 20% of Custom Software Development jobs were sent abroad from the US. It is projected that by 2007 this figure will be 47%.



Now a lot of people reading this article will probably be involved in custom software development so the question comes, how did all this happen? And more importantly – how can we fight back?

So are the foreigners to blame? The author would contend that the blame can actually be laid fairly and squarely at the feet of some companies a little closer to home – namely Microsoft *et al*.

Consider :-

We've now moved into the area where computer systems can reasonably be considered commodities. So let's take a look at the history of commodities. A simple example would be furniture. When people first began to build furniture it was a very specialist craft and only the rich were able to afford it. Thus furniture became something that was passed down from generation to generation as heirlooms. (This led to a memorable put down in UK politics where a well known but moneyed politician referred to a less wealthy politician as 'the kind of person "who bought his own furniture"').

The specialists were able to charge handsomely for their time and frequently became wealthy in their own right, spawning schools of apprentices and imitators. As more people owned property and had disposable income more people could buy furniture but there weren't enough craftsmen to go around and, in any case, that was still too expensive. As there were so many people wanting to buy, entrepreneurs could invest in mechanized ways of creating furniture and mass production of furniture was born. Bringing this more up to date, we only have to look at cars. Originally the preserve of the very rich, Henry Ford standardized production and brought cars to the masses (as long as you wanted a black car).

Now if you were to plot a curve of skill levels versus cost per hour you would get a normal curve. The really skilled people charge a lot per hour and there are very few of them. The partially skilled people are relatively inexpensive and exist in the industry in greater quantities. The unskilled have no place in the industry.

It therefore follows that if you can make the delivery of software componentized so that the ACTUAL skill levels required to deliver a component are relatively low but specialist, then you can churn out a LOT more components. Take as an example a factory line worker assembling a specific part of a car. The task itself may be complex but it can be learned and once learned repeated ad infinitum.

If there is sufficient demand for standardized custom software (not the oxymoron it appears) then you can set up assembly lines of Business Analysts -> DBAs -> C++ programmers -> VB Programmers -> ISA experts -> IIS Experts -> you get the picture. Each level is specialist BUT you can become an MCP in them in a matter of weeks - not true of truly skilled jobs.

Once at this level you (as the vendor of the tools used) can sell huge amounts of software. It is actually in your interests to keep this componentization as unitised as possible as you will sell more. But once you reach this level where skills may be acquired by rote there is bound to be an outflux of work to where such skills are even cheaper.

Of course the corollary is that the only people who can afford to develop software using this model are the people who can afford the production line, thus effectively alienating the SME who then becomes dependant on other software factories and must reduce their expectations accordingly.

How can ANYONE in their right mind believe that a team of 5-10 people can deliver a custom solution more economically than a team of one? But that is the myth that MS have sold and why? Because then they'll sell 5-10 pieces of software not 1.

So if you’re a small to medium sized shop and you don’t want to invest in the huge capital outlay required to set up a production line facility – and you don’t want to see your work taken over by somebody with less skill but cheaper resources you have to look once again at your skill sets. Looking back at our earlier example – car manufacturers. What car manufacturers continue to do well in these days of cheaper foreign production? I know that many examples exist in the US and in the UK we have manufacturers such as Morgan and Bristol – who not only sell cars successfully but have waiting lists for their products!

How do they manage this? Do they ignore the benefits of componentization in favour of lovingly handcrafting everything? No! The combine the best of both worlds, they source components for basic building blocks then they hand assemble custom deliverables. They provide, as one of our colleagues used to say, “a boutique experience”. And how does Fortune magazine suggest that this issue be addressed? At http://www.fortune.com/fortune/careers/articles/0,15114,588335,00.html the author says

"The answer for programmers, as for anyone in a competitive business, is to specialize to the point that you offer greater productivity, or change careers,"…And of another worker.. He developed new programming methods, "which allowed him to write certain specialized applications about eight times faster than his typical Bangalore competitor," says Mitchell. "Now he has a cost advantage. But only in those narrow areas."

As a supplier of integrated 4GL technologies Revelation Software believe that 4GLs are once again on the up because they are the only way SMEs can fight back AND the only way skilled programmers can see to their job staying in this country. Note once again that the blame does not lie on the “foreign countries” for taking advantage of the componentization of software. It lies with those who have componentized it. Regretfully as demand exceeds supply this is bound to happen. Our defence ought not to be a Luddite one – of smashing up the means of production in the hope of stemming the tide. Rather we ought to realise that not everyone wants or needs mass produced software. So our defence is to do what we do already but to do it better!

Also, the answer to 'how can a small software house deliver more / faster ' lies in another 1980's maxim: "Don't automate - obliterate" The idea is to solve problems by not having them. In particular, when you have a lot of layers in a process then the cost of handing off between layers can exceed the benefit of the layers. The Microsoft model of a software factory embeds a very high process cost that can be reduced by cheap labour, but can never be totally removed.

In business this led to the 'caseworker' model, where caseworkers brought knowledge to decision makers. In computing this has not yet happened. However a 4GL in competent hands can make it so. Decent 4GLs permit the rapid prototyping of applications with the eventual end user a very real possibility. An integrated 4GL environment permits the developer to have the best of both worlds – using the 4GL to set up the framework and initial business model and the 3GL to instantiate more complex business rules. Add in the tight coupling of business rules to the database layer and we are able to once again deliver bespoke software applications at near to mass production prices. We keep our jobs and our clients receive a better service!

We'd strongly encourage you to download an evaluation copy of our OpenInsight software which can be found at http://www.revsoft.co.uk. There you'll also find easy to follow tutorials to help you to get the most from your download. Remember the next time someone tells you that foreign companies are stealing your work remind them it wouldn't have been possible without the help of an all too American company!

More Consistency – Aaron Kaplan

We were talking the other day about how an application should flow. Some people said that apps should follow Windows standards. The more cynical wondered "what standards?", but we're not here to bash Microsoft. In the end, an application should flow how the user expects it to flow. For example, when you save a row, does the window clear? That depends on what type of application you are writing. If this is an ARev app you are converting, especially one that's living in tandem, then, Windows standards aside, perhaps that window should clear the screen, since that's how users expect it to be.

It's a fine line between F9 for save and Ctrl-S for save. Which is correct is not as much for you to decide, but for the users. Programmers and designers shouldn't force their whims or the flavour of the day on to the end users, but should work in conjunction with them, to ensure that the application does what the users want, and how the users want. This means, that if an end user really, really, really wants a "Are you sure you want to save?" message, then a "Your record has been saved" message, as much as it might gall you to put that in, at the end of the day, it's their system.

Scales of Coding

Our discussions about consistency moved us into two different topic realms. One was about handling maths when working with ICONV data. The other was about Hungarian notation. In the end, most of us were of the opinion that, at least in terms of numbers, ICONV/OCONV is more for data validation and consistency rather than as a storage mechanism. Not that we're advocating the needless waste of disk space, and I suppose a megabyte here and a megabyte there and soon you're talking gigs….but, in the end, it's not as important as it was even five years ago.

The question is really around how to handle the maths. If I have my numerics stored as ICONV MD2, and I do something like

```
origPrice = record< OrigPrice$ >
reduction = record< Reduction$ >

customerPrice = origPrice - reduction
```

we're all pretty set on what the math is.

But, suppose I'm adding in tax, and we're in one of those places were the tax is variable based on location of the store, the delivery address or the billing address. Suppose now that the tax value is also stored MD2. Our code now reads

```
origPrice = record< OrigPrice$ >
reduction = record< Reduction$ >
taxAmt = record< TaxAmt$ >
customerPrice = ( origPrice - reduction )
actualPrice = customerPrice + ( customerPrice * taxAmt)
```

actualPrice is very different if we change that code to

```
taxAmt = oconv( taxAmt, "MD2" )
actualPrice = customerPrice + ( customerPrice * taxAmt )
```

The idea is that it's important, when working with numbers, to ensure that your scaling is correct. The best way to do that is when working with numbers, to always OCONV your data out to the correct scale before working and manipulating them.

How all this relates to Hungarian notation will have to wait until later in SENL.

XP Edit Control Extras – Carl Pates

In the last issue of SENL we took a brief look at enabling OpenInsight applications to use Windows XP visual styles. This capability has become an integral part of OpenInsight since the 7.2 release and along with this change also comes some interesting new capabilities.

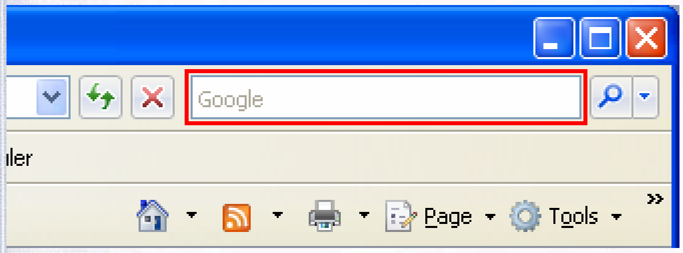
In this article we are going to look at two features that Microsoft added to the standard Edit Control that are available to OI systems running with XP visual styles:

"Cue Banner" support
"Balloon-Tip" support

Cue-Banners

A Cue Banner is textual information (a cue) that is displayed by the edit control when it contains no data. It is designed to prompt or help the user to enter information into the control.

The use of this feature is very common these days – mostly it can be seen in Web Browser search controls such as the IE7 search box shown below.



To implement this feature is quite straightforward and involves sending a simple message, EM_SETCUEBANNER, to the control along with the text that you want to use. Microsoft documents the message as:

```
EM_SETCUEBANNER

Syntax:

    lResult = SendMessage( hwndControl,
                           message,
                           wParam,
                           lParam );

Parameters

    hwndControl          Handle to target edit control
    message        Message ID - should be EM_SETCUEBANNER
    wParam         Not used - must be zero
lParam      Pointer to a Unicode string that contains the text to display as
the textual cue

Return Value
```

If the message succeeds, it returns TRUE (1). Otherwise it returns FALSE (0).

One thing to note the EM_SETCUEBANNER message is that the text string must be in Unicode, not ANSI or UTF8, so we will have to do a little pre-processing before we can send it.

Note that this uses the standard Windows API SendMessage function that will already have been prototyped on your OI system so there's nothing to worry about there.

For those of you not familiar with SendMessage it's a fundamental part of Windows UI programming and is used by applications to access a control's functionality. Every UI object in Windows (i.e. windows and controls) exposes an interface by which it can be manipulated. This takes the form of a series of messages (identified by a number) that the object responds to and the SendMessage function is used to access this interface.

SendMessage takes four integer arguments and returns an integer that usually indicates success of some sort.

```
SendMessage

Syntax:

   lResult = SendMessage( hwndControl,
                          message,
                          wParam,
                          lParam );


Parameters

   hwndControl          Handle to target control
   message        Message ID
   wParam         Message specific information
   lParam   Message specific information

Returns

The return value specifies the result of the message processing - it is message
dependant
```

Example

```
   retVal = SendMessage( hwndObj, WM_GETTEXT, cbText, lpText )
```

EM_SETCUEANNER – an example

The function

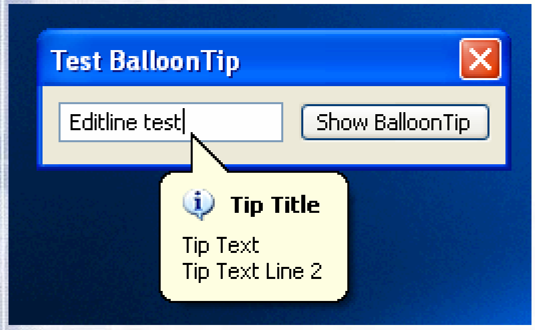
```
function setEditCueBanner( objEdit, cueText )

   declare function sendMessage, isUTF8, ANSI_Unicode, UTF8_Unicode
   declare function get_Property

   equ ECM_FIRST$         to 0x1500
   equ EM_SETCUEBANNER$   to ECM_FIRST$ + 1

   * // Get the control handle
   hwndEdit = get_Property( objEdit, "HANDLE" )

   cueText := \00\ ; * // Null terminate

   * // Now translate the text to Unicode
   if isUTF8() then
      cueTextW = UTF8_Unicode( cueText )
   end else
      cueTextW = ANSI_Unicode( cueText )
   end

   lockVariable cueTextW as LPWSTR
   lpCueTextW = getPointer( cueTextW )

   retVal = sendMessage( hwndEdit, EM_SETCUEBANNER$, 0, lpCueTextW )

   unlockVariable cueTextW

return retVal
```

To Use

```
   retVal = setEditCueBanner( @window : ".EDITLINE_1", "Search" )
```

Implementing Balloon-Tips

The next feature we are going to look at is the Edit control Balloon-Tip. Basically this is an extended multi-line tooltip control that can optionally display an icon along with the text like so:



Again, this feature is implemented by use of the Windows API SendMessage function but this time it's a little more complex as we will have to define and use a 'C' structure to pass along with the message (EM_SHOWBALLOONTIP) as you can see from the message definition below:

```
EM_SHOWBALLOONTIP

Syntax:

   lResult = SendMessage( hwndControl,
                          message,
                          wParam,
                          lParam );

Parameters

   hwndControl          Handle to target edit control
   message        Message ID - should be EM_SHOWBALLOONTIP
   wParam         Not used - must be zero
   lParam    Pointer to an EDITBALLOONTIP structure that contains information
about the balloon tip to display.

Return Value

If the message succeeds, it returns TRUE (1). Otherwise it returns FALSE (0).
```

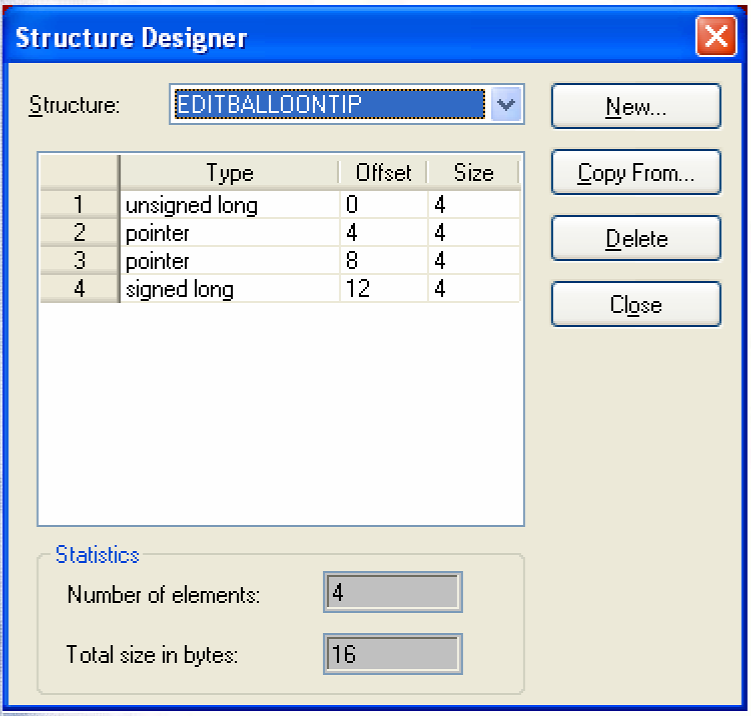
Describing the full use of 'C' structures in OI is beyond the scope of this article so for now we'll just cover what we need for the EM_SHOWBALLOONTIP message.

The EDITBALLOONTIP struct

This is a structure comprised of four separate variables and defined by Microsoft as:

```
EDITBALLOONTIP

   typedef struct tagEDITBALLOONTIP {
      DWORD   dcStruct;
      LPCWSTR pszTitle;
      LPCWSTR pszText;
      INT     ttiIcon;
   } EDITBALLOONTIP, *PEDITBALLOONTIP;


Members

   cbStruct Contains the size in bytes of the structure
   pszTitle Pointer to a UNICODE string that contains the balloon tip title
   pszText  Pointer to a UNICODE string that contains the balloon tip text
   ttiIcon        Specifies the icon to display:

            TTI_ERROR          Use the error icon
TTI_INFO    Use the information icon
            TTI_NONE           Don't use an Icon
            TTI_WARNING        Use the warning icon
```

C-Structures in OI are created by the DEFINE_STRUCT form, so we need to execute this and translate the definition above to one that OI can understand like so:



Structure: EDITBALLOONTIP

	Type	Offset	Size
1	unsigned long	0	4
2	pointer	4	4
3	pointer	8	4
4	signed long	12	4

Statistics

Number of elements: 4

Total size in bytes: 16

Once you have created this we can then use it in our programs to create a balloontip like so:

```
function setEditBalloonTip( objEdit, tipTitle, tipText, tipIcon )

   declare function get_Property, struct_Len, var_To_Struct, isUTF8
   declare function ANSI_Unicode, UTF8_Unicode, SendMessage

   equ ECM_FIRST$               to 0x1500
   equ EM_SHOWBALLOONTIP$       to ECM_FIRST$ + 3

   equ TTI_NONE$                to 0
   equ TTI_INFO$                to 1
   equ TTI_WARNING$             to 2
   equ TTI_ERROR$               to 3

   hwndEdit  = get_Property( objEdit, "HANDLE" )
   tipTitleW = tipTitle : \00\    ; * // Null terminate
   tipTextW  = tipText : \00\

   * // Translate the tip icon to a numeric value to match the
   * // TTI_ equates above
   locate tipIcon in "INFO,WARNING,ERROR" using "," setting tti else
      tti = 0
   end

   * // Translate the two text components to Unicode
   if isUTF8() then
      tipTitleW = UTF8_Unicode( tipTitleW )
      tipTextW  = UTF8_Unicode( tipTextW )
   end else
      tipTitleW = ANSI_Unicode( tipTitleW )
      tipTextW  = ANSI_Unicode( tipTextW )
   end

   lockVariable tipTitleW as LPWSTR
   lockVariable tipTextW as LPWSTR

   * // Create the EDITBALLOONTIP struct:
   * //
   * // The first member is set to the size of the structure (for
   * // future version change protection)

   ebts = ""
   ebts<1> = struct_Len( "EDITBALLOONTIP" )
   ebts<2> = getPointer( tipTitleW )
   ebts<3> = getPointer( tipTextW )
   ebts<4> = tti

   * // Translate to it its internal 'C' format
   ebts = var_to_Struct( ebts, "EDITBALLOONTIP" )

   * // Now get the structure address
   lockVariable ebts as BINARY
   pedts = getPointer( ebts )

   * // And set the tooltip …
   retVal = sendMessage( hwndEdit, EM_SHOWBALLOONTIP$, 0, pedts )
   unlockVariable ebts
   unlockVariable tipTextW
   unlockVariable tipTitleW

return retVal
```

To use:

```
retVal = setEditBalloonTip( @window : “.EDITLINE_1”,         |
                          “Tip Title”,                     |
                          “Tip Line 1” : \0D0A\ : “Tip Line 2”, |
                               “INFO” )
```

S/Log – Updating the Windows Event Log from OI – A Free Utility - Andrew McAuley

Introduction

With the introduction of OI 7.1.1 it became possible to intercept debugger crashes and substitute the call to the debugger with a call to a developer's own routine. For people deploying commercial applications this a real benefit as users frequently ignore crashes and just continue without providing vital feedback. To capitalize on this capability Sprezzatura are pleased to announce the general availability of S/Log - a routine for updating the Windows Event Log with any information the developer wishes to put there.

Installation

Installation is straightforward. Unzip the downloaded file (found at http://www.sprezzatura.com/slogdownload.htm) to a CLEAN subdirectory then log into the application in OpenInsight you wish to use S/Log in. Open the system monitor and type

RUN RDKINSTALL "Location"

and press return - where location is where you have unzipped the slog.zip file to.

Using S/Log

S/Log installs the following records into your system.

Stored Procedure Debug Tables

- ZZ_EVENTLOG

Stored Procedure Executables

- FORMATMESSAGEA
- FORMATMESSAGEW
- REGISTEREVENTSOURCEA
- REGISTEREVENTSOURCEW
- REPORTEVENTA
- REPORTEVENTW
- ZZ_EVENTLOG
- FORMATMESSAGE
- REGISTEREVENTSOURCE
- DEREGISTEREVENTSOURCE
- REPORTEVENT

Stored Procedure Inserts

- EVENTLOG_EQUATES
- FORMATMESSAGE_EQUATES

The routine that you'll actually be using is ZZ_EventLog which takes four parameters as follows

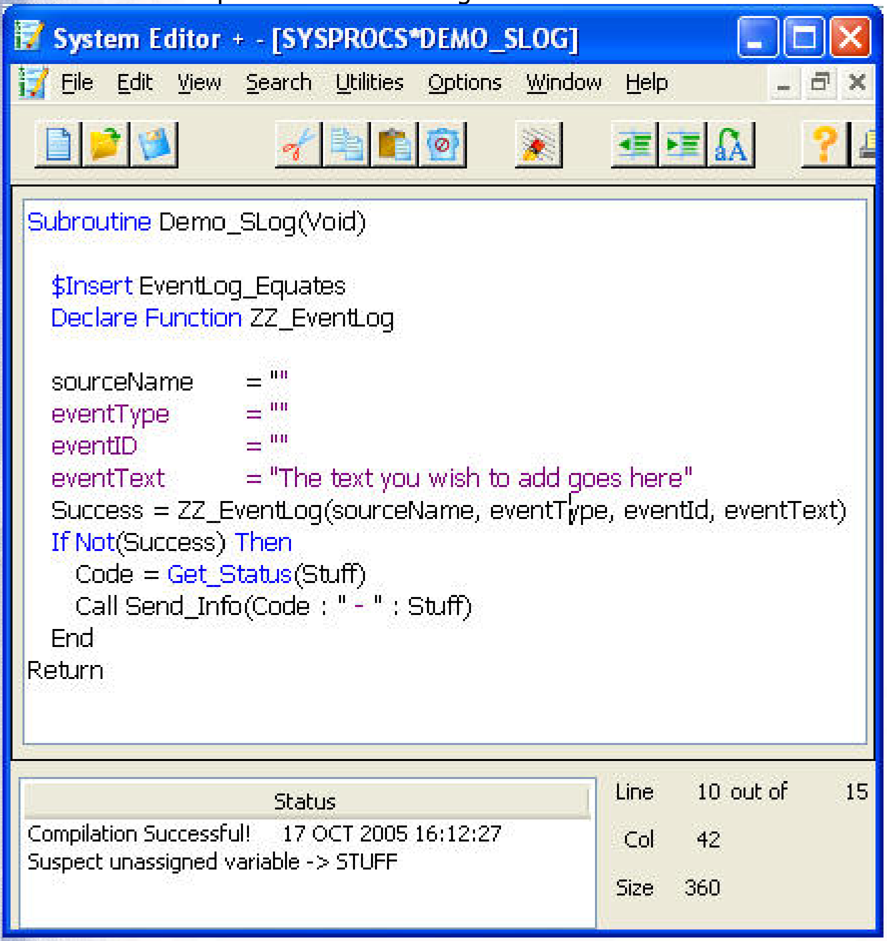
Result = ZZ_EventLog(sourceName, eventType, eventID, eventText)

The system will provide meaningful defaults for all of these parameters other than eventText.

SPREZZATURA

EventText

This is the text to put into the event log.

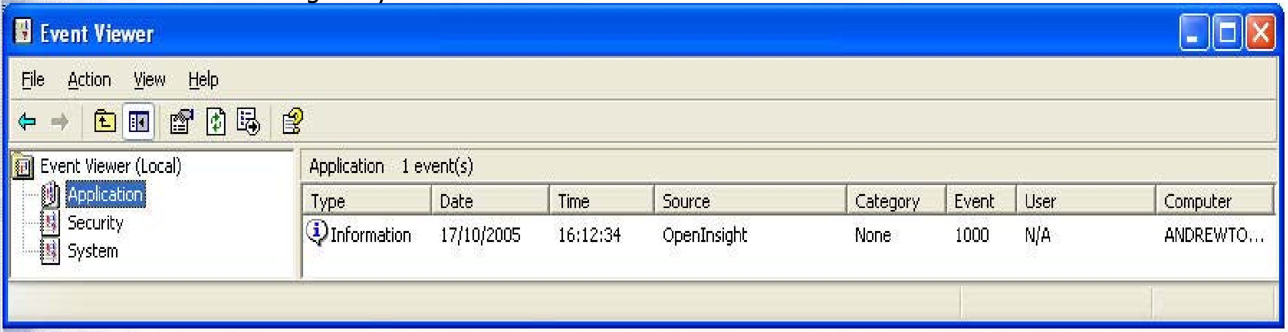

System Editor + - [SYSPROCS*DEMO_SLOG]
File Edit View Search Utilities Options Window Help

```
Subroutine Demo_SLog(Void)

  $Insert EventLog_Equates
  Declare Function ZZ_EventLog

  sourceName      = ""
  eventType       = ""
  eventID         = ""
  eventText       = "The text you wish to add goes here"
  Success = ZZ_EventLog(sourceName, eventType, eventId, eventText)
  If Not(Success) Then
    Code = Get_Status(Stuff)
    Call Send_Info(Code : " - " : Stuff)
  End
Return
```

```
Status
Compilation Successful!     17 OCT 2005 16:12:27
Suspect unassigned variable -> STUFF
```

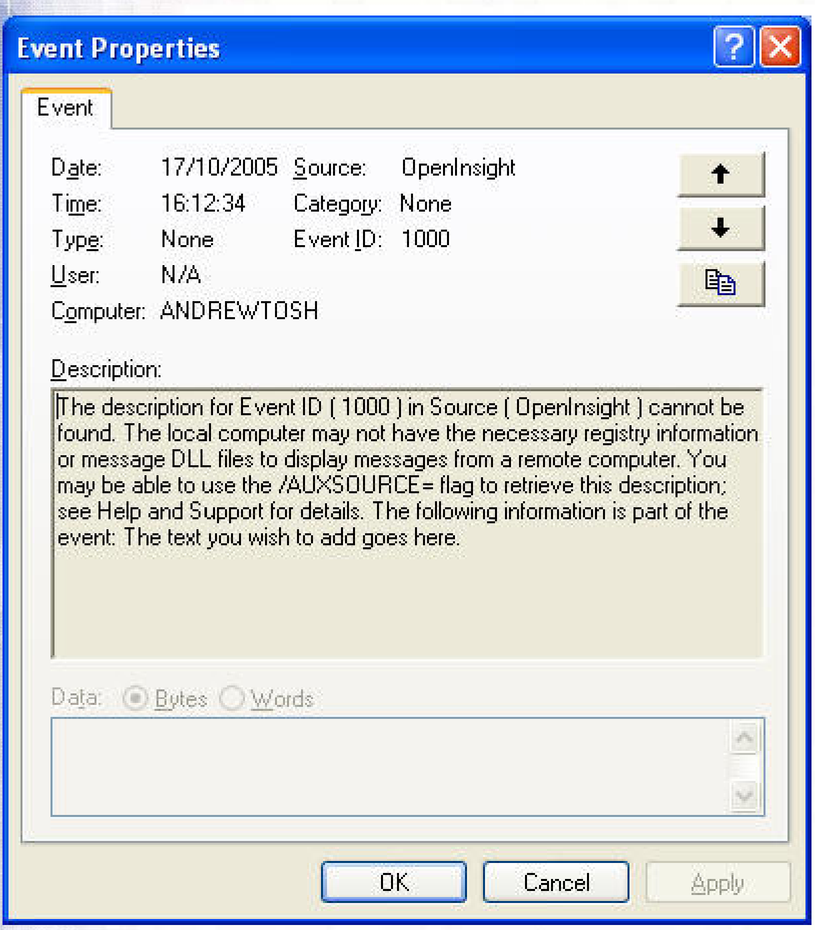
Line 10 out of 15
Col 42
Size 360

This results in an event log entry as follows :-



Application 1 event(s)

Type	Date	Time	Source	Category	Event	User	Computer
Information	17/10/2005	16:12:34	OpenInsight	None	1000	N/A	ANDREWTO...



```
Date:      17/10/2005  Source:    OpenInsight
Time:      16:12:34    Category:  None
Type:      None        Event ID:  1000
User:      N/A
Computer:  ANDREWTOSH
```

Description:

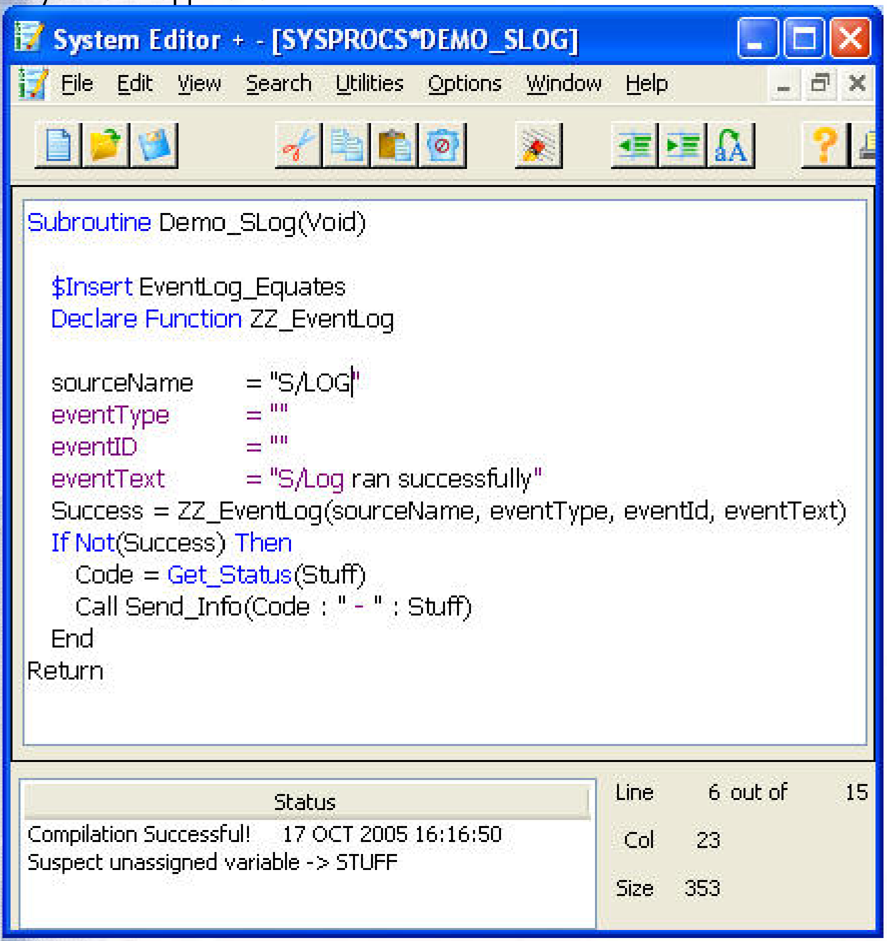
```
The description for Event ID ( 1000 ) in Source ( OpenInsight ) cannot be
found. The local computer may not have the necessary registry information
or message DLL files to display messages from a remote computer. You
may be able to use the /AUXSOURCE= flag to retrieve this description;
see Help and Support for details. The following information is part of the
event: The text you wish to add goes here.
```

Data: [✓ Bytes] Words

SPREZZATURA

SourceName

This is the source name to put into the event log. The default is "OpenInsight" but you can change this to be your own application.

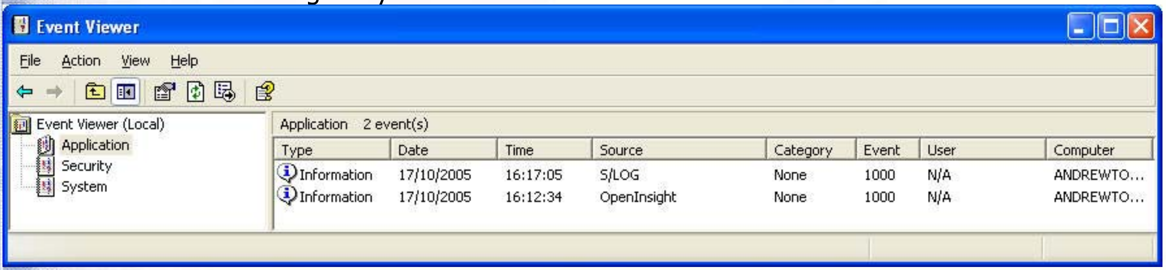

System Editor + - [SYSPROCS*DEMO_SLOG]
File Edit View Search Utilities Options Window Help

```
Subroutine Demo_SLog(Void)

  $Insert EventLog_Equates
  Declare Function ZZ_EventLog

  sourceName     = "S/LOG"
  eventType      = ""
  eventID        = ""
  eventText      = "S/Log ran successfully"
  Success = ZZ_EventLog(sourceName, eventType, eventId, eventText)
  If Not(Success) Then
    Code = Get_Status(Stuff)
    Call Send_Info(Code : " - " : Stuff)
  End
Return
```

```
Status
Compilation Successful!    17 OCT 2005 16:16:50
Suspect unassigned variable -> STUFF
```

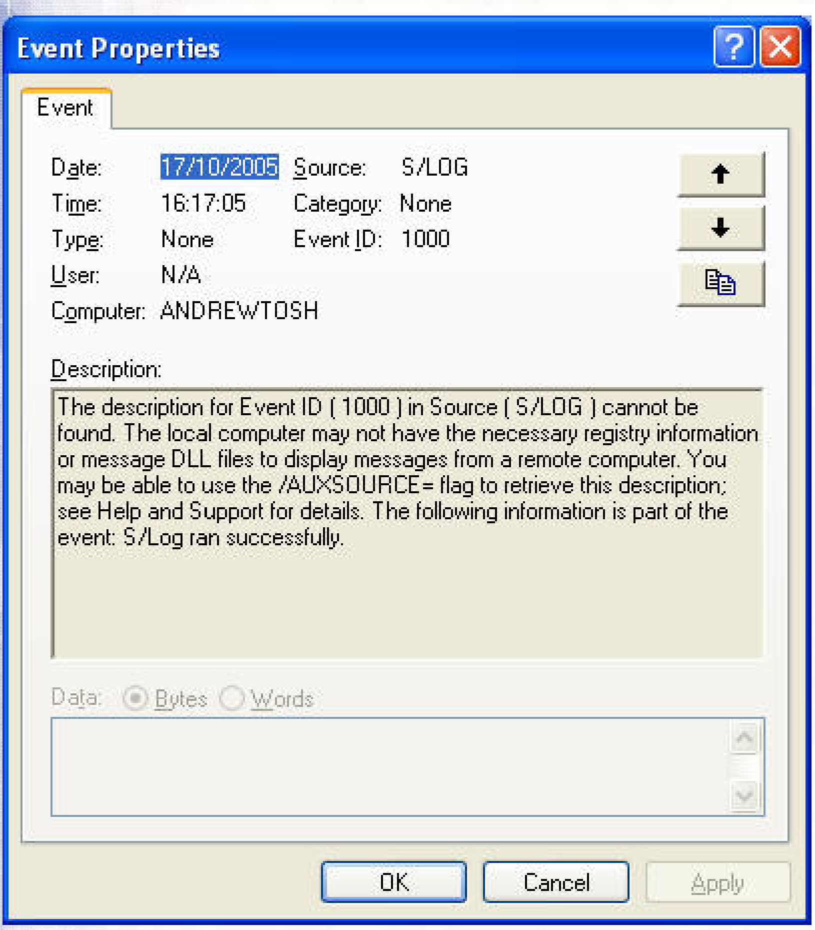
Line 6 out of 15
Col 23
Size 353

This results in an event log entry as follows :-



Application 2 event(s)

Type	Date	Time	Source	Category	Event	User	Computer
Information	17/10/2005	16:17:05	S/LOG	None	1000	N/A	ANDREWTO...
Information	17/10/2005	16:12:34	OpenInsight	None	1000	N/A	ANDREWTO...



Date: 17/10/2005 Source: S/LOG
Time: 16:17:05 Category: None
Type: None Event ID: 1000
User: N/A
Computer: ANDREWTOSH

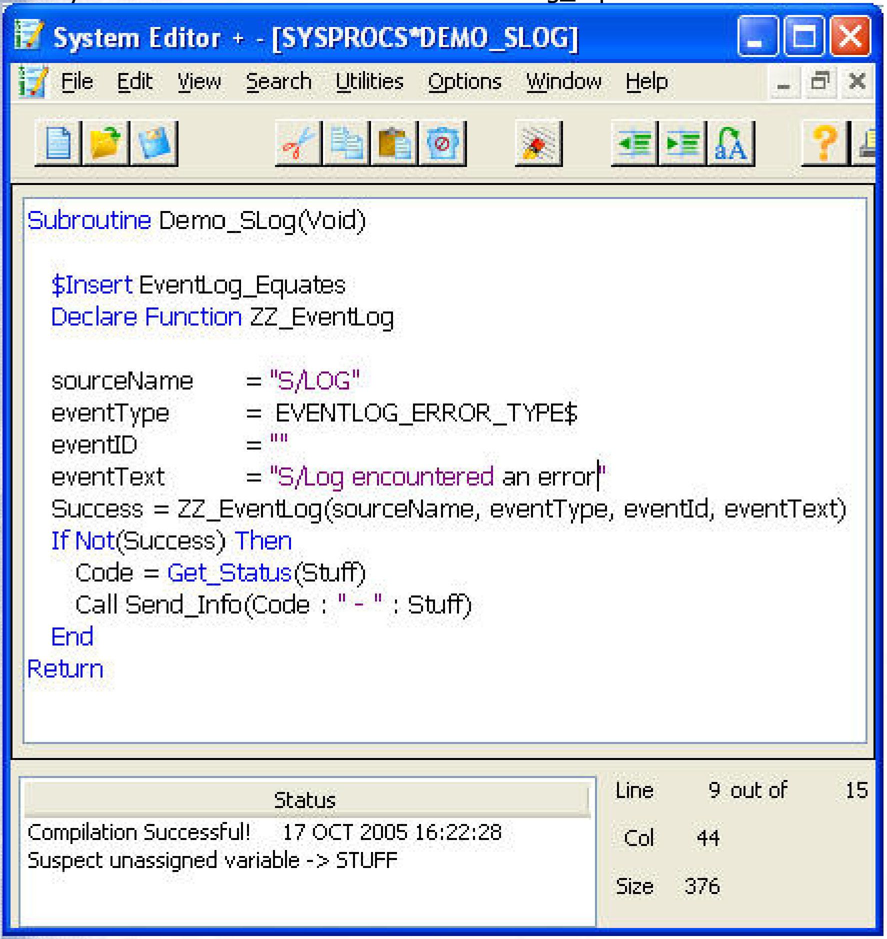
Description:

The description for Event ID (1000) in Source (S/LOG) cannot be found. The local computer may not have the necessary registry information or message DLL files to display messages from a remote computer. You may be able to use the /AUXSOURCE= flag to retrieve this description; see Help and Support for details. The following information is part of the event: S/Log ran successfully.

Data: [✓ Bytes] Words

EventType

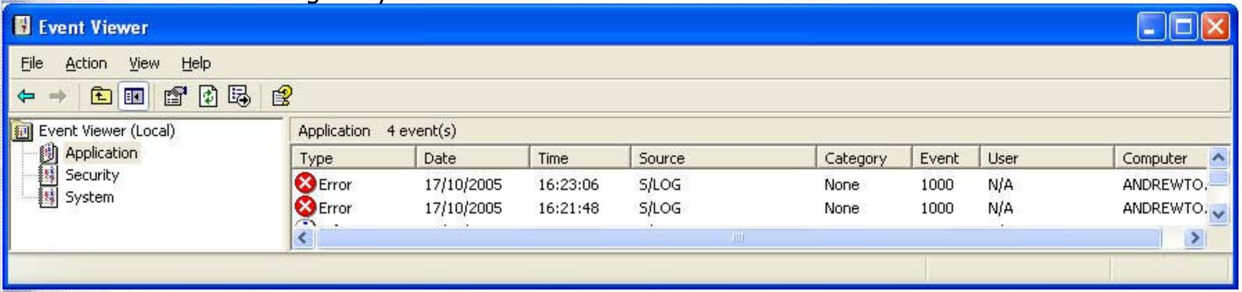
This is the event type to put into the event log. The default is None (Success) but you can change this to be any of the values in the insert row EventLog_Equates.


System Editor + - [SYSPROCS*DEMO_SLOG]
File Edit View Search Utilities Options Window Help

```
Subroutine Demo_SLog(Void)

  $Insert EventLog_Equates
  Declare Function ZZ_EventLog

  sourceName      = "S/LOG"
  eventType       =  EVENTLOG_ERROR_TYPE$
  eventID         = ""
  eventText       = "S/Log encountered an error"
  Success = ZZ_EventLog(sourceName, eventType, eventId, eventText)
  If Not(Success) Then
     Code = Get_Status(Stuff)
     Call Send_Info(Code : " - " : Stuff)
  End
Return
```

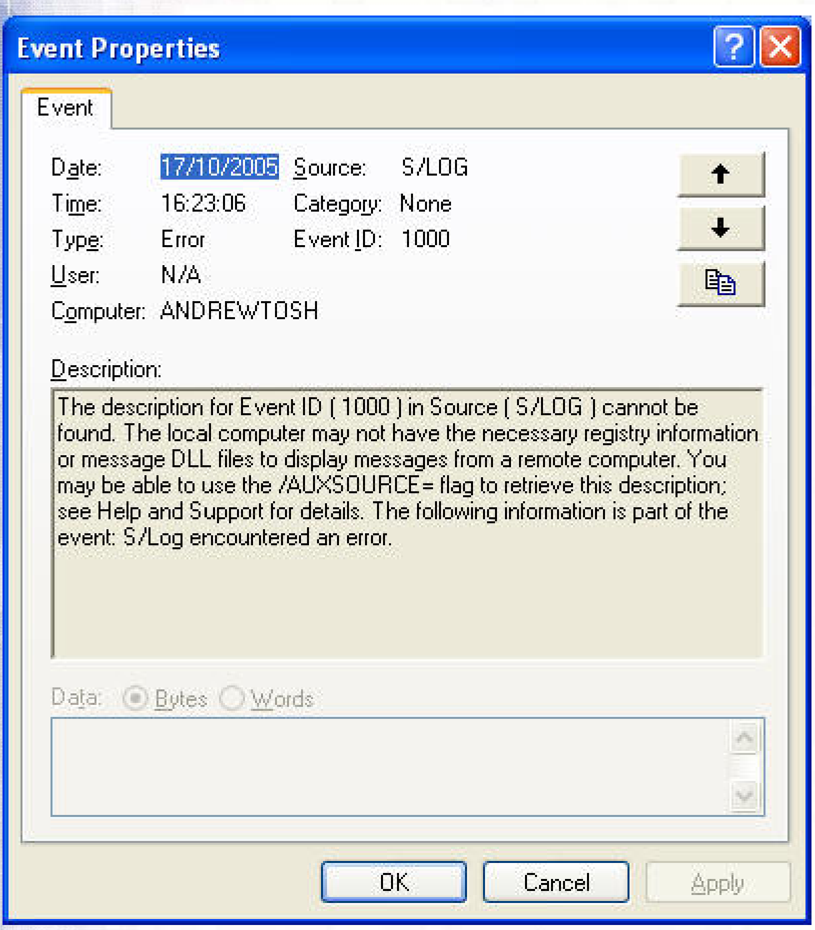
```
Status
Compilation Successful!    17 OCT 2005 16:22:28
Suspect unassigned variable -> STUFF

Line    9 out of    15
Col     44
Size    376
```

This results in an event log entry as follows :-



Application 4 event(s)

Type	Date	Time	Source	Category	Event	User	Computer
Error	17/10/2005	16:23:06	S/LOG	None	1000	N/A	ANDREWTO.
Error	17/10/2005	16:21:48	S/LOG	None	1000	N/A	ANDREWTO.



Date: 17/10/2005 Source: S/LOG
Time: 16:23:06 Category: None
Type: Error Event ID: 1000
User: N/A
Computer: ANDREWTOSH

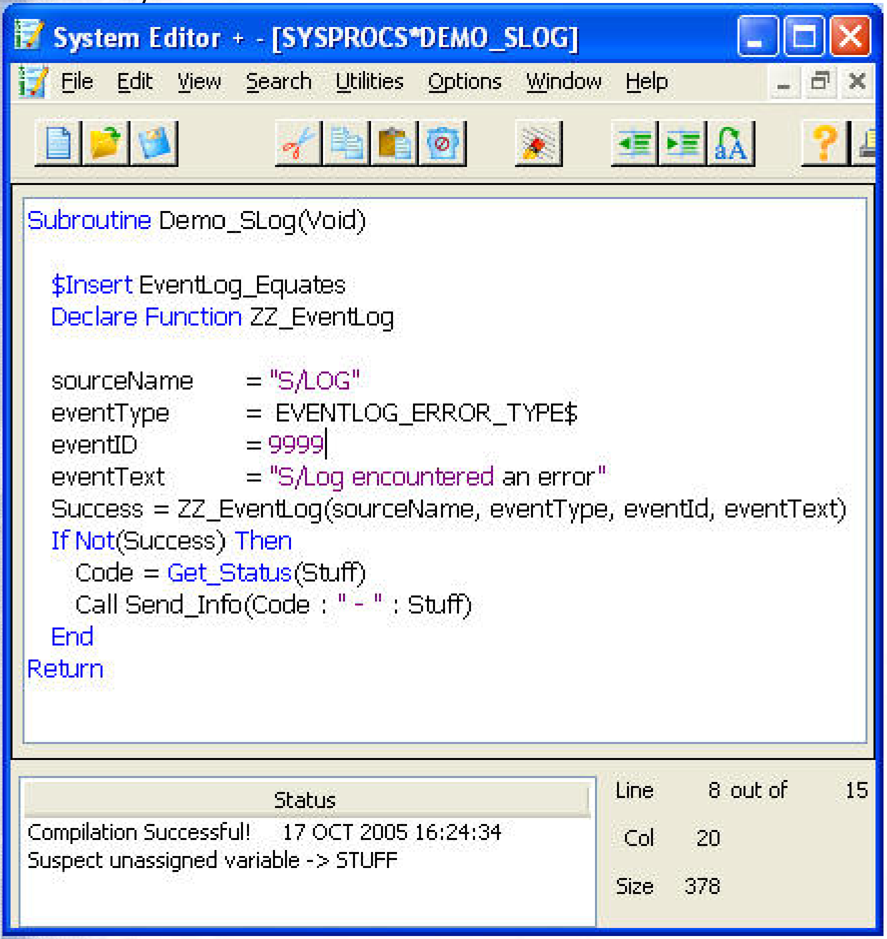
Description:
The description for Event ID (1000) in Source (S/LOG) cannot be found. The local computer may not have the necessary registry information or message DLL files to display messages from a remote computer. You may be able to use the /AUXSOURCE= flag to retrieve this description; see Help and Support for details. The following information is part of the event: S/Log encountered an error.

Data: Bytes Words

SPREZZATURA

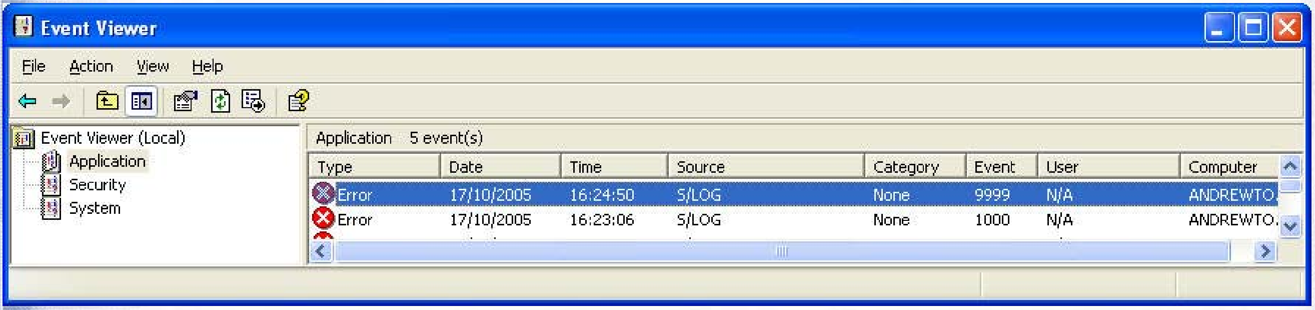
EventId

This is the event id to put into the event log. The default is 1000 but you can use whatever numbering convention you want.


System Editor + - [SYSPROCS*DEMO_SLOG]
File Edit View Search Utilities Options Window Help

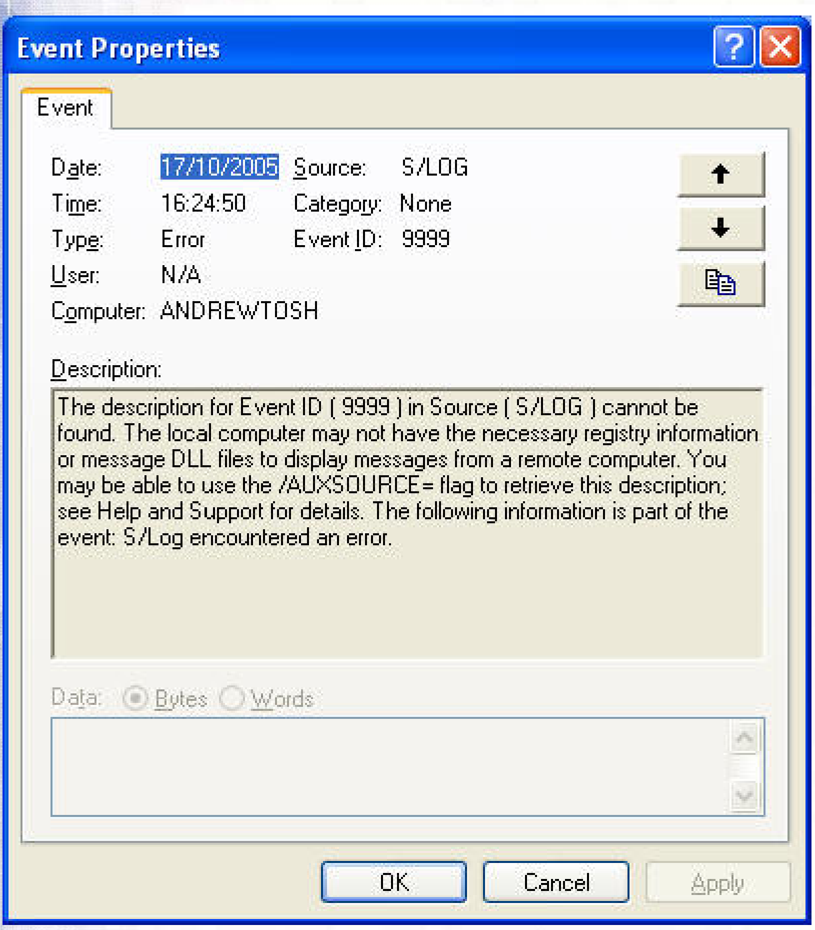
```
Subroutine Demo_SLog(Void)

  $Insert EventLog_Equates
  Declare Function ZZ_EventLog

  sourceName      = "S/LOG"
  eventType       =  EVENTLOG_ERROR_TYPE$
  eventID         = 9999
  eventText       = "S/Log encountered an error"
  Success = ZZ_EventLog(sourceName, eventType, eventId, eventText)
  If Not(Success) Then
     Code = Get_Status(Stuff)
     Call Send_Info(Code : " - " : Stuff)
  End
Return
```

```
Status
Compilation Successful!    17 OCT 2005 16:24:34
Suspect unassigned variable -> STUFF

Line    8 out of    15
Col    20
Size   378
```



Application 5 event(s)

```
Type    Date        Time      Source   Category   Event   User   Computer
Error   17/10/2005  16:24:50  S/LOG    None       9999    N/A    ANDREWTO...
Error   17/10/2005  16:23:06  S/LOG    None       1000    N/A    ANDREWTO...
```



Date: 17/10/2005 Source: S/LOG
Time: 16:24:50 Category: None
Type: Error Event ID: 9999
User: N/A
Computer: ANDREWTOSH

Description:

The description for Event ID (9999) in Source (S/LOG) cannot be found. The local computer may not have the necessary registry information or message DLL files to display messages from a remote computer. You may be able to use the /AUXSOURCE= flag to retrieve this description; see Help and Support for details. The following information is part of the event: S/Log encountered an error.

Data: [✓ Bytes] Words

Error Trapping

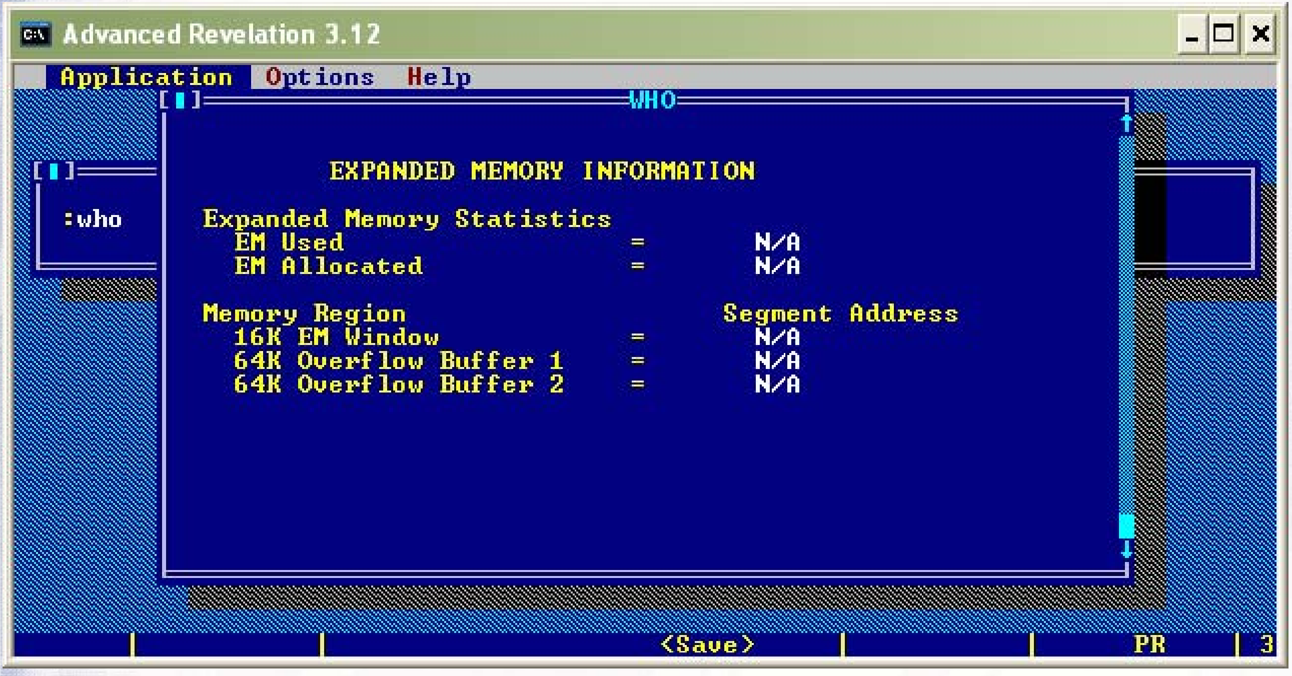
If S/Log fails it will return a zero. The developer can check for this and use Get_Status for a meaningful description of the error encountered.

SPREZZATURA

EMS Memory Under XP – Aaron Kaplan

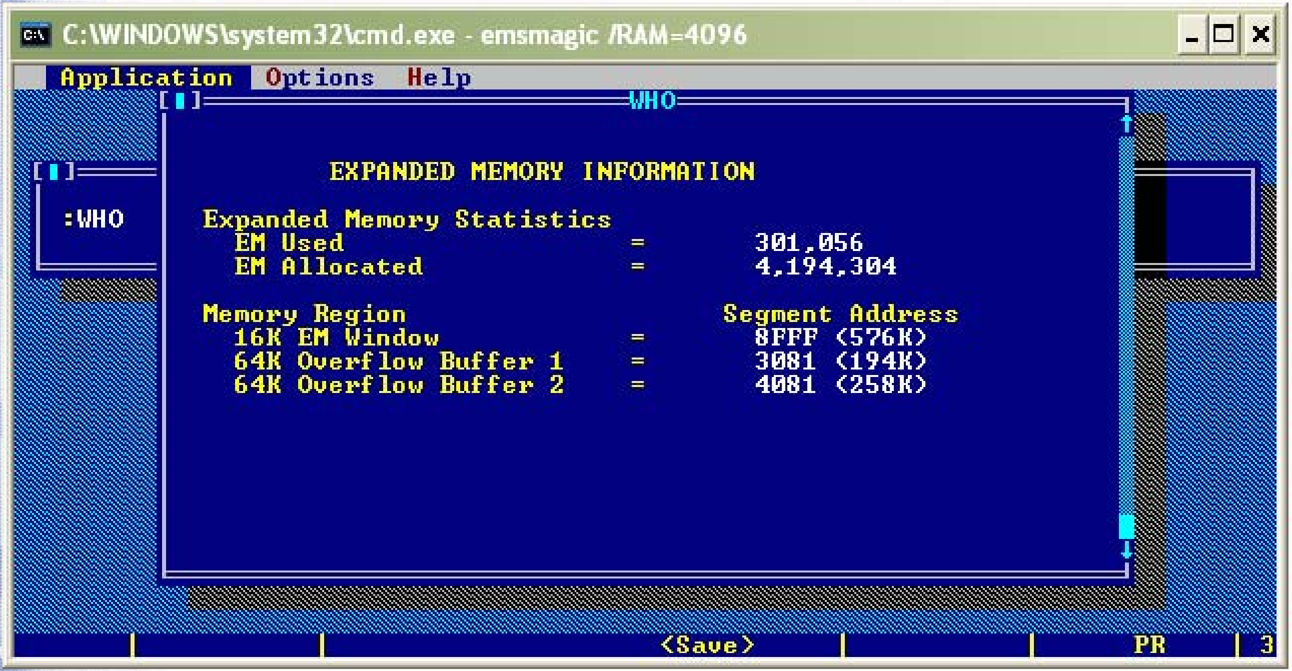
With all the advantages modern computing has given us, for those using Advanced Revelation, there is one disadvantage we have been forced to work with, the loss of EMS memory. However, Phat Code has released EMS Magic 1.0, a software EMS emulator. Using the Windows Installer, or just unzipping to a location in your path, gives all the EMS your AREV systems will need.

Before



```
EXPANDED MEMORY INFORMATION

Expanded Memory Statistics
  EM Used                    =      N/A
  EM Allocated               =      N/A

Memory Region                    Segment Address
  16K EM Window              =      N/A
  64K Overflow Buffer 1      =      N/A
  64K Overflow Buffer 2      =      N/A
```

After



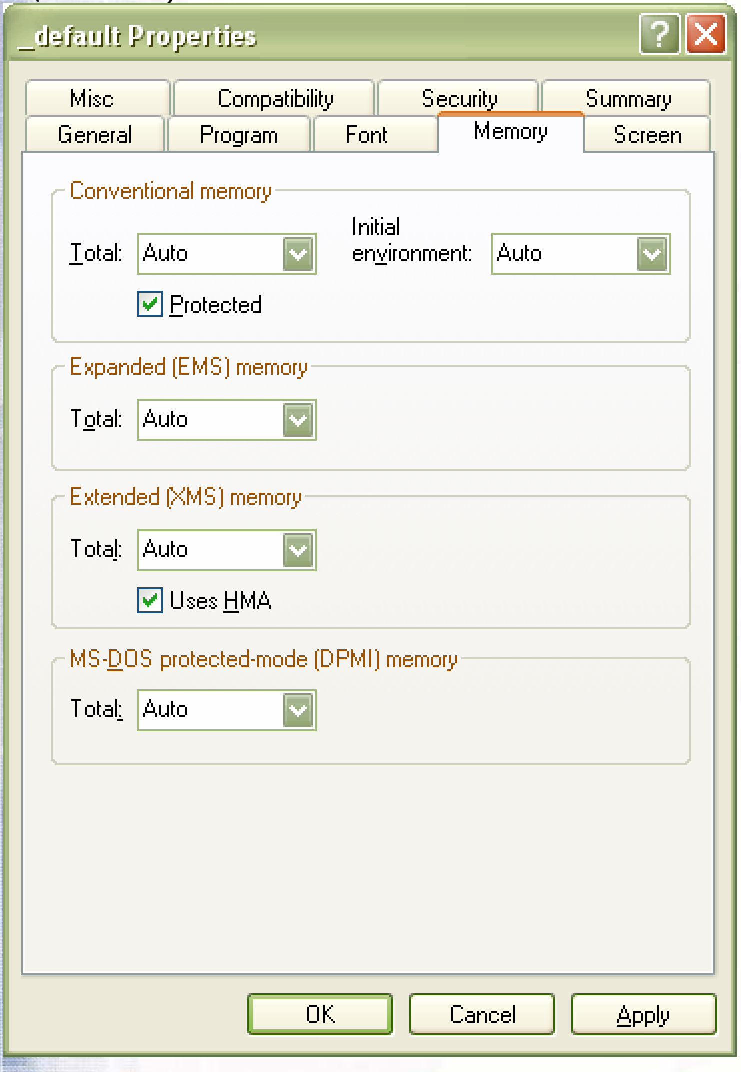
```
EXPANDED MEMORY INFORMATION

Expanded Memory Statistics
  EM Used                    =      301,056
  EM Allocated               =      4,194,304

Memory Region                    Segment Address
  16K EM Window              =      8FFF (576K)
  64K Overflow Buffer 1      =      3081 (194K)
  64K Overflow Buffer 2      =      4081 (258K)
```

To automate from the desktop, create a simple batch file:

```
EMSMAGIC /RAM=4096
AREV /X /M4096
```

Nothing could be simpler or extend the life of your AREV programs. It works with both the 2.1 NT Service and the Universal Driver 3.0.0.3 network products.

Because EMSMAGIC loads an instance of COMMAND.COM, you need to ensure that your _DEFAULT.PIF file has been correctly defined for EMS. _DEFAULT.PIF is loaded in the %WINDIR% directory (which is usually C:\WINDOWS)



Conventional memory
Total: Auto
Initial environment: Auto
☑ Protected

Expanded (EMS) memory
Total: Auto

Extended (XMS) memory
Total: Auto
☑ Uses HMA

MS-DOS protected-mode (DPMI) memory
Total: Auto

You can limit the EMS value to 4096, but only if you're sure other applications will not require memory in excess of that amount.

As much as we'd love to go on and on with technical tips, there's nothing more we can tell you that's not already in the documentation.

You can download EMS Magic from http://www.phatcode.net/projects.php?id=102.

S/OCKET – A commercial Utility – Andrew McAuley

One of the things we're frequently asked as consultants is "how do I access a socket from within OpenInsight?". Well there are routines built into OI to assist with this but as you know, here at Sprezz Towers we like our utilities to be completely under our control where possible. To address this issue we developed our own suite of routines, documentation for which follows.

S/Ocket

Sprezzatura's S/Ocket is a 32-bit DLL that enables a developer to include basic TCP/IP communications to a server from their 32-bit OpenInsight application/s and ARev and OI16-bit applications using an OI32 runtime.

Whereas the socket functionality in OpenInsight 7.1 is designed to handle high-level protocols such as FTP and HTTP, S/ocket is designed to provide more flexibility to the developer by allowing 'raw' data streams to be sent and received, thus granting the ability to implement custom and extended TCP protocol programming.

There are 8 functions that have been prototyped in OpenInsight and each is highlighted below to give you an idea of what can be achieved using S/Ocket.

- zzx_TCPConnect
- zzx_TCPClose
- zzx_TCPWrite
- zzx_TCPWriteLn
- zzx_TCPRead
- zzx_TCPReadLn
- zzx_TCPReadEx
- zzx_TCPReadLnEx

Definitions

zzx_TCPConnect

This function makes a connection to a server on a specified port and must be called before any further communications with the server can take place.

```
retVal = zzx_TCPConnect( lpDestination, lpPort )
```

Arguments

lpDestination	This is a null terminated string that specifies the host name (eg. 'www.sprezzatura.com') or IP Address (eg. '127.0.0.1') to connect to.
iPort	This is the number of the port to connect to on the host.

Returns
If the function is successful a handle to the connection is returned (this must be used in subsequent communications with the server). If the function fails for any reason then 0 is returned.

Example
```
declare function zzx_TCPConnect
lpDest  = 'www.sprezzatura.com' : \00\
iPort   = 80
hSocket = zzx_TCPConnect( lpDest, iPort )
```

SPREZZATURA

zzx_TCPClose

This subroutine is used to close the connection to a server previously opened with the zzx_TCPConnect() function.

```
call zzx_TCPClose( hSocket )
```

Arguments:

hSocket	Handle to an open connection (ie the value returned from zzx_TCPConnect).

Returns
N/A

Example

```
declare function zzx_TCPConnect
declare subroutine zzx_TCPClose
lpDest = 'www.sprezzatura.com' : \00\
iPort = 80
hSocket = zzx_TCPConnect( lpDest, iPort )
if hSocket then
      * // Do some comms stuff....
      call zzx_TCPClose( hSocket )
end
```

zzx_TCPWrite

This function is used to send data to a server.

```
retVal = zzx_TCPWrite( hSocket, lpBuffer, iBufferLen )
```

Arguments:

hSocket	Handle to an open connection (ie the value returned from zzx_TCPConnect).
lpBuffer	Variable containing the data to write.
iBufferLen	Number of bytes contained in lpBuffer. If this value is less than 0 then lpBuffer is assumed to be a null-terminated string and it's length calculated accordingly.

Returns
Returns 1 if successful, 0 otherwise.

Example

```
declare function zzx_TCPConnect, zzx_TCPWrite
declare subroutine zzx_TCPClose
lpDest  = 'www.sprezzatura.com' : \00\
iPort   = 80
hSocket = zzx_TCPConnect( lpDest, iPort )
if hSocket then
      * // send some data to the server
      lpBuffer   = 'Some data to send' : \00\
      iBufferLen = len( lpBuffer ) - 1
      retVal     = zzx_TCPWrite( hSocket, lpBuffer, iBufferLen )
      * // More processing ....
      call zzx_TCPClose( hSocket )
end
```

zzx_TCPWriteLn

This function is used to send data to a server but an EOL string is appended to the data before it is sent (ie... Char(13) and Char(10))

```
retVal = zzx_TCPWriteLn( hSocket, lpBuffer, iBufferLen )
```

Arguments:

hSocket	Handle to an open connection (ie the value returned from zzx_TCPConnect).
lpBuffer	Variable containing the data to write.
iBufferLen	Number of bytes contained in lpBuffer. If this value is less than 0 then lpBuffer is assumed to be a null-terminated string and it's length calculated accordingly.

Returns
Returns 1 if successful, 0 otherwise.

Example

```
declare function zzx_TCPConnect, zzx_TCPWriteLn
declare subroutine zzx_TCPClose
lpDest  = 'www.sprezzatura.com' : \00\
iPort   = 80
hSocket = zzx_TCPConnect( lpDest, iPort )
if hSocket then
      * // send some data to the server
      lpBuffer   = 'Some data to send' : \00\
      iBufferLen = len( lpBuffer ) - 1
      retVal     = zzx_TCPWriteLn( hSocket, lpBuffer, iBufferLen )
      * // More processing ....
      call zzx_TCPClose( hSocket )
end
```

zzx_TCPRead

This function is used to read data from a server.

```
retVal = zzx_TCPRead( hSocket, lpBuffer, iBufferLen )
```

Arguments:

hSocket	Handle to an open connection (ie the value returned from zzx_TCPConnect).
lpBuffer	Variable to read the data from the server into.
iBufferLen	Number of bytes contained in lpBuffer.

Returns

Returns 1 if successful, 0 otherwise.

Example

```
declare function zzx_TCPConnect, zzx_TCPWrite, zzx_TCPRead
declare subroutine zzx_TCPClose
lpDest  = 'www.sprezzatura.com' : \00\
iPort   = 80
hSocket = zzx_TCPConnect( lpDest, iPort )
if hSocket then
      * // send some data to the server
      lpBuffer   = 'Some data to send' : \00\
      iBufferLen = len( lpBuffer ) - 1
      retVal     = zzx_TCPWrite( hSocket, lpBuffer, iBufferLen )
      if retVal then
            * // Get some data from the server
            lpBuffer   = str( \00\, 2048 )
            iBufferLen = len( lpBuffer )
            retVal     = zzx_TCPRead( hSocket, lpBuffer, iBufferLen )
      end
      call zzx_TCPClose( hSocket )
end
```

zzx_TCPReadLn

This function is used to read a line of data from a server. By default lines are delimited by Char(10) but it is possible to specify a different delimiter.

```
retVal = zzx_TCPReadLn( hSocket, lpBuffer, iBufferLen, lpDelim )
```

Arguments:

hSocket	Handle to an open connection (ie the value returned from zzx_TCPConnect).
lpBuffer	Variable to read the data from the server into. This should be large enough to accept the longest line that can be returned from the server.
iBufferLen	Number of bytes contained in lpBuffer.
lpDelim	null-terminated string specifying the line delimiter. To use the default delimiter (char(10)) here pass a char(0) in this argument.

Returns
Returns 1 if successful, 0 otherwise.

Example

```
declare function zzx_TCPConnect, zzx_TCPWrite, zzx_TCPReadLn
declare subroutine zzx_TCPClose
lpDest  = 'www.sprezzatura.com' : \00\
iPort   = 80
hSocket = zzx_TCPConnect( lpDest, iPort )
if hSocket then
      * // send some data to the server
      lpBuffer   = 'Some data to send' : \00\
      iBufferLen = len( lpBuffer ) - 1
      retVal     = zzx_TCPWrite( hSocket, lpBuffer, iBufferLen )
      if retVal then
            * // Get a line of data from the server using the default
            * // delimiter. Note that we are not expecting any line
            * // here to be longer than 2048 chars.
            lpBuffer   = str( \00\, 2048 )
            iBufferLen = len( lpBuffer )
            lpDelim    = \00\
            retVal     = zzx_TCPReadLn( hSocket, lpBuffer, iBufferLen,
            lpDelim )
      end
      call zzx_TCPClose( hSocket )
end
```

zzx_TCPReadEx

This function is used to read data from a server.

retVal = zzx_TCPReadEx(hSocket, lpBuffer, iBufferLen,
iReadTimeout, lpErrorBuffer, iErrorBufferLen)

Arguments:

hSocket	Handle to an open connection (ie the value returned from zzx_TCPConnect).
lpBuffer	Variable to read the data from the server into.
iBufferLen	Number of bytes contained in lpBuffer.
iReadTimeout	Number of milliseconds that the connection should wait for the peer connection to become readable using the protocol stack. Defaults to infinite (<=0).
lpErrorBuffer	Pointer to a variable to receive any error information.
iErrorBufferLen	Size of lpErrorBuffer in bytes.

Returns
Returns 1 if successful, 0 otherwise.

Example

```
declare function zzx_TCPConnect, zzx_TCPWrite, zzx_TCPReadEx
declare subroutine zzx_TCPClose
lpDest  = 'www.sprezzatura.com' : \00\
iPort   = 80
hSocket = zzx_TCPConnect( lpDest, iPort )
if hSocket then
      * // send some data to the server
      lpBuffer   = 'Some data to send' : \00\
      iBufferLen = len( lpBuffer ) - 1
      retVal     = zzx_TCPWrite( hSocket, lpBuffer, iBufferLen )
      if retVal then
            * // Get some data from the server
            lpBuffer      = str( \00\, 2048 )
            iBufferLen    = len( lpBuffer )
            lpErrorText   = str( \00\, 2048 )
            iErrorTextLen = len( lpErrorText )
            iTimeOut      = 0
            retVal        = zzx_TCPReadEx( hSocket, lpBuffer, iBufferLen,
                              iTimeout, lpErrorText, iErrorTextLen )
      end
      call zzx_TCPClose( hSocket )
end
```

zzx_TCPReadLnEx

This function is used to read a line of data from a server. By default lines are delimited by Char(10) but it is possible to specify a different delimiter.

```
retVal = zzx_TCPReadLn( hSocket, lpBuffer, iBufferLen, lpDelim )
```

Arguments:

hSocket	Handle to an open connection (ie the value returned from zzx_TCPConnect)
lpBuffer	Variable to read the data from the server into. This should be large enough to accept the longest line that can be returned from the server.
iBufferLen	Number of bytes contained in lpBuffer.
lpDelim	A null-terminated string specifying the line delimiter. To use the default delimiter (char(10)) here pass a char(0) in this argument.
iReadTimeout	Number of milliseconds that the connection should wait for the peer connection to become readable using the protocol stack. Defaults to infinite (<=0).
lpErrorBuffer	Pointer to a variable to receive any error information.
iErrorBufferLen	Size of lpErrorBuffer in bytes.

Returns
Returns 1 if successful, 0 otherwise.

Example

```
declare function zzx_TCPConnect, zzx_TCPWrite, zzx_TCPReadLnEx
declare subroutine zzx_TCPClose
lpDest  = 'www.sprezzatura.com' : \00\
iPort   = 80
hSocket = zzx_TCPConnect( lpDest, iPort )
if hSocket then
      * // send some data to the server
      lpBuffer   = 'Some data to send' : \00\
      iBufferLen = len( lpBuffer ) - 1
      retVal     = zzx_TCPWrite( hSocket, lpBuffer, iBufferLen )
      if retVal then
            * // Get a line of data from the server using the default
            * // delimiter. Note that we are not expecting any line here
            * // to be longer than 2048 chars.
            lpBuffer      = str( \00\, 2048 )
            iBufferLen    = len( lpBuffer )
            lpDelim       = \00\
            lpErrorText   = str( \00\, 2048 )
            iErrorTextLen = len( lpErrorText )
            iTimeOut      = 0
            retVal        = zzx_TCPReadLnEx( hSocket, lpBuffer,
                              iBufferLen, lpDelim, iTimeout, lpErrorText,
                              iErrorTextLen  )
      end
      call zzx_TCPClose( hSocket )
end
```

Ordering your copy of S/Ocket

S/Ocket is not designed as an end user tool and is priced accordingly. S/Ocket is licensed per developer copy of 32-bit OpenInsight. Thus once a developer has legitimately acquired the software they may incorporate it into all of their delivered applications but they may not redistribute it to other developers or to different developer serial numbers - just email Sales with your order number, invoice details, OpenInsight serial number and OpenInsight version and we'll send you a copy of the product by email.

ARev Users - We have also not forgotten our ARev developers. S/Ocket has been designed in such as way that it can easily be used as a runtime OpenInsight system working alongside any ARev 3.12 system and sharing the common data tables. However, if you are using custom MFSs then some additional setup may be required for S/Ocket to respect the MFS. If you wish to use S/Ocket with ARev please note that you will need to acquire the appropriate SDP through your usual Revelation Software product supplier or your local Sprezzatura office.

Prices are as follows:-

Product	USA	UK	Rest of world
S/Ocket for OI	$1,600.00 USD	£850.00 GBP	£850.00 GBP

Please note:

- prices do not include any applicable local Government taxes.
- each copy of "S/Socket" is licensed to one OpenInsight 32-bit serial number only.

Use of Hungarian Notation in Untyped languages – Aaron Kaplan

As mentioned earlier in SENL, we were going to talk about how Hungarian notation and how this can work with Basic+ and other untyped languages. If you ask most programmers about Hungarian notation and what it means, they'll probably tell you that it means prefixing the variable name with an indicator of its type. For example, if we're storing the number of times a patient has seen a particular doctor, we might have a variable called intSeenDoctorCount. This means that everyone looking at this code will know that this variable is of type integer.

In an untyped language, this method is always unhelpful. Everything would be charMyVar or undefMyVar, rendering the entire notation useless. Alternatively, we could define the variables with the type expected to be used, and that's a matter of developer preference or company standards. We know it doesn't actually define the type, just how we expect to use the character strings.

What many people don't know is that Hungarian notation is actually two different notational systems. The one described above, the one most people mean, is called Systems Hungarian notation. There is another form, Apps Hungarian notation, and that variant is much more compatible with untyped languages such as Basic+.

In Apps Hungarian notation, the variable is prefixed with how it is expected to be used and what context the variable is used in the real world view of the code. For example, if working with graphics packages, you could have a variables pxLineSize and twipLineSize, which indicate that the line size is being stored in either pixels or twips.

By using Apps Hungarian notion, we can easily see if our variables are internal or external representation, the units the variable is holding, and perhaps even the location the data was obtained from. By prefixing your variables with ic or oc, you can easily tell the format of the variable. With that, you'd never make the mistake of a line of code

```
@RECORD< INVOICE_DATE$ > = ocInvoiceDate
```

You'd know that the ocTotalAmt has been formatted externally and should probably be converted to it's internal format.

Another example is if you are working with patient names, by saying patPatientName or examPatientName, you'd know which file the patient name has come from. This would be important when accessing historical data if the patient's name had changed.

As we're sure you can see, Apps Hungarian notation has a useful role to place in Basic+ programming.

SPREZZATURA

Peripheral Trivia

As this issue of SENL was put to bed we fed the inner man with:

TV :	Heroes
Book:	How to break Web Software – Andrews and Whittaker
CD:	The Evolution Control Committee – "Plagiarythm Nation V2.0"
WEB:	http://www.kayak.com

Join us :	Send Mail to Admin@Sprezzatura.com with subject SUBSCRIBE SENL
Leave Us:	Send Mail to Admin@Sprezzatura.com with subject UNSUBSCRIBE SENL
Change of Address:	Leave at the old address & join at the new one
Web Info:	http://www.sprezzatura.com/
Tell us what you'd like to see in SENL:	info@sprezzatura.com