intl-spectrum.com

INTERNATIONAL SPECTRUM®

THE MULTIVALUE TECHNOLOGY MAGAZINE | JANUARY/FEBRUARY 2018

Spotlight

4GLs

Also In This Issue:

- Using MultiValue with ASP.NET OWIN
- UI/UX: Final Thoughts
- Compression, Transmission, and Python – Oh, my!

INTERNATIONAL SPECTRUM

37TH ANNUAL MULTIVALUE CONFERENCE

DESTIN
FLORIDA

The Premier MultiValue User Event

APRIL 23 – 26, 2018

THE HENDERSON

DESTIN, FLORIDA

www.intl-spectrum.com/conference

The **International Spectrum MultiValue Conference** is the only place that provides diverse solutions and support for integrating and developing your Enterprise applications into the world of business and technology.

This annual event is the place to find the **knowledge**, **people**, **resources**, and **technologies** for enhancing your MultiValue (D3, UniVerse, UniData, Reality, jBase, Caché, QM, and OpenInsight) software applications.

Register Now!

INTERNATIONAL SPECTRUM
THE MULTIVALUE TECHNOLOGY MAGAZINE

JANUARY/FEBRUARY 2018

CONTENTS

COVER

FEATURES

DEPARTMENTS

From the Inside

Spectre and Meltdown

By now, you have probably heard all about the major bugs in Intel, ARM, and AMD processors. Since this vulnerability affects just about all processor types, you will probably need to update your phone, your tablet, your PC, and all of your servers in the coming weeks. Why and how does it affect everything?

What is this Bug About?

I hope everyone has been trying to keep up with the details of this bug, but in case you are not completely sure what it is, I'll cover the key points.

Modern (since the early 2000s) processors use what's known as "out-of-order" execution. This is somewhat similar to the way databases will read-ahead on the disk to improve performance. The CPU will execute a series of instructions before the first one has been completed.

Unfortunately, an attacker can pass code that will fail and, using some trickery and the magic of caches, build and steal (at up to 500 kb/s) all of the kernel memory. This means things like passwords, that are normally secured in kernel memory, can be stolen very easily by an attacker.

Should I Patch?

YES!!!!

This is a deadly serious bug, that is easy to exploit. Patching the problem has been more complex than originally thought. Make sure you go through your normal test cycle with patch validation. None of the patches provided have gone through major testing so there may be issues that crop up over the next year.

Here is a good comprehensive listing of patches, provided by Allan Hirt of SQLHA. Yes, I know, it's an SQL site, but he did a really good job putting together the most comprehensive reference I've seen so far.

http://sqlha.com/2018/01/04/no-good-terrible-processor-flaw-sql-server-deployments-nearly-everything-need-know/

Will this affect my VMWare guest?

YES!!!

Actually, this bug means that it can be used to read information across guest boundaries. So if one guest is running code using this exploit, its able to access information that another guest was using.

Will This Impact My Performance?

Probably YES!

We are only just now figuring out how big of a hit we will take on speed of execution. I've talked to the MultiValue Database providers to see what their thoughts are. As of this writing, they are still evaluating. It really depends on the file types that are being used, and how your database

INTERNATIONAL SPECTRUM
JANUARY/FEBRUARY 2018

NATHAN RECTOR
President

CHARLES BAROUCH
Editor

TRACEY RECTOR
Layout

Learn more about the MultiValue Symbol and see what MulitValue Technologies and MultiValue Communities exist to help you support and manage your business and systems. To find out more visit

http://www.intl-spectrum.com

MISSION STATEMENT *International Spectrum* magazine's editorial mission is to be the premier independent source of useful information for users, developers, and resellers of MultiValue database management systems, open systems business database solutions, and related hardware, software, and peripherals. Published bimonthly, International Spectrum provides comprehensive coverage of the products, companies, and trends that shape the MultiValue marketplace as well as the computer industry at large — helping its readers get the most out of their business computer systems.

International Spectrum is published six (6) times per year at the subscription price of $40.00 U.S. in the U.S.A.; $45.00 U.S. in Canada and Mexico; $50.00 U.S. for other countries. Single copy rates are $7.00 U.S. in the U.S.A. and Canada, and $9.00 U.S. in all other countries. International Spectrum is published by International Specturm, Inc., 3691 E. 102nd Ct., Thornton, CO 80229; Tel: 720/259-1356; Fax: 603/250-0664 E-Mail: request@intl-spectrum.com.

PRINTED IN USA

NEWS RELEASES/UNSOLICITED ARTICLES
International Spectrum is eager to print your submissions of up-to- the-minute news and feature stories complementary to the MultiValue marketplace. Black and white or color photographs and diagrams are welcome. Although there is no guarantee a submitted article will be published, every article will be considered. Please send your press releases, articles, and queries to: editor@intl-spectrum.com. International Spectrum retains all reprint rights.

is setup to cache frames and groups in memory. Check with your database providers to find out your options.

If you are running on virtual hardware, then there is a large chance that you will see more of an impact than databases running on bare metal.

In order to help with performance, Microsoft is offering a registry option to not include the microcode fixes. The advantage being that the patch would become optional, but in the longer term, if you are audited, you may be out of compliance.

All this will get better over time as software patches are released and databases are altered to run fewer kernel calls. (This is something that jBase is currently doing with their Dynamic Files).

Ultimately, the CPU chips will be fixed. Intel has already committed to a fixed chip set by the end of the year. Personally I would wait until they have had enough of these chips running before upgrading since I believe this will be a bigger issues to resolve than they are letting on. We have already seen lots of "rush job" patches come out of Intel, which they withdrew within 24 hours of release.

You should also read more here: https://joeydantoni.com/2018/01/04/spectre-and-meltdown-what-does-this-mean-for-your-sql-servers/

NATHAN RECTOR
President
International Spectrum
nathan@intl-spectrum.com

Spotlight 4GL

BY CHARLES BAROUCH

I can manage your company's books if I have enough chalk and sidewalk space. Just because I can, doesn't make it a good idea. Likewise, I can write and manage complex software with just the things I can write myself. When you get tired of creating your own tools for everything — all the time, every time — you might want to step up to a Fourth Generation Language (4GL).

When we write code a line at a time, we are using a Third Generation Language (3GL). Examples include: mvBASIC, everything in the C-family, Python, and PHP. The natural step-up from a 3GL is a framework. jQuery for JavaScript and and JSTL for Java are examples of frameworks. MultiValue environments are, arguably, also frameworks because they supply built-in tools for interacting with your data. While adding a framework is a powerful approach to accelerating programming speed and expanding your toolkit, you may want to go even further and advance to a 4GL.

> *...libraries, sets of objects, a method for describing a program without resorting to hand coding, surgically placed areas where hand-code can be added, and an IDE to guide you through.*

Unlike a framework, a 4GL usually includes a combination of code and an Integrated Design Environment (IDE). That means that while a framework connects your code to existing code, the 4GL also helps you write code.

Stick Language

I used to work with an amazing developer named Roy Gordon. Back in the '80s, he developed a programming language called Stick Language which compiled into mvBASIC. To write a program in Stick Language required mastery of fifteen hundred parameters. Obviously, nobody — other than me and Roy — was going to do that. The rest of the team used a 4GL, also developed by Roy, which allowed you to drag-and-drop fields, and fill in a few details, to generate the code without writing any.

Stick Language worked, in part, because we were only trying to build one type of software. Developing a more generalized 4GL usually involves providing hooks for 3GL code in specific place. As we roll through the various software offerings in this series, you'll get some clear examples of how 3GL and 4GL intertwine.

IDE, Do You DE?

To understand what an IDE is, I'll give you an example of a fairly robust — but still entry-level — IDE: Notepad++. If you've ever used it, you'll be aware that it has language-specific templates which provide color-coding, indentation guides, and code-completion. While it does not come with an mvBASIC option, there are a good ones available, including two here: http://www.pickwiki.com/index.php/NotepadPlusPlus.

If you've used Eclipse, you might want to check this out: http://www.u2logic.com/tools.html. And there's also a stand-alone IDE here: http://www.mvdeveloper.com/pages/mvdeveloper.htm.

Notepad++ does fail the test of a truly robust IDE in one important way. A full IDE would also include an easy method for parametric entry.

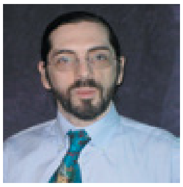
I Strongly Object

4GLs don't have to be object-oriented, but that's the way to bet. Just as the 4GL is a natural evolution of a framework, sets of objects are the natural evolution of libraries. Where a library is a collection of routines, an object is a collection of routines plus a context. For example, in a language like Delphi or C# you might find an object which encapsulates all the routines and graphical aspects of a button, combining the code which manage the state of the object, the events it triggers, its graphical appearance, and the graphical changes it can have.

Parts > Whole

A modern 4GL is going to provide some level of all of these things: libraries, sets of objects, a method for describing a program without resorting to hand coding, surgically-placed areas where hand-code can be added, and an IDE to guide you through.

The rest of the articles in this series will introduce you to how all of those things are presented, and balanced, in software like OpenInsight, Evoke, Nucleus, and others. As they say in Australia: *Horses for courses.* Different horses run better and worse on different tracks. Different design paradigms work better and worse for different projects. This isn't about picking *the best one;* it's about picking *the best one for your particular needs.* **IS**



CHARLES BAROUCH is the CTO of HDWP, Inc. and the Publisher at HDWPbooks. You can read his writing in International Spectrum, Theme-Thology, Novo Pulp, Pax Solaria, PerehelionSF, and the Interrogative series, which begins with Tiago and the Masterless.

BY NATHAN RECTOR

Using OWIN Security with MultiValue Data Part 1

OWIN is the latest framework that .NET developers are using to build ASP.NET applications. OWIN's Identity framework is an evolution of the ASP.NET 2.0 membership system which has been around for several years. It provides better flexibility than the older system, and decouples the Identity framework from IIS.

The intent of OWIN is to create a security framework that can work with ASP.NET applications, even if they aren't running on IIS. This is good news for people developing in Mono for Linux or those creating self-hosting applications.

At its core, OWIN is a middleware model to handle the separation of ASP.NET applications from the actual hosting environment. I will focus mainly on the security and membership side of things but it does much more.

If you are using ASP.NET MVC 6 for your web framework, you will have seen that OWIN is referenced automatically. By default, MVC will assume you are using the traditional ASP.NET identity model, but OWIN's identity model can be extended easily to include access to other identity systems.

> *This is good news for people developing in Mono for Linux or those creating self-hosting applications.*

When starting a new project, you will see something like the code in *Figure 1* the Startup.Auth class. This code is boilerplate, and shows you some of what you can do with the OWIN security. A couple things to point out, the Startup.auth class already includes the built-in providers to handle all the common Social Media logins like Microsoft, Twitter, Facebook and Google. When creating customer-based websites, this is a really nice feature. It also has the code in place to handle Two-Factor logins, so you don't need to build this yourself any more.

Another important thing to note is there are references to ApplicationDbContext, ApplicationUserManager, and ApplicationSignInManager in the Startup.Auth class. These classes are instrumental in making the identity system work. Let's look further into these pieces.

ApplicationDbContext

We'll start with the ApplicationDbContext. All this class does is handle the connections to the database and make the stored-procedure (BASIC subroutine) calls. Each of the MultiValue databases have different ways to do this, so the code you will see in <*Figure 2*> is very generic.

This will be the hardest part of any of the coding you will see in these examples, mainly because you have to know how to establish a connection to your MultiValue database with the .NET API offered by your database provider. If you need more examples, you can find them at http://www.intl-spectrum.com/resource/category/44/NET.aspx.

Another issue you may run into is the fact that OWIN is based on the .NET ASYNC/WAIT/Tasks model. This can affect how you code the connection to your MultiValue Database, since many of the APIs don't natively support Async/Wait methods.

```
Public Sub ConfigureAuth(app As IAppBuilder)
    ‘ Configure the db context, user manager and signin manager to use a single instance per
request
    app.CreatePerOwinContext(AddressOf ApplicationDbContext.Create)
    app.CreatePerOwinContext(Of ApplicationUserManager)(AddressOf ApplicationUserManager.
Create)
        app.CreatePerOwinContext(Of ApplicationSignInManager)(AddressOf
ApplicationSignInManager.Create)

    ‘ Enable the application to use a cookie to store information for the signed in user
    ‘ and to use a cookie to temporarily store inforation about a user logging in with a third
party login provider
    ‘ Configure the sign in cookie
    ‘ OnValidateIdentity enables the application to validate the security stamp when the user
logs in.
    ‘ This is a security feature which is used when you change a password or add an external
login to your account.
    Dim _AuthenticationOption As New CookieAuthenticationOptions()
    _AuthenticationOption.LoginPath = New PathString(“/Account/Login”)
    _AuthenticationOption.AuthenticationType = DefaultAuthenticationTypes.ApplicationCookie
    _AuthenticationOption.Provider = New CookieAuthenticationProvider

     ‘ Place an expiration on the Login
    ‘_AuthenticationOption.Provider.OnValidateIdentity = SecurityStampValidator.
OnValidateIdentity(Of ApplicationUserManager, ApplicationUserStore)(TimeSpan.FromMinutes(30),
Function(manager, user)
    ‘
                  user.GenerateUserIdentityAsync(manager)
    ‘
              End Function)

    app.UseCookieAuthentication(_AuthenticationOption)
    app.UseExternalSignInCookie(DefaultAuthenticationTypes.ExternalCookie)

    ‘ Enables the application to temporarily store user information when they are verifying
the second factor in the two-factor authentication process.
    app.UseTwoFactorSignInCookie(DefaultAuthenticationTypes.TwoFactorCookie, TimeSpan.
FromMinutes(5))

    ‘ Enables the application to remember the second login verification factor such as phone
or email.
    ‘ Once you check this option, your second step of verification during the login process
will be remembered on the device where you logged in from.
    ‘ This is similar to the RememberMe option when you log in.
    app.UseTwoFactorRememberBrowserCookie(DefaultAuthenticationTypes.
TwoFactorRememberBrowserCookie)

    ‘ Uncomment the following lines to enable logging in with third party login providers

    ‘app.UseMicrosoftAccountAuthentication(
    ‘    clientId:=””,
    ‘    clientSecret:=””)

    ‘app.UseTwitterAuthentication(
    ‘   consumerKey:=””,
    ‘   consumerSecret:=””)


    ‘app.UseFacebookAuthentication(
    ‘   appId:=””,
    ‘   appSecret:=””)

    ‘app.UseGoogleAuthentication(New GoogleOAuth2AuthenticationOptions() With {
    ‘   .ClientId = “”,
    ‘   .ClientSecret = “”})

End Sub
```

Figure.1

```
Public Class ApplicationDBContext
    Implements IDisposable

    ‘’’ <summary>
    ‘’’ This is used by Startup.Auth to create single reference to the database context.
    ‘’’ </summary>
    Public Shared Function Create() As ApplicationDBContext
        Return New ApplicationDBContext()
    End Function

    Public Sub New()
        ConnectionName = “localhost”
    End Sub

    Public Class CallSubroutineResults
        Public Property RequestItem As New mvDynamicArray
        Public Property Data As New SpectrumRecordArray
        Public Property CallError As New mvDynamicArray
    End Class

    ‘’’ <summary>
    ‘’’ Makes a subroutine call using Async/Wait coding.
    ‘’’ </summary>
    ‘’’ <param name=”ProgName”></param>
    ‘’’ <param name=”RequestItem”></param>
    ‘’’ <param name=”UserItem”></param>
    ‘’’ <returns></returns>
    Public Function CallSubroutineAsync(ByVal ProgName As String, ByVal RequestItem As String, 
ByVal UserItem As String) As Task(Of CallSubroutineResults)
        ‘ Creates a task object that can be used with Await
        Return Task.Run(Of CallSubroutineResults)(Function()
                                                           ‘ Setup the results
                                                           Return CallSubroutine(ProgName, 
RequestItem, UserItem)
                                                       End Function)
    End Function

    Public Function CallSubroutine(ByVal ProgName As String, ByVal RequestItem As String, 
ByVal UserItem As String) As CallSubroutineResults
        ‘ SUBROUTINE PROG.NAME(APPLICAITON.ITEM,REQUEST.ITEM,DATA.ITEM,NONE1,NONE2,ERROR.ITEM)
        Dim _ApplicationItem As String = “ASP.NET MVC”
        Dim _ErrorItem As String = “0”
        Dim _CommunicationErrorMsg As String = String.Empty
        Dim _Results As New CallSubroutineResults

        ‘ Gets next available connection
        Dim _Connection As New mvDatabase
        _Connection.ConnectionSettings.Add(“HostName”, ConnectionName)

        ‘ Make sure we are connected before calling the subroutine
        If Not _Connection.IsConnected Then
            If Not _Connection.Connect() = mvDatabase.eConnectionStatus.Connected Then
                _Results.CallError.StringValue(1) = “999”
                _Results.CallError.StringValue(2) = “Connection Error”
                _Results.CallError.StringValue(3) = _Connection.LastException.ToString
                Return _Results
            End If
        End If

        ‘ Call the subroutine
        Try
            _Connection.CallHostProg(ProgName, _ApplicationItem, RequestItem, UserItem, 
String.Empty, String.Empty, _ErrorItem)
        Catch ex As Exception
            ‘ Communication error
            _Results.CallError.StringValue(1) = “998”
            _Results.CallError.StringValue(2) = “Call to “ & ProgName & “ Error: “ & 
ex.Message
            Return _Results
```

Figure.2

```
        End Try

        ‘ Cleanup the connection
        _Connection.Disconnect()
        _Connection = Nothing

        ‘ Updates the Call Results
        _Results.CallError.Text = _ErrorItem
        _Results.RequestItem.Text = RequestItem
        _Results.Data.Text = UserItem
        Return _Results
    End Function

    ‘’’ <summary>
    ‘’’ Name of the server we are connecting to.
    ‘’’ </summary>
    Public Property ConnectionName As String

#Region “IDisposable Support”
    Private disposedValue As Boolean ‘ To detect redundant calls

    Protected Sub CheckIfDisposed()
        If disposedValue Then
            Throw New ObjectDisposedException(“SpectrumDBContext”)
        End If
    End Sub

    ‘ IDisposable
    Protected Overridable Sub Dispose(disposing As Boolean)
        If Not disposedValue Then
            If disposing Then
                ‘  dispose managed state (managed objects).
            End If

            ‘  free unmanaged resources (unmanaged objects) and override Finalize() below.
            ‘ set large fields to null.
        End If
        disposedValue = True
    End Sub

    ‘ This code added by Visual Basic to correctly implement the disposable pattern.
    Public Sub Dispose() Implements IDisposable.Dispose
        ‘ Do not change this code.  Put cleanup code in Dispose(disposing As Boolean) above.
        Dispose(True)
    End Sub
#End Region
End Class
```

Figure. 2 - Continued

The good news is that it is really not very hard to wrap synchronous methods and functions into Task-based processes. If you have not gotten involved with Async/Wait coding yet, now is the time to start looking at it. To see an example of creating an Async/Wait-aware function from a synchronous call, take a look at CallSubroutineAsync.

The main purpose of the code in *Figure 2* is to call a subroutine with the following arguments in *Figure 3*.

```
SUBROUTINE PROG.NAME(APPLICATION.ITEM, REQUEST.ITEM, DATA.ITEM, NONE1, NONE2, ERROR.ITEM)
```

Figure. 3

I will get into this subroutine in a moment. For now, assume the following:

- Information about the application calling this subroutine will be provided in `APPLICATION.ITEM`.
- Request for the database and subroutine to process will be passed in using `REQUEST.ITEM`.
- Information will be returned from the subroutine through `DATA.ITEM`
- Any glaring errors should be returned in `ERROR.ITEM`
- The `NONE1` and `NONE2` arguments are there for future reference if needed.

I would highly recommend that you always use a single subroutine as the only point of contact whenever you are working with your MultiValue data from outside of the Database. This allows you to keep your business logic where it belongs. It also gives you a choke point for turning off all functions during maintenance. And it offers a single point for audit and security.

ApplicationUser

Before you can create the ApplicationUserManager and ApplicationSignInManager, you will need to create the ApplicationUser class. This class is used to hold information about the user for the rest of the OWIN Identity system to use. It will also contain the main methods to use in order to interact with the database.

This class requires you to implement the IUser(Of String) interface *[Figure 4]*. The interfaces will require you to implement an Id and UserName property. There are several other properties and methods I would recommend creating in this class as well *[Figure 5]*:

- wSaveAsync – Which will also call the subroutine to handle saving any changes made to the ApplicationUser class.

You will notice in the *Figure 5* that the LoadAsync creates a variable REQUEST.ITEM (the actual variable name is _RequestItem) and in REQUEST.ITEM<1> it has a value of "READ.USER" and in REQUEST.ITEM<2> it has the Id of the user we need information for.

One of the nice things about OWIN is that you can assign an arbitrary Id (primary key, record Id) to the user instead of having to translate the UserName every time like you used to have to do in ASP.NET 2.0 membership framework. For obvious reason, this makes pulling information from the database much easier.

SPECTRUM.OWIN.USER Subroutine

Let's take a quick look at the subroutine that we are calling. The first thing we need to do is to decide which request it needs to process *[Figure 6]*. You will notice that the code actually has three request types, or events, that we want it respond to:

The READ.USER event is to read the OWIN user information from the database, and return it back to the ApplicationUser Class. I have not gone into the full structure of the DATA.ITEM dynamic array yet.

```
Imports System.ComponentModel.DataAnnotations
Imports System.Security.Claims
Imports System.Threading.Tasks
Imports Microsoft.AspNet.Identity
Imports Microsoft.AspNet.Identity.Owin
Imports mvOpenAPI.api

Public Class SpectrumUser
  Implements IUser(Of String)

  Dim _Id As String = String.Empty
  Dim _UserItem As New mvDynamicArray

  Public ReadOnly Property Id As String Implements IUser(Of String).Id
      Get
        Return _Id
      End Get
  End Property

  Public Property UserName As String Implements IUser(Of String).UserName
      Get
          Return _UserItem.StringValue(1)
      End Get
      Set(value As String)
          _UserItem.StringValue(1) = value
      End Set
  End Property

...

End Class
  Public ReadOnly Property DBContext As SpectrumDBContext
   Get
    If _DBContext Is Nothing Then
        Try
         _DBContext = HttpContext.Current.GetOwinContext.Get(Of SpectrumDBContext)
        Catch ex As Exception
        End Try
        If _DBContext Is Nothing Then
            _DBContext = SpectrumDBContext.Create
        End If
    End If
    Return _DBContext
   End Get
  End Property
Private _DBContext As SpectrumDBContext
```

Figure. 4

```
Public Class SpectrumUser
    Implements IUser(Of String)

   ....

    Public Async Function LoadAsync() As Task
        Dim _RequestItem As String = “READ.USER”
        _RequestItem = _RequestItem & mvFunctions.AM & _Id

        Dim _DataItem As String = String.Empty
        Dim _Result As SpectrumDBContext.CallSubroutineResults = Await DBContext.
CallSubroutineAsync(“SPECTRUM.OWIN.USER”, _RequestItem, _DataItem)
        If _Result.CallError.BooleanValue(1, 0, 0) Then
            ‘ Error, can’t find the information
        Else
            ‘ Extract the User Id and User Item
            Update(_Result.Data.Item(0).Text, _Result.Data.Item(1).Text)
        End If
    End Function

    Public Async Function SaveAsync() As Task
        ‘ Saves the information to the database
        Dim _RequestItem As String = “WRITE.USER”
        _RequestItem = _RequestItem & mvFunctions.AM & _Id

        ‘ Send the data to the server
        Dim _DataItem As String = String.Empty
        Dim _Result As SpectrumDBContext.CallSubroutineResults = Await DBContext.
CallSubroutineAsync(“SPECTRUM.OWIN.USER”, _RequestItem, _DataItem)
        If _Result.CallError.BooleanValue(1, 0, 0) Then
            ‘ Error, can’t find the information
        Else
            ‘ Extract the User Id and User Item
            Update(_Result.Data.Item(0).Text, _Result.Data.Item(1).Text)
        End If
    End Function

  ...

End Class
```

Figure. 5

```
SUBROUTINE SPECTRUM.OWIN.USER(APPLICATION.ITEM,REQUEST.ITEM,DATA.ITEM,NONE1,NONE2,ERROR.ITEM)

...

*
ERROR.ITEM = 0
DATA.ITEM  = “”
BEGIN CASE
  CASE REQUEST.ITEM<1> = “FIND.USER”
    GOSUB 1000
  CASE REQUEST.ITEM<1> = “READ.USER”
    GOSUB 2000
  CASE REQUEST.ITEM<1> = “WRITE.USER”
    GOSUB 3000
  CASE 1
    ERROR.ITEM = 2 :AM: REQUEST.ITEM<1> :” not implemented”
END CASE
*
RETURN

...

2000 * Read User Information
*
ID = REQUEST.ITEM<2>
```

Figure. 6

```
READ USER.ITEM FROM OWIN.USER.FILE, ID ELSE
   ERROR.ITEM = 1 :@AM: id :” is not a valid user.”
   RETURN
END
*
DATA.ITEM<1> = USER.ITEM<1> ;* UserName
DATA.ITEM<2> = USER.ITEM<2> ;* Password
DATA.ITEM<3> = USER.ITEM<3> ;* Email

...

*
RETURN

...

3000 * Write User Information
*
ID = REQUEST.ITEM<3>
READU USER.ITEM FROM OWIN.USER.FILE, ID LOCKED
   ERROR.ITEM = 3 :@AM: “Record is currently locked by another process.”
   RETURN
END ELSE
   USER.ITEM = “”
END
*
USER.ITEM<1> = DATA.ITEM<1> ;* UserName
USER.ITEM<2> = DATA.ITEM<2> ;* Password
USER.ITEM<3> = DATA.ITEM<3> ;* Email

...

*
RETURN
```

Figure. 6 - Continued

The WRITE.USER event is to take the information from the ApplicationClass and place it back into the database.

The last event is the FIND.USER. This event will be used to find the record to use when someone first logs in.

The DATA.ITEM will have around 20 attributes that will be used by the ApplicationUser class. As you look at the dynamic array definition I just provided, you will see there are references to two-factor authentication, claims, roles, and external providers. I will talk more about these pieces of data in later articles.

DATA.ITEM<1> = User Name
DATA.ITEM<2> = Password
DATA.ITEM<3> = Email
DATA.ITEM<5> = Cell
DATA.ITEM<7> = Security Stamp
DATA.ITEM<8> = TwoFactor Enabled
DATA.ITEM<9> = Lockout
DATA.ITEM<10> = Lockout Offset
DATA.ITEM<11> = Access Failed Count
DATA.ITEM<12> = First Name
DATA.ITEM<13> = Last Name
DATA.ITEM<14> = Company Name
DATA.ITEM<15,1> = External Login Provider
DATA.ITEM<16,1> = External Login Token
DATA.ITEM<17,ccnt> = Claim Field Name
DATA.ITEM<18,ccnt> = Claim Field Value
DATA.ITEM<19,rcnt> = Roles

Conclusion

This may seem like a lot of work, but it is really just building a set of code that can be used over and over again. This first article covered much of the hard part, but there are a few more pieces that you will need to make everything connect together.

I will continue showing those other components and how it all connects together in upcoming articles in the series. **IS**

NATHAN RECTOR
President
International Spectrum
nathan@intl-spectrum.com

FROM THE PRESS ROOM

Groware Eases Inventory Management Challenges for Nursery Owners and Growers by Launching Mobile Apps Using Bluefinity's Rapid App Platform: Evoke

Groware, one of Canada's leading providers of software for the grower and nursery markets, has partnered with global IT company Bluefinity International to create mobile app versions of its inventory management software, which is already improving inventory management for its customers.

Using Bluefinity International's rapid app development platform, Evoke, Groware has developed beta mobile apps to help its customers ease inventory management challenges by allowing them to stock take accurately, plan their crops and manage their business more effectively.

Evoke provided Groware with a low cost and completely flexible platform that integrated with its existing databases and enabled it to use its existing code to quickly and easily create mobile apps. In just a few months of working together, Groware launched its beta mobile apps to rave customer reviews.

For Groware, a key benefit of using Evoke has been the level of support provided by Bluefinity throughout the development process and that Evoke enables apps to be created and optimised for any device with the right look and feel for different customers.

Prior to using Evoke, Groware had introduced handheld devices to help customers update tasks and stock take more easily in the field. However, with most customers located in rural areas with limited internet coverage, updates were rarely carried out in real-time and the handheld devices needed to be synced later with other software. Inevitably data inaccuracies would creep in.

Michael Heeney, Owner of Groware says, "Customers need real-time technology to enable truly accurate inventory management and it was our goal to offer mobile apps to solve these issues. Whether customers are using a desktop, handheld device, mobile phone or tablet, the apps can be automatically customised for each device so they look completely professional. Evoke is completely flexible and enables customers to make edits in real time.

"If someone is out in the field using the app to stock take, they can post live data and make adjustments in real time too - it is enabling far more accurate inventory management and our customers love it. It is improving the management of their operations and streamlining their processes."

"Evoke isn't limited to the handheld devices either and it doesn't need to be constantly internet based as it can provide for off line work, and fully automated synchronisation when connection is re-established. In future we'll be re-working all of our software so that it is generated using Evoke, which will benefit our customers hugely," adds Mr Heeney.

Another benefit of Evoke for Groware's customers is that some of them have seasonal businesses, such as Christmas tree companies. As the subscription to Evoke is a user based monthly subscription - it is highly cost effective for such companies to use the platform to manage their business seasonally and only pay for the solution when they need it.

Evoke enables the rapid development and deployment of mobile, desktop and web, hybrid and native apps for any device and operating system. It uses a single app design and code base, and enables apps to be created for multiple device types, databases and operating systems including iPads, iPhones, Android and Windows phones and tablets, plus Windows, Linux and Apple desktops.

Evoke's "no code" option, uses point and click technology to add Evoke business functions that read, update calculate, restrict and adapt areas such as database entries, the user experience and the navigation through the app. This has opened up building professional, sophisticated business apps to everyone.

Malcolm Carroll, Director, Bluefinity International said, "We are delighted to have enabled Groware to achieve its goal of launching mobile apps successfully for its customers. Using Evoke, it has the right platform in place to achieve further growth and success and we look forward to a long partnership with the company." ■

MacWise Version 16.2.5 Released

MacWise is now compatible with Mac OS High Sierra. And the latest version of Kermit (9.0.304) is now included with MacWise.

FUNCTION KEYS WITH KERMIT

Some host servers absorb the first escape character sent from MacWise when using Kermit. VT100 and VT220 emulations use escape characters in the function key commands. To get around this problem, MacWise now includes special function key sets for VT100 and VT220.

If your function keys do not work, select Function Key

From the Press Room

Sets from the Emulate Menu in MacWise to choose your fkey set. This can also be a problem for arrow keys that use escape. You can edit arrow keys and add an extra escape which is {ES

Select Arrow Keys... from the Emulate Menu to edit each arrow key.

This does not affect other emulations such as Viewpoint and Wyse because they do not use escape characters for fkeys and arrow keys. ■

Entrinsik Informer Named a Top Performer for Business Intelligence Software in 2018 Customer Success Report

Entrinsik Inc. recently announced it has been named a "Top Performer" for business intelligence tools in the 2018 Summer Customer Success Report published by FeaturedCustomers.com. FeaturedCustomers.com is the number one platform for finding and reviewing IT software through verified customer success content. With nearly 200,000 customer endorsements from some of the most recognized brands, FeaturedCustomers.com helps its prospective users make informed IT purchasing decisions.

Entrinsik Informer simplifies the often complex process of accessing, cleansing, blending and analyzing data from multiple sources by leveraging a modern, plug-in architecture and easy-to-navigate user interface. Informer enables organizations around the world to extract critical insights buried within their data, whether that data is in ERP systems, spreadsheets, or the cloud. Organizations can aggregate this disparate data to create a cohesive, curated, governed data hub for self-service reporting, visualization and data analysis.

The Customer Success Report is based on the curation of authenticated case studies, testimonials, and reviews from across the web, telling a complete story of vendors in the BI space through real customer experiences. Scores are determined by a number of factors including the number of reviews, customer success content, social media presence, and vendor momentum based on web traffic and search trends.

The Top Performer designation is awarded to vendors with significant market presence and resources and enough customer success content to validate their vision. Highly rated by customers, Entrinsik Informer consistently publishes customer-driven content and curates dozens of positive reviews from multiple third-party sites.

"We are once again honored to be recognized for our dedication to the success of our customers and are excited to be included in FeaturedCustomer.com's report. "The team at Entrinsik work very hard to deliver a feature-rich solution at a competitive price to organizations looking for a proven BI and data analytics platform to drive business success," says Doug Leupen, CEO at Entrinsik. ■

Mercury Database Console Begins Development of Version 5.0 FLASH!

Paradigm Systems is pleased to announce that they have begun development of Mercury 5.0 which will be known as FLASH.

Mercury FLASH is going to be a complete redesign of the Mercury Database Console product with an exciting new web-based interface. This new release will make it easier for Paradigm Systems to push out product updates, and provide our clients with the features and tools they need to successfully manage their growing U2 database.

Mercury Database Console and FLASH includes an extensive analysis feature that learns your database as it grows. The more knowledge it is feed, the more effective it becomes at both supporting and empowering the way you like to work. This extended analysis provides optimal sized based on your current size, as well as, projected future growth. ■

Universal Driver NUL v5.1 Released

Universal Driver NUL v5.1 for OpenInsight 10 - This is a 64-bit server side installation for use with OpenInsight 10. The Universal Driver NUL v5.1 allows files from OpenInsight 10 and OpenInsight 9.4 to share the same Universal Driver. This version has been updated with the latest ACE libraries. Management of the Universal Driver is done via the OI Console.

Universal Driver NUL v5.1 for OpenInsight 9.4 - This is a 64-bit server side installation for use with OpenInsight 9.4 and a 32-bit client side installation for use with OpenInsight 9.4. The Universal Driver NUL v5.1 allows files from OpenInsight 10 and OpenInsight 9.4 to share the same Universal Driver. This version has been updated with the latest ACE libraries. Management of the Universal Driver is done via the UD Manager. ■

FROM THE PRESS ROOM

Zumasys Launches Cloud Managed Service Offering with Acquisition of TCS Network Consulting

Zumasys, Inc announced it has acquired the assets of Orange County based managed services provider, TCS Network Consulting.

As part of this transaction, Keith Hermance has joined Zumasys as a Senior Solutions Consultant. All members of the TCS team have transitioned to Zumasys' headquarters located in Irvine, CA.

"TCS has a proven track record of providing proactive, managed IT services for a fixed monthly fee to clients across Southern California," said Paul Giobbi, Founder and CEO of Zumasys. "We believe that customers running on The Zumasys Cloud and those using our award-winning Desktop-as-a-Service (DaaS) solution will be keenly interested in this all-you-can-eat service model."

Zumasys' new managed services division will help customers seeking to eliminate the administrative overhead and variable cost associated with managing their Cloud/IT environments. Customers receive 24/7 monitoring of their IT environments and proactive responses to issues for a fixed monthly fee.

Cloud managed services are surging in demand across the globe due to the growing need of businesses to curb operational and infrastructural costs involved in business processes according to a 2017 report by MarketsandMarkets™. The cloud managed services market is expected to grow from $27B in 2017 to more than $53B by 2022.

"Managed services are ideal for customers who want the benefits of a full time IT department without all of the overhead," said Keith Hermance. "I am looking forward to creating a world-class cloud managed services offering and I am excited to join the Zumasys team." ■

International Spectrum 2017 Anthology

Now Available in Print!

January/February 2017 to November/December 2017

In a Single Bound and Printed Volume.

Order Your Copy Today!

Available on Blurb:
http://www.blurb.com/b/8523259-international-spectrum-2017-anthology

UX/UI Part V

BY CHARLES BAROUCH

Next issue we'll get on to a new topic, however, I do have some final thoughts to share on User Interface (UI) and User Experience (UX).

Help Doc!!!

UI/UX isn't just about coding. While it is nice to add screenshots to a help document, I like to go a step further and use a little bit of HTML to create flipbooks.

Like a wizard, which guides you through a process, a flipbook is a series of pages which act like a walk-through for your software. Emphasizing the visual by making the screenshots the centerpiece of the help, instead of just tacking them on at the end, speeds up comprehension. If we want our work to be appreciated, we need to make the people who use it successful.

Of course, because we create flipbooks in HTML, adding links to related help, or to online resources becomes trivial. The goal with help, as it is with software, is to make things work in a way which makes sense.

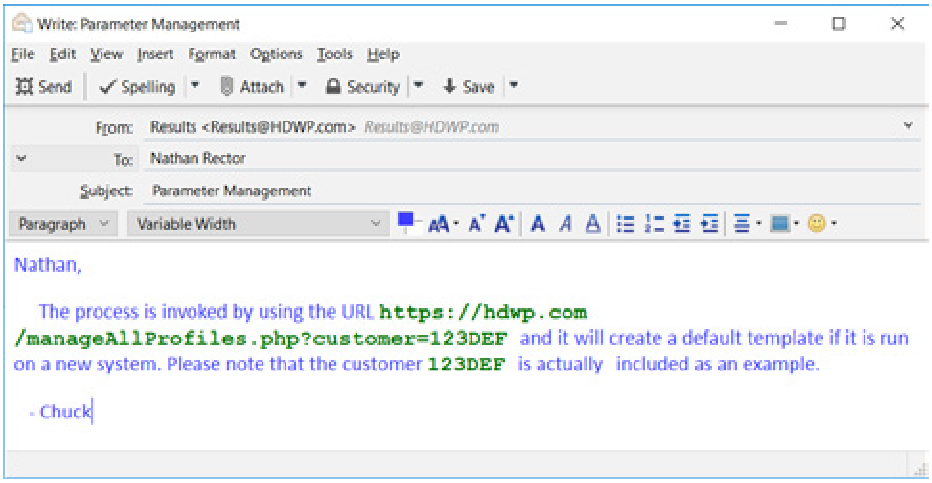
Effective transfer of knowledge isn't just about words.

Suspect is Mail

While the program you create and receive email with has a UI/UX aspect, the mail itself can also benefit from a little kansei engineering. I routinely have to write e-mails which refer to business concepts and contain example code or commands. I write the computer parts in twelve-point Courier New using green, bold letters. This sets it apart from the rest of the document in a way that the reader can immediately discern.

I use this technique for all code and commands, not just MultiValue. Remember, most people outside of our industry don't see HTML, Java, mvBASIC, or Swift as different from each other. Simply marking the code as code is a way for the people who don't understand to know what to ignore. The few recipients on the e-mail who care about the code have what they need. The others don't feel excluded from the chain, as they might if we were to start sending different e-mails to different groups.

I do the same when I have to paste in outputs (reports, totals, etc.). This has the added benefit of making columns line up because Courier New is a mono-spaced font. Making documents more easily readable improves the user experience



From: Results <Results@HDWP.com> Results@HDWP.com

To: Nathan Rector

Subject: Parameter Management

Nathan,

The process is invoked by using the URL **`https://hdwp.com/manageAllProfiles.php?customer=123DEF`** and it will create a default template if it is run on a new system. Please note that the customer **`123DEF`** is actually included as an example.

- Chuck

Figure.1

If you don't see why formatting is so important, take a moment to review the approach we take to this magazine. Tracey does an amazing job of directing your eyes where they need to go by providing a clean, clear, and consistent layout. She keeps your interest by changing up a reasonable amount of things between issues. Not everyone happens to have a background in layout, but we can all make an effort to craft our written communications. Effective transfer of knowledge isn't just about words. White space, length of sentences, even your choice of punctuation, they all add to the results.

Clear and Presentations

What goes for text goes for verbal communication. Separating the parts of your presentation to meet each audience segment maximizes the value of the time everyone spends listening to you.

If you know that the V.P. attending your meeting is likely to get pulled out for a call, you should load what they need into the early part of your talk and let the rest wait. If you know that the engineers are perpetually late, move the bits critical to them into the middle. I often arrange my timeline based on attention spans. I know who is likely to tune out and who is likely to take a while to warm up to the topic.

Hard Stop

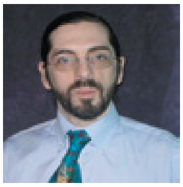
Taking UI/UX beyond software includes factoring in hardware. Some of us older folks remember when you could buy PC keyboards with special keycaps that told you which word processing shortcuts were enabled by which keys. While ordering a Word Perfect or Microsoft Word keyboard has fallen out of fashion, adding a laminated page to the keyboards your IT team provides might make everyone's job easier. Data Entry could get their cheat-sheet and a different page could be attached for Finance and yet another for the Receiving Dock.

I've even seen shops where additional physical controls have been added to PCs to do tasks which could have been done with key combinations. They do it because it makes it easier on the users.

My favorite hardware UI/UX story in the MultiValue world is Dick Pick's reaction to the then-new idea of using a mouse. He felt that your hands both belonged on the keyboard so he proposed a version of the mouse which was more like a commercial sewing machine's foot pedal. He said, "Why click when you can stomp."

Buy the Way

UI/UX isn't something you "sprinkle on" after the work is done, it should be part of the core process of designing and implementing every phase of your project. For those of you who want some help making a better and more consistent user experience, watch the next several issues for articles spotlighting 4GLs. **IS**

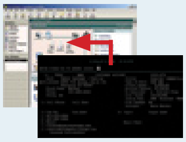


CHARLES BAROUCH is the CTO of HDWP, Inc. and the Publisher at HDWPbooks. You can read his writing in International Spectrum, Theme-Thology, Novo Pulp, Pax Solaria, PerehelionSF, and the Interrogative series, which begins with Tiago and the Masterless.

MARKETPLACE

ACCOUNTING

Natec Systems
www.natecsystems.com | nrector@natecsystems.com



COMPLIANCE

SJ+ Systems Associates
www.sjplus.com | sjoslyn@sjplus.com

CONSULTING

Drexel Management Service
www.drexelmgt.com | dconboy@drexelmgt.com

Execu-Sys, LTD
www.eslny.com | mh@eslny.com

HDWP
www.HDWP.com | results@HDWP.com

Modern MultiValue, LLC
www.ModernMultiValue.com | info@ModernMultiValue.com

PICK Programmers Shop
www.pickprogram.com | brian@pickprogram.com

Precision Solutions
www.precisonline.com | Kevin@PrecisOnline.com

DATABASE

Ladybridge Systems Ltd
www.ladybridge.com | sales@Ladybridge.com

REPORTING

Brian Leach Consulting, LTD
www.brianleach.co.uk | brian@brianleach.co.uk

TERMINAL EMULATOR

Zumasys
http://www.zumasys.com/products/accuterm/

LETTERS TO THE EDITOR

Have an opinion on an article: Agree, disagree, or enhancement to an article from a previous issue? International Spectrum and our authors are interested in hearing from you!

E-mail: editor@intl-spectrum.com

WANT TO SEE A SPECIFIC TOPIC?

International Spectrum is looking for writers, feedback, and topic ideas. We all have specific topics and issues that we need answers to find solutions for. Send us an E-mail with topics you would like to have covered in the magazine or on the website.

E-mail: nathan@intl-spectrum.com

WANT TO WRITE?

Expand your professional credentials, and provide us with an article.

Give us a rough and ugly outline, and we will help you refine it, proof it, and make it press ready. Or you can give us something polished, proofed, and press ready to publish.

Share your thoughts and expertise with over 10,000 fellow MultiValue developers and users.

E-mail: editor@intl-spectrum.com

NEED A MENTOR?

Mentors give developers the ability to ask industry experts for direction, code examples, and/or just ask them to see if something makes sense. Sometimes, all you need is a resource or example to start or complete a project.

Check with us to see who is available for mentoring, and how you can take advantage of it to save your business or company money.

E-mail: nathan@intl-spectrum.com

WANT TO BE A MENTOR?

We have many retired or semi-retired professionals out there that would love to share their knowledge of MultiValue development. If you are one of them, please contact us to see what mentoring is all about.

E-mail: nathan@intl-spectrum.com

Compressing Traffic for UniObjects

BY BRIAN LEACH

One of my clients has been looking at a particular pinch point in their application: fetching potentially large volumes of data for presentation in a data-bound grid.

As well as the usual issues of building data, and it lacked parallelism. It runs a selection, grabs the data, transfers it and then builds a DataSet — all serial and with plenty of options for asynchronous building to improve the overall response.

Additionally, the physical architecture at one of their customers' sites means that what can be large volumes of traffic being transferred over the UniObjects protocol from their server is slow. The data itself should be highly amendable to compression so the question came — could I find some way to compress this?

If you look at the members of the old Session object, there is a compression flag. That might seem like a good starting point *but it is not settable.* It is also conspicuously missing from the U2 Data Provider for .NET, suggesting that it was a hoped-for extension that never got built. So, no go there.

So why not leverage the python support? ...this seems like a natural choice.

But in this case all that is really needed is to compress the data itself before it is sent over the wire, and using a form that can be quickly and readily reinstated on the other end. The application makes a call to a UniBasic subroutine to generate and return the data, so if I can compress using a standard format before it leaves the subroutine and decompress it the other end once it has been received, we should be in business.

It should be quick, of course, since there is no point adding time for the compression that would undo any performance improvements from reducing the traffic. I have a compression routine in UniBasic that I use for my installer packages, but in that situation I am not overly concerned about speed.

To GCI or Not To GCI

My first thought was to use a GCI function to compress the data. For those who have not familiar with GCI, it is a way of extending UniVerse with native C code. Writing GCI on Windows is simpler since it can call DLLs, but this needs to run on Windows and Linux and on Linux it is potentially painful. And, other than C#, I've never particularly enjoyed writing C code. Delphi is always an option on Windows, but I've had problems calling the equivalent FreePascal from GCI on Linux.

So GCI remains an option, but only if I can't find something simpler to administer.

Calling Python

So why not leverage the python support? I've been looking for excuses to find things for python to do, and this seems like a natural choice. Python supports zlib compression, and with a little bit of jiggery-pokery that can be turned back using the .NET System. Compression.DeflateStream. ZLib uses the same encoding as deflate with the addition of a header, checksum and optional dictionary. The only difficulty is managing the encoding at each end to convert these to and from string representation.

Simple enough? Well, nearly — it took me two goes to get it working. It's the old issue of Unicode rearing its ugly and malformed head.

UniVerse 11 introduces a PyCallFunction() call into UniBasic. This calls a function in a Python module from a nominated location and can pass zero

or more arguments. The function return value is captured and assigned in the UniBasic program <*Figure 1*>.

Which works well for regular data, and anything that can be represented as standard Unicode.

In Python 2 you could compress a string. A string was nothing more than a managed set of bytes, as it is on UniVerse, old style Delphi, etc. etc. A string could represent anything and we didn't care which of the 256 possible values filled each byte. Then along came Unicode, and everyone adopted it for their strings and now you can't use strings for any kind of binary data. Delphi sensibly has a RawByteString to fill the gap but for .NET and python there is no equivalent. There is no Unicode formatting for old style 8-bit data that works since Unicode confuses the absence of representation with non-existence. It's a fundamental flaw in Unicode.

In Python 3 you can only compress something that is byte-able, like a series of bytes. Converting a string into a byte array is relatively straightforward so long as you can find an equivalent encoding format. UniVerse always passes strings in as Unicode, so a field marks (char 254) get translated, for example, to its UTF-8 equivalent pairing. Pass that in and out and you get to see the c3 be (195 190 decimal) pairing in Figure 2.

This snippet first converts the original packet to bytes, then those bytes into a hex encoding before passing back the hex encoded string. When passed a 4 field dynamic array from UniVerse, the resulting encoded string returned to UniVerse, and its MX0C conversion, are given in Figure 3.

You could choose to keep the original markers which will be neatly handled on the client side (that also expects UTF-8) or convert them into low order characters such as char(1) to char(3) for field, value and subvalue markers if you don't want to double up. So in normal operations we can pass a string, decode it into bytes and compress it as in Figure 4.

Returning the compressed result is more of a problem: UniVerse requires that this is returned as a string. UniVerse strings are not the same as Python strings and whilst UniVerse encodes into Unicode on the call, to get the data back UniVerse automatically takes the platform encoding, which defaults to cp1252 (Latin-1). This is an extended ASCII encoding and one that does not include the whole range of 8 bit characters. Put simply, even if you encode stuff using an 8 bit ISO representation inside your Python function, you cannot directly return a string containing binary data to UniVerse.

```
Value = PyCallFunction('myModule','myfunction','Arg1','Arg2'..)
```

Figure. 1

```
def pkt(origPacket):
  newBytes = origPacket.encode('utf-8')
  newPacket = ''.join(format(x, '02x') for x in newBytes)
  return newPacket
```

Figure. 2

```
496e207468652064656570206f662074686520736561c3be496e2074686520646565702077617465722062
6c7565c3be4c69766564206120666973682077686f20636f756c642077697368c3be4
6e642065616368207769736820776f756c6420636f6d652074727565
```

In the deep of the seaÃ¾In the deep water blueÃ¾Lived a fish who could wishÃ¾And each wish would come true

Figure. 3

```
import zlib
def zip(origPacket):
  bytes = origPacket.encode('utf-8')
```

Figure.4

From the Python perspective once we've turned this into a set of bytes we should keep it there, and whilst this

```
def zip(origPacket):
      bytes = origPacket.encode(‘utf-8’)
      newBytes = zlib.compress(bytes)
      newPacket = ‘’.join(format(x, ‘02x’) for x in newBytes)
      return newPacket
```

Figure. 5

```
ModuleName = ‘u2zip’

Zipped = PyCallFunction(ModuleName,’zip’, InData)
If @PYEXCEPTIONTYPE NE ‘’ Then
   ErrText = @PYEXCEPTIONMSG
   ErrText<-1> = @PYEXCEPTIONTRACEBACK
End Else
  RealZipped = IConv(Zipped,’MX0C’)
End
```

Figure. 6

makes good sense all the time we’re working in Python (the same would be true in .NET), in order to return it to UniVerse without going through an intermediate file, I need to turn it back into a string that will fit with the Latin-1 encoding. I can’t simply take those bytes and encode them using, for example, iso8539-1 which would preserve the content byte-for-char since the subsequent UniVerse re-encoding on exit will trash that.

Since this is only in-memory and UniVerse is sharing the same process space, it’s not too much of an overhead to simply hex encode it as above; not least because whilst it is wasteful there is a single conversion code on the UniVerse side to decode it back and if the engineers have been sensible it should only need one memory reallocation (since you know the new string will be exactly half the size). It might also be possible to base64 encode it, but Python sulks at that <*Figure 5*>.

Now I can call this and fetch my compressed data on the UniVerse side <*Figure 6*>.

That all looks reasonable and it’s saving some space now, so time to build this into my subroutine call from the client.

Note that the module needs to be in a path where UniVerse will discover it: this is held in a *.pth file. See the U2 Python guide for details. Note also that UniVerse caches Python calls for a session for speed, so any modifications to the Python routine require that you log out and in again.

```
Temp = encode(“BASE64A”, 1, RealZipped, 1, OutData, 1)
```

Figure. 7

Calling from .NET

I need to get that same set of bytes generated from Python back in .NET. This runs into exactly the same problem - if you try to get that as a string using the UniSubroutine.GetArg() method, it will try to encode it as Unicode and the result will be a horrible mess that will not decode correctly. Again I could simply base64 encode it <*Figure 7*>.

And then convert it on the .NET side <*Figure 8*>.

But now I've stuck a load more bytes into the traffic for the encoding scheme when I'm trying to reduce them. There is still a net gain — base64 encoding is quite lean — but that is obviously not what I want.

Fortunately, all is not lost. You can fetch a subroutine argument as a string, or as a dynamic array object (a UniDynArray). The latter exposes a useful method: ToByteArray(). So I can return the underlying bytes that got transferred over the wire from my UniVerse subroutine <Figure 9>.

Is that all there is to it? Not quite – Deflate and ZLib compression are not quite the same. ZLib adds a header, checksum and optional dictionary to the data. These can be stripped off easily, or there are other zlib libraries that can be leveraged: but this is enough or what I need. **IS**

BRIAN LEACH is a MultiValue developer, consultant, trainer and author, and a board member of the Rocket Software Users Group. You probably knew that. Find him at http://www.brianleach.co.uk.

```
byte[] bytes = Convert.FromBase64(OutData);
```

Figure. 8

```
UniDynArray outData = subr.GetArgDynArray(2);
byte[] bytes = outData.ToByteArray();

Put these together and I can decode the compressed data to get the original results:

string decode (byte[] bytes) {
   string result = string.Empty;

   MemoryStream s = new MemoryStream(bytes);
   DeflateStream z = new DeflateStream(s, CompressionMode.Decompress);
   const int size = 4096;
   byte[] buffer = new byte[size];
   using (MemoryStream memory = new MemoryStream()) {
       int count = 0;
       do {
           count = z.Read(buffer, 0, size);
           if (count > 0) {
               memory.Write(buffer, 0, count);
           }
       }
       while (count > 0);

       // now convert these back into a string
       memory.Position = 0;
       StreamReader r = new StreamReader(memory);
       result = r.ReadToEnd();
   }
   return result;
}
```

Figure. 9