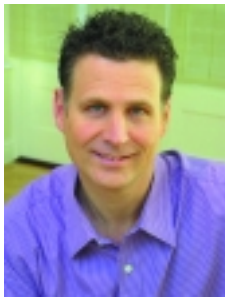
mv.NET:
Bridging the Gap From MultiValue to Microsoft .NET

mv.NET is the brainchild of U.K.-based BlueFinity International, a company founded in 2002 by a group of software designers and developers whose mission was to create "state-of-the-art software tools for the rapid creation of professional, business-oriented, MultiValue database-centric applications utilizing Microsoft's .NET environment." mv.NET (its first production release became available in December 2004) melds the design team's vast experience in creating both end-user applications and the lower-level developer tools used to create those applications, and their knowledge of the .NET framework. BlueFinity's engineering team has been working with the.NET framework since its early pre-beta releases in 1999, the company said. Coupled with the design team's knowledge of the .NET environment is a history with MultiValue. The company's engineers have experience working with all flavors of MultiValue databases, ranging from the early days with McDonnell Douglas to the modern days with jBASE, IBM, Raining Data and others.

In this article, BlueFinity's senior developer, David Cooper, reveals what differentiates mv.NET from other .NET offerings in the MultiValue market; the underpinnings of the product's design; and how it bridges the gap for MultiValue users who need to make the leap to Microsoft .NET.



DAVID COOPER
senior developer at BlueFinity International

Q At its inception, what was the purpose for developing mv.NET?

ANSWER: We saw that there was a gap in the market to provide a complete MultiValue development environment for Microsoft .NET able to connect into all major flavors of MultiValue platforms, not only the latest versions of the vendors' MultiValue databases but also previous versions too. By that I mean that we needed to not only provide the development environment and handle connectivity but also to handle management and administration as well. Plus, and by no means a *small challenge*, was to make "everything for the developer" work from within Visual Studio .NET.

We had seen several early attempts of the MultiValue database vendors to provide .NET interfaces (for example, IBM U2 with UO.NET, which is only available on the very latest releases) while the rest were still struggling with COM based interfaces. Generally speaking, the .NET interfaces provided by the vendors to date have really only enabled the minimum object level possible. This has resulted in the developer having to take responsibility for setup and management of the whole environment and to implement a lot of additional code to ensure an efficient and high-performance application. The true cost is not only manifested by long lead-times for delivery of applications to market but also, in our experience, a higher cost of ownership in the long term.

So, the combination of no fully cross-platform lead from the MultiValue database vendors, Microsoft's .NET technology becoming pervasive and winning the battle in many key areas against Java (plus a mega-dollar marketing budget) and, not least, our vision and experience, resulted in the creation of our flagship product — mv.NET. mv.NET is therefore the .NET development and deployment product of choice for MultiValue databases.

Q How does mv.NET help meet the needs of today's MultiValue users?

ANSWER: The key characteristic which we had to ensure for mv.NET was the striking of the optimum balance between making the .NET environment a familiar and comfortable space for the Multi1Valued developer — allowing them to continue utilizing all of the cool things that you can do with MultiValue technology,

Continues on page 30

mv.NET: Bridging the Gap From MultiValue to Microsoft .NET

Continued from page 29

while at the same time making sure that the really cool things which you can do with .NET were still fully available. No simple task.

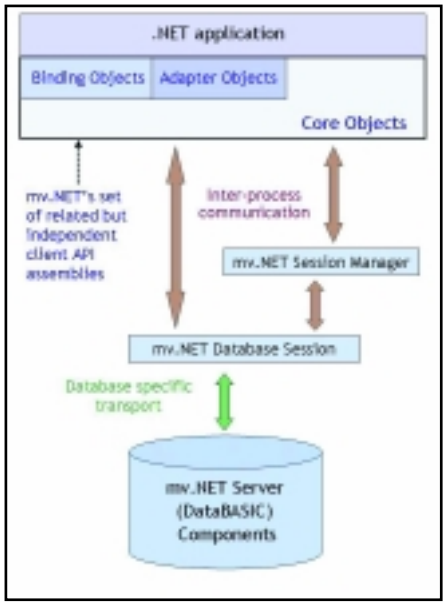
With all the undoubted skills that Microsoft has brought together to bring its .NET technology to market it also appears to have the "right product at the right point in time." You only have to see the major ramp up in .NET application deployments this year and the speed that applications are being rolled out to see that it was vital for MultiValue developers to also be able to ride on the crest of the wave. mv.NET, therefore, needed to deliver all the application environments of Web, Web Services and Rich Client and provide .NET style data access technology through a full ADO.NET managed data provider. Quite simple when written down quickly!

So for any MultiValue VAR, dealer or end user that has the need for Microsoft .NET, then mv.NET is available today. In addition, mv.NET is the ideal technology for some sites that today don't even know that they want Microsoft .NET — let me try and explain by looking at a couple of examples.

A large corporation has Oracle and SQL Server together with several core business applications running on UniData. They also have a new .NET policy and need access to data in the UniData database from Oracle and SQL Server based applications. Using mv.NET and Web Services, the data can be presented to the Oracle and SQL Server applications on-demand. A straightforward project to implement and the UniData system is no longer seen as a remote data island.

A medium sized company has all its core business applications from a VAR running on D3. The company has grown rapidly and the applications are tightly matched to the company's business needs. The new CFO has a requirement for a new application, so stock answer — buy in an SQL application. Wrong! mv.NET with Binding Objects will enable RAD application development that makes full use of the corporate data in D3. Application look and feel is the equal of anything an SQL vendor can offer, plus there is tight integration with existing core data. All this is achieved without the inherent risks of replacing the application and the database.

It's about time that we had a look at the architecture behind mv.NET.



As can be seen from the diagram, Core Objects provide the foundation underpinning both Binding Objects and Adapter Objects — but more on them later.

Initially, let's look at mv.NET's Data Manager. This tool can be run standalone or as a VS.NET add-in and acts not only as a setup/management utility but also as an invaluable developer tool. The starting point in Data Manager is to set up the configuration database which holds all information on all available MultiValue databases (Server Profiles), details of all available accounts on those servers (Account Profiles) and Session Pooling configu-

mv.NET: Bridging the Gap From MultiValue to Microsoft .NET

Continued from page 31

Binding Objects components can be used outside of the Visual Studio environment, but really come into their own when used in conjunction with mv.NET's extensive VS.NET add-in technology. Dockable toolbars providing databinding summary information along with sophisticated auto positioning of automatically created bound controls which act to save the developer valuable time in their task of creating state-of-the-art applications.

Adapter Objects provide the developer with a range of components designed to allow efficient ADO.NET based access to MultiValue databases. In fact this functionality brings the MultiValue developer as close to the SQL Server developer in terms of technology and functionality as s/he is ever going to get. In order to provide a comprehensive ADO.NET solution we have provided the following two groups of components:

- MultiValue database specific implementations of the ADO.NET classes/interfaces
- Visual Studio.NET add-in components to aid developer productivity in the use of Adapter Objects

So that we can present a MultiValue oriented ADO.NET managed data provider, Adapter Objects provides the developer with mv.NET specific classes, most of which inherit from the corresponding .NET framework IDbxxx interface. As I think you will now begin to expect from us, we have worked hard to ease the use of Adapter Objects from within the Visual Studio IDE. To do that we have provided a range of VS.NET extensions which are used in various places within the VS.NET IDE.

Q What database environments is mv.NET compatible with?

ANSWER: At the beginning I mentioned that a fundamental design aim for mv.NET was to provide connectivity to ALL the main MultiValue databases. We have done this and are providing the developer with the greatest flexibility to deliver applications across the widest range of databases. We also know that many organizations like the comfort zone and don't want to have to install the "latest" database release, so mv.NET is not ageist and will run on many of the old releases.

We are able to ensure, by using telnet as one of the various connection methods, the widest database support possible. Testing has also found that with the majority of databases very good performance can be obtained and good security is implicit with the correct firewall implementation and use of VPN connections for remote sites. However, wherever the database supplier has a suitable connect method we will make use of it, i.e., UniObjects for U2 and UniVision's SAC. Today we support D3, jBASE, mvBASE, Reality, UniData, UniVerse and UniVision and have immediate plans for mvEnterprise and Power95 and possibly Revelation. If there are any that we have missed, please contact us and we are sorry to have missed you out! *Continues on page 44*

mv.NET: Bridging the Gap From MultiValue to Microsoft .NET

Continued from page 43

Q How is it different from other products that are in the same arena?

ANSWER: This question is in fact very straightforward to answer. Our developers come from two groups and have built up a long-term working relationship, plus they like the odd beer together as well. The MultiValue group has skills from previous COM based projects but had also spent time working as application developers. In implementing mv.NET, the first group therefore has taken into account the "requirements of the developer" as much as the pure "functional specification." The second group has been Microsoft .NET through and through from its inception. To them the challenge has been to enable mv.NET functionality from within Visual Studio .NET IDE. A challenge that has stretched their skills to the limit several times — the inside of Visual Studio being a "wondrous place." So why are we different? Well, mv.NET is unique in providing a complete Visual Studio .NET environment for the MultiValue developer. Having said that, owning VS.NET is by no means mandatory for the use of the product.

Core Objects and Binding Objects have been implemented with the practical application development experience I mentioned earlier as well as with our MultiValue and .NET skills. From the many comments passed to us so far, developers are finding ease of use and productivity there from the moment they finish the installation of mv.NET. In fact, when the many readers of this article have the opportunity to try out Binding Objects and Adapter Objects, I am sure that they will also find that mv.NET has moved .NET development well on to the next level, way beyond anything else on the market today.

> "You only have to see the major ramp up in .NET application deployments this year and the speed that applications are being rolled out to see that it was vital for MultiValue developers to also be able to ride on the crest of the wave."

The combination, therefore, of the way the MultiValue implementation has been undertaken and the fully integrated MultiValue aware Visual Studio .NET IDE ensures that mv.NET stands out as the product of choice in this market.

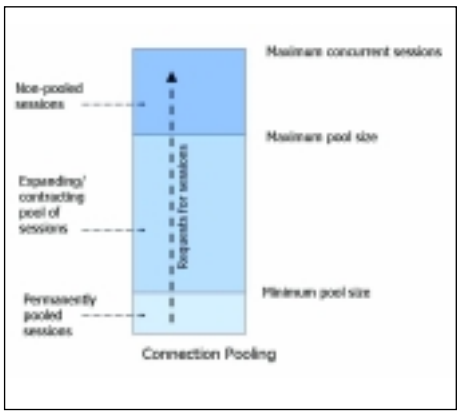
Q What are some of mv.NET's more unique features and functions?

ANSWER: As users, we all expect "immediate" response from today's applications. With this requirement in mind, mv.NET has been engineered with in-built "data fetch on demand" technology which is under the control of the developer. Therefore, the application interface response time can be automatically safeguarded even in situations where large volumes of data are being returned or where network connections offer restricted bandwidth, thus removing a significant issue from the shoulders of the developer.

Stateless application support within the product allows developers to cut development timescales by automatically persisting data and status history across application invocations.

So far we have looked at the management of data flowing from server to client and retaining user and data status. The management of the connection itself is the next point to address.

For Web and Web Services applications, Connection Pooling is employed to ensure the most efficient use is made of mv.NET and the underlying database licenses. mv.NET's session pooling and sharing management provides a sophisticated environment for striking a balance between the need to minimize database license consumption and the need for rapid connection establishment. The following diagram summarizes the key aspects of mv.NET's session management:



The three thresholds on the right allow the IT manager to both guarantee a minimum number of database connections while at the same time allow the connection pool to grow in times of high demand. They are also able to fix absolutely the maximum number of database licenses that an application can consume.

We were also asked, "Can you reduce the number of database licenses we need to allocate for rich client applications?" Plus, I guess, there was also a hidden agenda of "Can you reduce the number of mv.NET licenses we need to buy?" The answer is "yes" to both questions, with one proviso. Develop rich client applications using optimistic locking and full use can be made of Session Sharing in mv.NET. We envisage user ratios to mv.NET run time sessions and database licenses of approximately 3:1 by using this feature. Now who said that MultiValue-based applications could not compete on price per seat!

Some of the points I have highlighted here have not gone unnoticed and we have already attracted several early adopters in this market across to mv.NET with the minimum of effort.

Q What's new in this latest release coming out in May 2005?

ANSWER: Release 2.0 is a significant step forward in the life of mv.NET. It brings the production release of Adapter Objects — a full implementation of an ADO.NET managed data provider which, when combined with configurable dynamic data normalization, brings MultiValue databases right up to the same level as SQL database implementations within the .NET environment. With this full implementation — and we really do mean FULL! — we provide comprehensive read and write functionality together with transaction boundaries, all based on true optimistic locking architecture. As with Binding Objects, we've also created a rich set of Visual Studio add-in technology to complement the core functionality of Adapter Objects.

Like all new releases we have fixed a few bugs and tidied a few more things up under the hood, but no fundamental changes were necessary to Core Objects or to Binding Objects. One area, however, where we have improved things is for the C# developer where the demonstration account, SOP, is now available in C# as well as VB.NET.

Q Can you give an example of a company that has implemented mv.NET — why was mv.NET chosen as the solution and how is it being used?

ANSWER: There are numerous, but one of the most interesting is that of Drexel Management Services, a major East Coast Reseller of jBASE, D3 and mvBASE, who chose mv.NET for all future .NET development for their clients. Drexel's president, Drew Conboy, looked at other data providers in the MultiValue database market, but felt that mv.NET was the most feature rich. Drexel required a product that was fully MultiValue enabled so the fact that mv.NET provides that and allows the developers complete flexibility with its total integration with Visual Studio .NET was a major selling point. Also of importance in the decision was the fact that mv.NET works with all the MultiValue databases and provides a single development and deployment environment for all of Drexel's diverse customer base. Drexel's first customer has a Web application using mv.NET on D3. **is**

For more information, please visit www.bluefinity.com.

Taking Control of Your Printer with OIPI

Continued from page 22

```
tableHead = "Quantity,Description,Unit Price,Ext Price"
table = ""
table<1> = "1,OpenInsight Printer Interface 2.0 -- Special limited time introductory price,$345.00,$345.00"
table<2> = "2,Second line item here,$1.00,$2.00"
table<3> = "1,Third line item here (no charge for this bonus item),$0.00,$0.00"
For I = 4 to 30
  table<I> = "1,Test line item here":(i-3):",$0.00,$0.00"
Next I
table<31> = ""
table<32> = ",Merchandise Total,,$347.00"
table<33> = ""
table<34> = ",Tax,,$12.00"
table<35> = ",Shipping,,$5.00"
table<36> = ",Handling,,$5.00"
table<37> = ""
table<38> = ",,Sub Total,$369.00"
table<39> = ""
table<40> = ",,Total,$369.00"
convert "," to @vm in table
convert "," to @vm in tableHead
x = Set_Printer("ADDTABLE", ">1080":@VM:"<4680":@VM:">1440":@VM:">1440":@FM, tableHead, table, RGB(192, 192, 192), "", "", TB_BOX_ROWS)
```

*Terminate this printing session.

```
x = Set_Printer("TERM", 1)
Return
```

Conclusion

The OpenInsight Printing Interface provides developers with the ability to create sophisticated high-quality output without having to learn PCL5 / PCL6 code. With the OpenInsight BFS to U2, developers using UniVerse and UniData can utilize the power of OIPI to gain control of their printer output. **is**