Breathing New Life into MultiValue by Moving to OpenQM

An interview with Justin Orton, Technology and Architecture Lead, Zerion Group

Tell us about Zerion Group and your history with MultiValue.

Zerion Group began as a training and consulting organization for the Epicor Eclipse ERP system. Epicor's Eclipse customers are primarily wholesale distributors for a variety of markets—mostly plumbing, HVAC, and electrical distributors. Naturally, those customers are our customers as well.

Over time, Zerion Group evolved and began building products, like our EDI Dashboard and Saaplicity Field Service software, which help our customers run their businesses more efficiently. When I joined the company about four years ago, my focus was supporting the EDI web applications that integrate with Epicor Eclipse. These custom applications integrate with our MultiValue database, which was on the UniVerse platform at the time.

Why stick with MultiValue? Why not something newer and flashier?

MultiValue is used in more places than people give it credit for. People tend to keep quiet about it, and that's kind of a shame, because it is an incredibly powerful database. I've been working with Pick for about 30 years, and I've consistently found that MultiValue is superior to virtually all other database styles. A relational database simply cannot compete with it. To get a relational database to match the performance of a MultiValue database requires more hardware, more code, more complex queries, and a lot more administration. That doesn't scale well. With MultiValue, you can do a lot more with fewer resources. You can build more quickly, and your application can be faster. I've found that a .NET application talking to a SQL database drags compared to exactly the same application running against a MultiValue database, hosted on a smaller hardware platform. One Pick server can far outperform two SQL servers clustered together, with all the cost of ownership that goes with that. Historically, the biggest downsides to Pick have been the front-end interfaces and the inability to connect to modern systems.

A lot of companies today are bringing together multiple best-of-breed solutions. They want one application for general ledger and accounting, another for inventory management, another for eCommerce, etc. And they want to tie all these together. So-called "modern" applications can do that because they talk to each other through APIs. The MultiValue community has been slow to adapt to that mentality.

But, thankfully, there are now ways to build modern apps with MultiValue databases. OpenQM is a really good example of that.

ZERION + Open QM

What made you switch to OpenQM?

I've worked with UniVerse for years, and I think it's a very good product. But we were looking for a more modern platform that could keep up with the direction we are taking our product portfolio. We are working on an exciting new application, and we plan to broaden our application scope to other APIs on other platforms. There are ways to use RESTful web services with our previous infrastructure,

but it would have required a huge investment in terms of licensing and hardware, and it would have still left us with a somewhat monolithic architecture.

When we made the decision to start looking for an alternative, we knew we wanted to stick with MultiValue. From there, we looked at every MultiValue option out there. Early on, two options led the pack: jBASE and OpenQM. And suddenly it wasn't about cost anymore. We realized that with OpenQM, we could get many more features already built into the product that we could benefit from immediately and would otherwise have had to build ourselves.

OpenQM has REST integration. It's right there. Nothing to be built. Nothing else to buy. It also has JSON capabilities native to the product—not extended add-ons. And it features exception handling and object-oriented development, which, as a .NET developer is huge because most applications today are architected in an object driven fashion. Object classes in our backend translate seamlessly to JSON objects the web side can consume. This really fits what we are trying to do.

When I realized we weren't going to have to custom build these things, I thought, I don't even care if it's the same amount of money. The benefits of what we are going to get are that significant.

OpenQM turned out to be a far lower total cost of ownership than we expected. And because the environment is familiar to our developers, our conversion process was easy. We basically just exported the data and converted it. Nearly 90% of our code stayed the same.

I know it's early days, but what benefits have you seen so far?

I did a case study on one of our applications that integrates with Eclipse to better understand the efficiencies of OpenQM. The whole purpose of this application is to capture data from various sources, map meta-data from it into a reporting database in QM and stage it for delivery into Eclipse. Custom code on the Eclipse side makes requests asking for data and brings in these messages for processing. Sounds like a very simple process, right? With our previous environment, that process required significant code on the Eclipse side to structure XML SOAP messages, and leverage UniBasic's HTTP extensions to make calls to a custom web service built in .NET. That's a custom .NET old-school SOAP web service that, in turn, uses a bunch of UniObject code to talk to UniVerse to process the requests.

OpenQM's ability to support REST APIs means the custom code in Eclipse has been reduced by nearly 80%. Instead of building SOAP messages, it's now a simple REST API call that receives a compact JSON response.

> ***"We reduced the code by 80% in less than 30 days with Open QM."***

As for that cumbersome .NET web service layer? We no longer need it, or the UniObject code, or the server it was hosted on! OpenQM receives the response directly and runs practically the same program that we ran in UniVerse but with some object-oriented enhancements that completely encapsulate the process. That's removed an entire layer of application support and training for us. And it eliminated an entire web server. We simply don't need it. It was a real "wow" moment for me.

Moving to an object-oriented approach has also enabled us to build Unit Tests in QM that can fully test each class. This modern approach to class development means making changes to data objects is safe and easy with each class being testable without having to walk through an entire application. Try doing that with a traditional MultiValue application! With OpenQM, out of the box, we dramatically simplified management while enhancing our messaging engine.

What has it been like working with Zumasys?

Obviously, we are very excited about OpenQM, but we were especially excited about the support we've received from Zumasys—even when we were just in the discovery phase. You always sort of expect great support during the sales cycle because they want to close the deal. But I was unprepared for the level of engagement we received. It didn't matter who we contacted; everyone was made available to us. Within eight hours of my original sales inquiry, I had a development environment up and running. I was being invited to GitHub and being sent samples of code.

It was a phenomenal experience that has only continued now that we are a customer.

By the way, Martin Phillips is amazing too! I was doing the analysis on converting one of our products, and there was feature we needed that QM didn't have. I told Martin that I thought I had a workaround. It was going to be a slight performance hit for us but not a showstopper. The next day, I got an email from him saying, "I've added it. I'm testing it now." Three days later, an email went out with a patch for that function. In four days, start to finish, the feature was in the product. And we hadn't even bought a license yet. That just blew my mind.

"QM is backed by a great company that believes in the future of MultiValue."

Why do you think using NoSQL is exciting for MultiValue?

When more customers start to see these types of results with products like OpenQM, it's going to breathe new life into the MultiValue community. I'm really excited about it, and I know the folks at Zumasys are too. They're investing a ton into tying their product into the modern ways of doing things like object-oriented programming, Docker containers, source control. Zumasys talks "modern development practices," and that will resonate with modern developers who find themselves working against a Pick backend or looking for something different to build the next big application with. With OpenQM and Zumasys, we know we're not just getting a great product, we're also getting a great partner who believes in the product.

For years, SQL has been the big guy in the room. They're everywhere. But NoSQL databases are getting more popular. And a MultiValue database is really a NoSQL database—particularly OpenQM. QM has collections where you can drop off complete XML or JSON documents and process them as you wish. You can query against them; you can write code against them. A company that is marketing that ability correctly has the opportunity to usher in a new golden age for MultiValue as people look for alternatives to SQL.

Are you excited about the future of MultiValue?

I see QM and jBASE giving MultiValue a whole new lease on life. And I'm excited about that. It's clear that Paul Giobbi and Zumasys feel the same way, which is why they're investing in it. They are tying the product into all the modern ways of doing things: object-oriented programming, Docker containers, GitLab, and GitHub. And that's very, very exciting. We don't just like QM because it's a good product. We like it because it's a good product that's backed by a great company that believes in the future of MultiValue.

About Justin Orton

Justin is a Consultant at Zerion Group and serves as Lead of Technology and Architecture. He has extensive experience in the field of systems integration and Electronic Data Interchange (EDI). In his role with Zerion Group, Justin provides EDI support to clients, internal interface development, and systems integration development. Originally from the U.K., Justin developed interfaces for government, manufacturing and wholesale distribution companies, then moved to the U.S., where he has managed EDI support teams for Hughes Supply, HD Supply, and Sonepar USA.

Justin's knowledge of wholesale distribution and the Eclipse platform allows him to help clients gain additional value from their IT investments. Justin joined Zerion Group because the company brings targeted expertise to the industry, providing timely and affordable solutions to companies that need a faster-than-average turn-around. Justin attended Norwich College in the U.K., graduating with qualifications in IT and business management. Before moving to the U.S., he also spent time working as a firefighter in the U.K. Justin loves to write, and recently published a novel. In his spare time, he likes to read, write, and travel.

E-mail justin@zeriongroup.com
Learn more: zeriongroup.com

For more information on OpenQM, please visit **www.openqm.com**