Event ID: 369455
Mothercare Group PLC Takes Their ESB to the Next Level With IBM WebSphere Message Broker version 8

[Karen Bannon]: Good morning, good afternoon, or good evening depending on where you are in the world, and welcome to today's webcast, Mothercare Group PLC Takes Their ESB to the Next Level With IBM WebSphere Message Broker version 8, brought to you by InformationWeek, IBM, and broadcast by United Business Media LLC.

I'm Karen Bannon, today's moderator. We want to make sure this event is as interactive as possible. So, I'd like to make just a few announcements before we begin. This webcast is designed to be interactive between you and the presenters. You can participate in a Q&A session at any time by asking questions during our presentation. Just type your question into the Q&A widget to the left side of the slide area window and click the Submit button. You may adjust the area of the widget such as the slide area by clicking on the lower right hand corner of the widget's window and dragging the mouse. Slides will advance automatically throughout our event. If you experience any problems with the program please refresh your browser or close your window to re-launch the presentation. You can also visit our webcast Help Guide by clicking on the question mark button on the docking bar on the bottom of the console window. And now, on to our presentation, Mothercare Group PLC Takes Their ESB to the Next Level With IBM WebSphere Message Broker V8.

We've got two knowledgeable speakers joining me for today's presentation, Ant Phillips and James Blackburn. Ant Phillips is a development lead at IBM's WebSphere Message Broker product development team. He is based at IBM Hursley in the UK and is responsible for a range of technologies in Message Broker which are used to build reliable and scalable applications. Anthony is also the Healthcare and Life Sciences Architect in Message Broker and leads the team who deliver the healthcare connectivity pack.

James Blackburn is the Chief Architect at Mothercare Group PLC, a UK-based global company synonymous with children and parenting through the Mothercare and Early Learning Center brands. As IT senior manager, he is responsible for all enterprise applications in use within the group and the leadership of architecture and software teams alongside the development of IT application strategy.

And with that, I'd like to hand the presentation over to our speakers.

Anthony Phillips: Thank you, Karen. This is Anthony Phillips. As Karen said, I'm an architect in the Message Broker development team in the UK. I'm going to spend about 15 minutes talking through some of the highlights of Broker version 8, which is being released at the start of December, in a few weeks' time.

15 minutes, of course, is nowhere near enough time to describe everything in this release. So, this will be a very quick run through, but hopefully it will give you some of the headlines and some areas of interest you might like to follow up on our web.

A very quick refresh on Message Broker. Message Broker is an integration engine. It connects applications and devices together so that they can share data. Looking back some 10 years or more to the start of Message Broker, our idea of a message was very limited. Essentially, taking messages off of a queue, doing something to them, a transformation, and then, pushing them back onto a queue.

Since then, of course, Message Broker has grown enormously in its capabilities so that it can deal with much larger sets of transports and technologies, everything from databases, Web services, queues, to packaged applications like SAP, Siebel, and PeopleSoft.

Of course, the message formats have expanded as well. Common formats like XML, CSB, COBOL through to industry specific formats like HL7 in healthcare and SWIFT in financial services. We focus very strongly on ease of use with our graphical data flows and also our patent-based technology, which provides out of the box configurable solutions for common technical problems. Also, the ability for anyone to create their own patterns using our pat-morphing technology.

Last, but certainly not least, everything we do has two personas, how solutions are developed and how they are administered. And as we walk through to the version 8 content, you'll see some major enhancements in both those areas.

Contents at a glance -- well, I'm not going to stop on here for too long as we'll be talking about most of this in the next few slides. It's there though if you want to get the slides after this call.

First big headline feature, Web administration of Message Broker. In Broker version 7, we provide two main tools, our Eclipse-based toolkit, for developing applications and Message Broker Explorer NBX to administer Broker.

In Broker version 8, we now have Web-based administration. You can perform all those common tasks like deploying applications, starting and stopping flows, creating execution groups all through a Web browser with a zero footprint.

We test against all the major browser manufacturers, Firefox, Chrome, and Internet Explorer, of course. As you'd expect, access to the Web applications is security enabled both at the transport level, through SSL, and also through application level security. Although the Web administration is a feature of Broker version 8, it can also be used to administer version 7 Brokers.

Record and replay. The question to ask is what happens when something goes wrong in your flows. Perhaps, for example, a transformation failed because a message contained some unexpected data. Record and replay picks up from that point onwards. It allows an administrator to capture that data and store it in a database. The data is captured using our existing event monitoring and capture capabilities. And any of our supported databases can be used, such as Oracle, SQL Server, and DB2 amongst others.

Through our Web application, an administrator can review the recorded event, edit the data, and replay the messages back through Broker. And importantly, there's search and filtering in there as well. As an administrator, I can find the messages that relate to a particular customer, an invoice, or search on whatever business data is relevant to me.

DFDL, Data Format Description Language. How do I model my data? Well, in the XML world, life is straightforward. Almost without exception, I use XML schemas. In the non-XML world, it's a little more complicated. And really, just about every product, both inside and outside of IBM has a different way of doing this.

For some years ago now, IBM helped to set up a new standard called DFDL, which provides a consistent way to model non-XML data, which is a huge remix if you think of the vast range of non-XML data out there. All the industry formats, the X12 standards, through to CSV data, COBOL copybooks, and, of course, user defined formats.

As a developer, I can now create DFDL schemas, as they're called in Message Broker, and all our nodes now support DFDL just like we support other domains like XML and data. We've built a new DFDL editor where I can just design and test my DFDL models. And I reiterate that last point, design and test. I can test my models within the editor without having to deploy them to Broker. Very fast turnaround time for that edit and test cycle.

Graphical transformations. Another major headline release feature in version 8. An all new graphical mapper. We've had a graphical mapper for many releases, of course, but there were some aspects to it which we weren't completely happy with. For example, we believed performance could be better. We also wanted to make it much simpler to use. Simplicity really reflects the constituents of people who use the mapper.

The graphical mapper often appeals to people who are relatively new to Broker because it has a low barrier to entry, with a very visual drag-and-drop metaphor. With the new mapper, we've made huge steps forward both in performance and also usability. And not just that, but maps are now stored in a standard file format called MSL, Mapping Script Language.

The MSL maps are simply deployed directly to the Broker runtime. You can also take those MSLs and use them in other products. That will increase over time as the new common graphical mapper is embedded in more and more products.

.NET. Perhaps more than any other feature in version 8, this has captured the imagination. The ability to run code for .NET in Message Broker. Broker can load the Microsoft Common Language Runtime, the CLR, and exit to .NET code just like it can load the JVM and run Java code. And to be absolutely clear, this isn't because we love Java any less. Java is still very important to us. It's just simply that our clients, our customers, also have investments in .NET code and need to use that in connectivity solutions.

How do I build my .NET code for Broker? Just like I do today, Visual Studio. I still develop my message codes in the Message Broker toolkit, but all .NET design, development, and debugging is done in Visual Studio. And to this end, we provide a set of templates for Visual Studio to help developers get started quickly. These templates support a variety of .NET languages including Csharp and VB.NET. I'll also add that we've done a large amount of work around performance so that .NET integrated into Message Broker is incredibly fast as we transition to and from the CLR.

We've covered a selection of the big hitters in version 8. Let's just spend a few minutes going through another 10 or so of the headline features very quickly. First up, patterns. We introduced patterns in Message Broker version 7, a different way to build applications. Parameter driven, production ready solutions.

In version 7, we provided a range of built in patterns that solved common integration problems. For example, putting a Web service front end on an in-queue application. And then, in version 7 6.1 and 6.2 through last year, we added the ability for anyone to create their own patterns, called Pattern Authoring.

And now in version 8, we've added the third generation of this technology so that even richer patterns can be created. We also support round-tripping so that your modifications to a pattern are maintained even if you recreate the pattern instance project.

You'll see we also added a new built-in pattern for .NET. The pattern creates a Web service from a class written in .NET. The pattern generates the message flows, the WSDL, the XML schemas, and everything so that calls to the Web service are routed through to method calls on the underlying .NET class.

We've added new file processing capabilities with Sterling Connect Direct nodes. Every release, we move our ever popular file processing option forward. We've always been able, of course, to process files on local or mounted file systems for a long time now. Also, connect FTP and SFTP hosts, and more recently, we can send and receive files through FTE MQ file transfer edition. That provides both guaranteed once and only once delivery of files.

With Sterling Connect Direct, we can process files directly on the Connect Direct network from our business partners coming from business partners or other data sources. Also, for application developers, we've opened up database support so that any UNIX ODBC driver can be plugged into Message Broker. We do ask if you want to make use of this feature, do come and talk to us, and we can provide you with good support.

Applications and libraries really improve the end-to-end experience in Broker solutions. They allow me to design, deploy, and manage solutions as a whole unit. The runtime sees an application as one atomic solution which can be started, stopped, deleted, and so on. Libraries allow me to package content for re-use so that it can be easily shared between applications.

Last, but not least, rather like the file nodes, we also leave our Web services story boards in every release. Our aim is always to make Broker the best platform for Web service solutions. In Broker version 8, we're introducing WS-Reliable Messaging support. HTTP is by its very nature an unreliable protocol. WS-RM has reliable message delivery, and I can configure a quality of service through WS-RM. For example, AtLeastOnce, AtMostOnce, or ExactlyOnce message delivery.

Let's change paths quickly to look at some of the Administration enhancements in Broker version 8. Platform support in 65-bit across the board for server platforms. We also support 32-bit Windows and Linux, just because of their prevalence on developer machines. The MQ prerequisite remains the same, 7.0.1. The upgrading is straightforward. We'll also support MQ 7.1 as a standard lifecycle enhancement, Java 6, and Microsoft .NET CLR version 4 developer platforms. And probably know that Java 7 was released recently, but it didn't quite make it in time for this release.

Like all Broker releases, we support N minus 2 migration, meaning I can migrate directly from the two previous releases. And so, for version 8, that means version 6.1 and version 7. There's no change to our strategy in this space. You can upgrade your Brokers in place using your MQSI migrate components command or you can install version 8 side by side on the same machine, on the same server with 6.1 and, or version 7. You can then incrementally move your applications across to version 8, if and when the time suits you.

Management APIs. We have had for a long time a Java API for managing Broker called the CMP API. In version 8, we've added a REST API for managing Broker. All the common applications are available through a REST state. Deploying solutions, listing applications, starting and stopping flows, and so on. The reason we've done this is to allow you to manage Broker from any platform, whether that be a mobile application, or a mashup, or a situational application running on a Web server.

Last slide covers some more administration enhancements. A new edition called Express Edition brings a version of Broker to the small enterprise and departmental server world. It has a slightly reduced set of capabilities, but importantly includes a .NET mode. And keep in mind as well it's the same code running in Express Edition as runs some of the world's most demanding applications in Advanced Edition. Those are applications running tens of thousands of transactions per second sustained and which simply cannot fail. It's the same code in Express Edition.

Workload Deployer, the next generation on from WebSphere Cloudburst enables easy deployment for virtual images for cloud infrastructure, both on AIX and x/Linux. Most important of all, the deployment image includes not only the operating system and the middleware in queue in Broker, but also all your own application content. Furthermore, you can customize the deployment process with your own configuration scripts.

Database administration. Those two enhancements that mean so much. Fast First Message processing. The first message has no startup cost as the database connections are already established. And immediate reconnect to the database in the event of a connection problem.

Last, but not least, Activity Trace. This is a new administration capability to understand what Broker is doing. Most important of all, it provides messages with meaning for System Administrator. For example, writing message to a queue, file ready for processing, updated database failed. And as you'd expect, the Activity Trace messages also have context information explaining more about the message. The queue name, the file that's ready for processing, and so on. We've designed Activity Trace to be very light weight so it can be left permanently running. If a problem does occur, then the information has already been captured, ready for diagnosis.

That's been a very fast run through of the headline features in Broker version 8. I'm going to hand over to James now, who's the Enterprise Architect at Mothercare. He's going to explain a little bit about his business and their experiences with Message Broker version 8.

James Blackburn: Thanks Ant. As already been discussed, my name's James Blackburn. I'm the Chief Architect here at Mothercare PLC. I'm just going to walk through what we've done with our ESB here and why we can take it to the next level with version 8.

For those of you who haven't been introduced to Mothercare before, we are a global supplier of children's wear to not only our UK business, but also over 1,200 stores in over 55 countries. We operate through three brands, Mothercare, Early Learning Center, and our social networking site gurgle.com.

We have annual sales of GBP 1.2 billion and maybe 22% of our annual sales come from over the Internet. In addition to all of our stores business, we've got offices worldwide, in the UK, India, Hong Kong, and China. And we also have a global supply chain with distribution operations not only in the UK, but also in China, and India.

Our mission is to be able to support the needs and aspirations of parents for their children worldwide. In order to support that mission, our systems need to be stretched to do quite a number of things. So, we need to support everything from our core merchandising systems, which is dealing with price and product management. Channel Focused Systems from e-commerce to stores. Sourcing systems; e0verything from factory right through to warehousing. International and wholesale operations and our UK supply chain. All of those discrete systems get plumbed together using our WebSphere Shared Messaging infrastructure underneath.

In order to support all of those systems, we centered our technology around three core strands. The IBM I or AS/400, Microsoft SQL Server, and our IBM WebSphere technology.

So, why does Mothercare need an ESB? Some of our core system challenges meant that IT was becoming a disabler to our business growth. Our core merchandise system has something in the region of 10 million lines of code, which is pretty similar to an operating system. We have 100-plus bespoke interfaces. Our business logic and enterprise transport is all combined. Adding a single application therefore meant that we had multiple application impacts.

What that meant was that a number of our core systems were all therefore interrelated to each other. As I said earlier on, if we change one of them, a number of other impacts were made. Whereas what we wanted to get to was our future state integration solution where we could hang all of our applications off our service bus. And therefore, if any one system needed to be changed, its impact to all the other systems was minimized.

But this solution hasn't just happened overnight. This has taken a number of years to develop. We started working with both IBM and Microsoft in 2006 when we did a proof of concept between both Message Broker and BizTalk. In 2006, we chose the IBM WebSphere Message Broker product to power our Enterprise Service Bus. We commissioned this in 2008 with a development of a number of our core application services.

Our first application went live in Q1 2009. Since then, we've changed about -- 3 IT and business change projects per year have been migrating into the service bus.

How we achieved that is through establishing a number of principles. We've connected a number of applications, our core merchandising and merchandise planning system. Our finance system, e-Commerce, and fulfillment.

And developing all of those, we've put together a set of principles, which means that we transformed into an OAGIS or ARTS-based canonical data format. We build and install applications based on adapters for specific integration. What that means is that we actually say that an individual application has its own execution group and a set of flows associated with it. This means that we can establish our core service providing generic transport and routing with our business services providing any choreography.

By establishing each application in these individual execution groups, we can make sure that our applications end up loosely coupled from each other. Our application configuration can therefore be managed by the business unit rather than our core IT function. Which means we can get closer to our goal of having one piece of business data input into one system but shared by many.

How do we achieve that logically? As I said, we use execution groups to separate off our applications, so we developed ourselves a core merchandising system adapter, which uses MQ. And then, we transform that into an economical data format. And then, we push that through to an application adapter, which transforms again to the interface specification. Therefore, the vast majority of data always gets transformed twice and all the local configurations looked up from an audit and routing database.

In order to achieve this, what do we actually use? Our software is built on a number of WebSphere technologies. So, we us MQ, as we've already described for our Enterprise level transport. We use version 7.0.1. We use IBM WebSphere Message Broker to transform and do a lot of our main core canonical data format mapping and our application adapters. We use version 6.1 for that. We also use the WebSphere Partner Gateway version 6.2 because it is required for some of our business-to-business external comms. But we also use Tivoli for our system control and monitoring. Our entire estate, for example, is -- over 43 queue managers and I have no dedicated MQ administrative support. Everything is managed through our Tivoli infrastructure.

If we've got all this environment here and we're migrating 3 applications a year. We're doing this dual mapping. We've got all the benefits that we can have automated routing and I've got Tivoli, why do I need version 8? For starts, I'll obviously get the benefits realized under version 7. I don't have to use DB2. I've got simpler installation. And quite importantly for us, I've got patterns.

But really for me, version 7 was very, very difficult for me to parse in and push in to sell to the business because a lot of these DB2 patterns is really benefits for IT. What version 8 allows me to do is bring on some of those benefits from version 7 and bring those to business realized benefits. We've got Microsoft environment support. We've got the Message Mapping user interface. I've now got a standardized Integrated Development Environment with many of my tooling now working in the standard Eclipse frameworks. And a number of administrative enhancements.

What I'm going to do now is step through some of the benefits that version 8 is going to give us and how we are currently implementing those.

Microsoft Integration. Database enhancements as Ant talked about, these should provide better message flow support for SQL Server, the pre-emptive connection, connect, and reconnect. But more importantly, what we're working with is the .NET support. That allows us to natively integrate with the Microsoft Business Stack, most notably Microsoft Dynamics. And a .NET node for integration with Visual Studio.

As I've already described, one of our key technologies that we use is Microsoft SQL Server in .NET environment. So, we really need a seamless integration between our Message Broker and our Microsoft environment.

What I'm going to do now is walk you through one of the use cases that we're currently working on here that allows us to take some existing message flows and apply some .NET nodes to them.

So, I've got an existing message flow which takes master data from a Legacy AS400 application into a finance application via Webservices. Interestingly, this is originally a Message Broker version 6 product, but as I've said, this is now 6.1. So, we've got our Merchandise Management Master System that's generating master data such as suppliers, customers, and products. And I need to replicate this data into our finance system.

Following our standard principles, that's a two step process. So, I take an input message flow that takes data from an MQ, that's common wire format for MQ, and generates an OAGIS message format. A separate message flow then process OAGIS message format into a Webservice call into our financial application.

But what I need it to do is I need to extend this flow to not only write into a external SQL database, but also make some in-line calls to Microsoft Dynamics. So, I've developed some new message flows to route to the database and .NET. Because we've approached this loosely coupled methodology, we've been able to take our existing message flows, the output to the OAGIS format, and the existing message flow carries on. It's none the wiser.

We then put extra entries on our routing database, which then duplicates the data automatically to a number of other flows. So, these new message flows are being developed to route to SQL database and into .NET.

If we have a look at the .NET flow in detail, what that allows me to do is take a standard MQ input and then route that through to a ESQL compute node. And then, we pass it through a series of .NET nodes so it allows us to log on to the application to manipulate the data into the .NET application database. Then, we log off. Then, we process the message through to our audit.

This message flow uses not only ESQL and the .NET node to interact with the .NET application. And again, this is completely independent of the previous message flows. Although, it's still using the same message source.

This is what Ant was talking about earlier on when he mentioned that we really need to, as a customer, we're working with a lot of our .NET applications. We really feel that version 8 allows us now to move forward and seamlessly connect with a number of our .NET instances.

Other features in version 8 that we're using, that Ant has already mentioned, the Graphical Mapper. Obviously, like many other customers, the original message map, although it was cumbersome to use, did actually do the job that it needed to do. But our current design process means that we tend to build message models and maps in a spreadsheet to pass to our offshore teams to develop.

What this allows me to do now is our message mapping user interface should allow my architects and designers to build a message map to directly pass offshore. This should lower the number of mistakes and issues that could occur. And if there's any changes to be made, we don't need to make the change on the spreadsheet, check it in, change the design. We can just change the message map object. We can then check that in. And then, the flow should update.

It will allow us to have much more of a dynamic interaction with our offshore teams. And it will also allow with a standard message format eventually, to exchange those message formats with our partners.

In addition to .NET and the mapping user interface, we've actually got some changes to our IDE, as I mentioned earlier on. Our aim here is to have a single IDE for a WebSphere developer that will allow support of unit testing. But because it uses Eclipse version 3.6, this now works better with a lot of our tooling. There's better integration with SubVersion. We've had challenges here where under 6.1 of Broker, sometimes the Subclipse and SubVersion interfaces don't work quite as we planned. But now, we have seamless integration with SubVersion. So, we can check in and check out directly from our toolkit.

We've got integrated database connectivity. So, we don't need a separate application or other JDBC plug-ins to the IDE. Also, important to us is we've also go integration to our tooling that allows us to do tasking out to our offshore teams through the Jira product Atlassian provider plug-in that works with 3.6 of the integrated IDE. And that means that individual developers can be working on a single task and a single message flow, checked in, and with the database all in one individual perspective on the screen without either having to switch perspective views or even applications.

It's getting ourselves close to my aim of not being able to move around between different applications and get to the point where a developer can work in a single application. Ant also mentioned that there are a number of other things in version 8. Particularly, some around the admin enhancements. And again, I've just picked out a couple of things that we've been working with here on version 8 that gives us some of the benefits.

We can deploy flows into a stopped state. That means we can deploy solutions much earlier. And a solution go-live is then just a simple case of starting up message flows as opposed to having to do automated development and implementation between a number of other third-party teams.

In particular, we also have enhancements to the activity logging. So, this builds on what was before in version 7. It allows for external views of the Broker events. So, we don't need to use APIs or the configuration manager repository viewer to actually figure out exactly what our Brokers are doing and who's deployed what, when, and where.

In terms of version 8, most people probably like me had a view that, okay, to be honest, version 8, yes, it's great. Gives me a whole load of new functions, but I'll wait until the first fix pack comes out. Or I'll wait for somebody else to find some of the issues. But to be honest, we haven't really found that with v8. We've been working on v8 with IBM since the alpha release, since Q1 2011.

We haven't really encountered any significant stability problems even with the alpha release. So much so that one of our projects set for deployment later next year, we've actually already started using version 8 in beta in one of our projects to start live coding of. We've been able to deploy version 6.1 flows directly without modification and without even having to do an MQ upgrade. So much so that we're confident that we can go straight into version 8 production migrations Q2 next year, without any significant overheads or changes to any of our solutions.

The primary reason why we selected Q2 as opposed to Q1 when the product is available at the end of this year is I've got a number of other in-slide projects that migrate in Q1 that have already been started coding before we even had the Alpha release of the product. So, we're going to deploy those projects first into v6.1. And then, we'll do an upgrade before our next set of projects are implemented.

I'm now going to hand you back to Karen. Thank you.

Karen Bannon: Great. Thanks so much, James. And now, before we begin today's Q&A, we'd like to ask everyone to fill out the feedback form that's opening on their computer. To complete the form please the Submit Answer button that's on the bottom of the page. Thanks so much in advance for filling out our feedback form since participation in this survey allows us to better serve you.

And with that, I think it's time for our Q&A. I did just want to call out that we are going to be pushing out a couple of resource slides right here for you. These will give you a little bit more information about today's presentation and the Message Broker v8 solution. So, it might be good and handy to have during Q&A to look at while we're discussing the presentations.

And with that, let's get started with our Q&A. The Q&A is actually an excellent opportunity to ask questions and start a dialogue with IBM about your own IT infrastructure or concerns you might have. As a reminder, to participate in the Q& A, just type your question into the text box located below the media player and then click the Submit Question button.

We have had some questions coming in. So, let's get started. Peter would like to know, “I've heard several file types can be directly deployed to Message Broker in v8. Can you explain what this means?” Ant, do you want to try that one?

Anthony Phillips: Yeah, sure. So, that's a good question. And this is a big change for Message Broker. In previous versions of Broker, all file types had to be -- well, either imported or at least processed by the toolkits before they were deployed. And in Message Broker version 8, I can now take those source files and deploy them directly to the runtime. And there are lots of good examples of this.

For example, WSDL, Web Service Descriptions, XML schemas. We can also source deploy ESQL scripts or message flows themselves in the form of deployable subflows. So, this is all about simplification, really making our deployment story much more simple and consistent. And it's also about making Message Broker easier to administer.

Karen Bannon: We have a question here. This question is from Christopher. Christopher would like to know, “Is the healthcare connectivity pack supported on Message Broker version 8?”

Anthony Phillips: Yes, absolutely. The background for this is a new product we released in May this year called the Message Broker Healthcare Connectivity Pack. And the product brings together technology we've had for many years in a healthcare product that sits on top of Message Broker. The key point being that Message Broker is a general purpose integration engine which gets applied to many different industries. Retail, transportation, financial services, and, of course, healthcare.

And the Healthcare Pack brings all that together along with content, which addresses key scenarios in the healthcare industry. For example, connecting clinical applications with HL7. So, yes, the Healthcare Pack is absolutely supported on top of Message Broker version 8 right from day one.

Karen Bannon: We have another question. This is about the new graphical mapper. This question comes from John. He'd like to know, “Is the new graphical mapper faster than the previous mapper?” And maybe you can tell us a little bit about what's changed.

Anthony Phillips: Yes, sure. And that's a very topical question. The short answer, yes, it is. It really addresses one of the weaknesses that I was talking about earlier in the slides. This is a weakness that we've seen in the previous versions of the graphical mapper.

The graphical mapper in version 7 and before really isn't as fast as we would've liked it to have been. When we compare it with some of the fastest transformation technologies in the product, such as ESQL and Java, and now, of course, .NET, the graphical mapper isn't quite the fastest amongst those yet. But it's getting pretty close. This is in part because of our investments in the underlying runtime technology. We've built a really efficient JIT compilation engine that executes the maps. And we're pretty confident over time, the mapper will really push our transformation technologies for that top spot.

Karen Bannon: We have another question about comparing Message Broker v7 and v8. This question is actually from Ron. Ron would like to know, “How does performance of DFDL compare with the MRM parser in Message Broker version 7 and before?”

Anthony Phillips: Yes, okay. So, let me think about it. So, broadly comparable, I think with some gains in key performance areas. The background to this is that we've had a goal all the way through that DFDL development in Message Broker to beat the MRM parser in pretty much every benchmark we could think of.

The MRM parser in Message Broker is the parser that can parse tight delimited data, comma separated text data, all those kind of non-XML formats. So, that's the kind of benchmark that we stand up against. We're not quite there yet as far as beating it on every benchmark, but the performance figures are really encouraging. Best of all, we have a clear road map that takes us forward to two times, three times, and beyond the parser performance of the MRM parser in the future.

I think for this first release of the DFDL parser, the figures are really encouraging. And looking forward for the road map, much better as well. So, good story there I think for the product.

Karen Bannon: We have a question here I think that both James and Ant could handle, but maybe we'll ask Ant to -- actually, we'll ask James to take this one first. We have a question here from Bill. Bill would like to know, “We're looking for some middleware to take care of integration between Lawson, which uses WebSphere and MS SQL and a number of vertical market products. Would this be a practical solution to handle all the interface between various applications?”

I know, James, this is a little bit what -- like what you've done. Maybe you want to address this first?

James Blackburn: Yeah, Bill. That's a great question because that's exactly some of the reasons why we chose the Message Broker product over a number of other products. Because in particular, we have quite a number of disparate applications over disparate different technology stacks with a AS/400 legacy type applications, back end. Although, we don't use Lawson here, it's a similar type of application when we move data off that on MQ, as I said earlier on. And a variety of other Microsoft SQL and Oracle products. We chose Broker because particularly for our performance and the ability of able to transform into different formats both in real-time and batch.

Karen Bannon: Ant, do you want to expand on this as well?

Anthony Phillips: Well, I think James has captured it. So, yes, absolutely. I think that's a very common use case for Message Broker is to help integrate the kind of Web presentation tier with different back ends. So, if it's whatever kind of presentation tier that might be, whether it's a WebSphere or whether it's another vendor's front end application, that's actually fine. And then, Message Broker has excellent support for a very wide range of databases.

In fact, that's something with which we’ve expanded on in Message Broker version 8 by opening up the ODBC driver manager so that any ODBC driver, within reason, can be plugged into Message Broker. So, yes, I think that's absolutely in the sweet spot really of what Message Broker does.

Karen Bannon: I just wanted to stop here for one second just to remind everyone that this actually is a great time to ask questions that do relate specifically to your own infrastructure questions and concerns. So, we can go very general about Message Broker itself or we can go a little bit deeper about your own IT infrastructure situation.

Our next question -- we actually have a follow up from Bill. Bill would like to know, “What skill set, training, or consulting services would we need to investigate to create an implementation?” And I think probably both of our speakers would have something to say to this.

James Blackburn: James, here. Yes, I mean, we obviously have been on a Message Broker journey for a few years now. And we had here some of the same nervousness when we first put it in about we've got to move to a more real-time integrated way of delivering our data. Whereas before, traditionally we're kind of mainframe batch overnight process based.

We did a bit of both really. We had the advantage that we're an AS/400 shop. So, that means that we do a lot of work in the Eclipse framework anyway that comes shipped with the product. So, transition to the Eclipse framework and the Eclipse way of working was quite easy for us.

But we did use a bit of consultancy from a number of our technology partners. More just to make sure that we were on the right lines and my vision of where I want to go with the platform wasn't too adrift of what other retailers are doing in our market sector. Just to make sure that we're not -- we implement something that's reusable, something that we can share, and something that I can swap partners around if I need to.

So, yes, we did use a bit of consultancy from IBM and our other technology partners, but most of it was our own making and our own design.

Karen Bannon: Ant, did you want to expand on that question as well? What -- ?

Anthony Phillips: I think from a product point of view, we go a long way. We really try very hard to make sure that people's first touch with the product is absolutely as good as it can be. And you see that from when -- the first points from which you launch the toolkit. You get a set of quick starts and we have a really very extensive set of samples that you can use to learn about how the product works. Those just import into the workspace and run straight away.

We also provide patterns, which is -- not only does patterns provide a very easy way to get production-ready solutions, but it's also a fantastic way of learning about how the product works and how to use it efficiently. That's one of the key aspects of the patterns that we provide in the product is that they exemplify best practice so you can instantiate the pattern. And then, what you get is a Message Broker project which implements a particular solution. And then, you can look in that project and the code that's been generated. The message flows and the scripts, and really understand how to use Broker efficiently for creating Web services, connecting applications together, and so on, and so forth.

So, there's lots of touch points with Broker to understand how to use the product. And that's just in the core product itself. And then, of course, that leads on to all the things that James was talking about, which is the kind of broader community or broader ecosystem of all the kind of consultants and training companies, which can obviously help you to build solutions as well.

Karen Bannon: We have a question here. I believe it's [Venigopal]. I apologize if I didn't get that right. They said, “You mention that Web console's available in WMB 7. Could you give more details? Is it a support pack or does it come with the fix pack?”

Anthony Phillips: Right. Yes, so, there is a -- there’s an existing fix -- support pack, I should say, which is available that provides some of the capabilities around record and replay. And what we've done in version 8 of Message Broker is to take that and build it into the core product. So, it's now a part of the supported product offering. And that addresses -- the support pack had some of the capabilities of the record and replay, but we've added quite a bit more to that.

The Web console that we're providing in version 8 has all of that record and replay, but it also has general administration capabilities with Message Broker. The ability to deploy applications, see what's deployed, start flows, stop flows, all those kind of really key capabilities. Then again, that's all built into this Web console, Web application, call it what you will.

The way we deploy that is we have a Web server that's built into Message Broker. In fact, we've had that for a very long time because that's some of the technology that we use for our HTTP and SOAP nodes. They require a Web stack, if you like. And so, we've had that for a while. So, we're simply repurposing that tool to obviously deliver Web content.

That's also sort of leads into another question which often comes up, which is can we also turn this off? And the answer is yes. We do appreciate that not everyone wants to have Web-based administration of their Message Brokers. So, for those people, we can also turn this capability off if you want to.

Karen Bannon: Sounds like it's very flexible and it definitely seems like it takes a lot of the complexity out of the equation. We have a question here now from Tom. Tom would like to know, “Is this compatible with both SQL Server 2000, and SQL Server 2005, and SQL Server 2008? All 3?”

Anthony Phillips: What a good question. So, definitely SQL Server 2008. I believe 2005 as well. Less sure about SQL Server 2000. I'll have to check the final support statement for what we're supporting for 2000. So, certainly the first two. Not sure about SQL Server 2000.

Karen Bannon: Ant, if they had questions about compatibility, is there a good way to get in touch with you or get in touch with someone at IBM that could help them out with these type of very technical questions?

Anthony Phillips: Yes, lots of different ways. So, you can either -- if you have an IBM representative that you talk to anyways, then, you can just ping them a question. If you want to get to contact through the IBM website, you can do that. Equally, there's a service channel, which is a channel we support not just for providing support around product defects, but that's an important part of it. But also, for simply answering questions. If you have questions, you can raise a service request and we'll answer that very quickly.

So, lots of different ways of getting into IBM. Whichever one suits you the best I think is the answer.

Karen Bannon: We're actually coming up to the end of our event, but we do have time for maybe 2 more questions. I have a question here from Larry. He'd like to know, “Unfortunately, we're running Message Broker 6.0. Is there any direct upgrade path from Message Broker 6.0 to Message Broker 8?”

Anthony Phillips: Right. Okay, good question. The answer is there's no direct path. So, this is something I mentioned earlier which is our strategy for Message Broker is and has been for a long time, to support N minus 2 migration, meaning that you can go from the previous two releases up to the current release.

Once we've released version 8, what you can migrate to version 8 is directly from version 7 or version 6.1. So, for someone who's on version 6, which I believe is going out of service very soon, if it hasn't already, then, the answer is to bring those -- bring your solutions into a 6.1 or a 7 workspace and rebuild them. And then, you've got something which you can take forward into version 8 directly.

As always, with our migration, we support two ways of doing that. You can either -- once you've got your solution in a 6.1 or a 7 format, then you can just deploy that directly to version 8. That's completely supported. Or if you want to, you can bring that solution into version 8 and do the migration to bring it up to version 8 compatibility if you like, and then, you can deploy that. So, either directly from previous version or bring it into 8 and deploy from 8.

Karen Bannon: Our final question of the day, this is actually coming from Don. Don would like to know, “Can I create message flows outside of the toolkit?”

Anthony Phillips: Yes, okay. There's another new feature in Broker version 8, which I didn't have time to talk through. So, in Broker version 8, we rounded out the Message Broker Java API so that it's a full API to develop message flows. Essentially, just about everything I can do in the toolkit, Message Flow Editor, I can do in the Java API. And this lets me create standalone applications and Web applications that construct message flows.

For example, perhaps I want to write a mashup, a situational application that takes some business data and creates a Message Broker implementation flow. Well, that's all now possible using the Message Broker API.

Karen Bannon: With that, I think we've come to the end of our presentation. I would like to just ask both of our speakers if they had any final thoughts to leave our audience with. Maybe Ant?

Anthony Phillips: Well, I think the call to arms is to get hold of this as soon as you have some spare time, perhaps over the Christmas, New Year vacations, and try it out. We're obviously very keen for you to try it out and give us your feedback. It's something which the Broker team -- perhaps their biggest strength is we really do listen to what all of you need in a product. Everything we do is driven by customer requirements. So, please, do give us your feedback.

Karen Bannon: Thanks. And James? Do you want to leave us with a final thought or two?

James Blackburn: Yes, I think it'd really echo what Ant is saying. In v8, particularly things like the .NET integration, the consideration that's been to the Web consoles is, we've got a product here that we're playing with, and using, and testing that is going to be able to allow us to deploy solutions based on it. So, again, directly what Ant said. We've got something here that the IBM labs have released that is customer-facing and -focused.

Karen Bannon: Well, thanks to both of our speakers and thanks to everyone out there for attending our event. For more information about today's webcast, please visit the resource links that opened before Q&A began. Within the next 24 hours, you'll receive a personalized follow up email with details and a link to today's presentation on demand. Additionally, you can view today's event on demand by visiting www.informationweek.com/events.

responsible for their content and our individual speakers, who're solely responsible for their content and their opinions.

On behalf of our guests, Ant Phillips and James Blackburn, I'm Karen Bannon. Thanks for your time and have a great day.