CASE STUDY

Which ESB Option is the Right One For You?

Apples and oranges. NServiceBus and Mule ESB. In the same way the first two are both tree-growing fruit, the latter are both ESBs. Sure, one is a framework and the other is pure ESB, but you would shop for them both in the same aisle, so to speak. Boiled down, an Enterprise Service Bus (ESB) is a software architecture model that enables you to integrate applications in a service-oriented architecture (SOA) and end up with a single unified architecture. In the long run, ESB's are more maintainable and extensible than multiple point-to-point integrations.

If you take a high-level view of NServiceBus and Mule, you can see where their differences lie. NServiceBus, dubbed the most developer-friendly service bus for .NET, is a code-centric, API framework that is good for messaging between .NET applications. Whereas Mule ESB, one of the first open source ESBs to be successful, is a configuration-centric integration-platform-as-a-service (IPAAS) solution well suited for heterogeneous environments.

As you would expect, they both have a fair share of benefits and challenges. NServiceBus cannot necessarily be plugged directly into existing WCF services without changes. In addition, it requires service layers as either custom Windows services or NServiceBus.Host.exe services. As for Mule ESB, if you're looking for suite functionality, you may have to combine it with other third-party products, as it is truly a pure ESB.

To paint a clearer picture of their differences, advantages, and drawbacks, consider a couple example scenarios.

Integration with SalesForce

For connectivity with NServiceBus for SalesForce integration, you need a web application listener exposed through the firewall to receive SalesForce events. Then follow the Command-Event pattern to handle events in the Services tier. Additionally, you need an implementation of the Bayeux protocol to receive streaming events and a custom Windows Service for polling implementation to guarantee message reliability. Finally, you need to implement and maintain custom code to call SalesForce APIs.

For the same scenario, Mule ESB connectivity requires you to add an endpoint for streaming and add a polling endpoint for reliability. The Mule ESB SalesForce connector provides extremely easy implementation without requiring code API access. What would take several thousand lines of C# code can be handled in a few hundred lines of Mule configuration code. As the SalesForce API evolves, Mule handles updating the connector code to match the latest API version.

Integration with COTS Database

When facing COTS database integration, you might not be able to have the COTS application send an NServiceBus message. You would need a scheduled polling solution, either an NServiceBus scheduler or a custom Windows Service, and a mechanism to track progress, like

a watermark. In addition, you would need to implement database access code and create and consume events and translate to APIs in various systems.

In this case, Mule ESB offers built-in polling and watermarking, built-in JDBC access/infrastructure, and the DataMapper, which can map data between message payloads and APIs.

Tools Comparison

Lining up each platform's tools side by side helps illuminate the comparison.

NServiceBus	MuleESB
Code Centric	Configuration Centric
C#.NET	Java/Invoke .NET Business Logic
Visual Studio	AnyPoint Studio (Eclipse)
MSMQ (other pluggable transports)	MSMQ, JMS/AMQP (ActiveMQ/RabbitMQ, etc.)
MS Build	Maven
Unity/Castle Windsor/Spring.NET/StructureMap/Autofac/Ninject/other	Spring (Beans)
NServiceBus.Host.Exe or custom service	Mule ESB/Mule Management Console
Windows Clustering	Active-Active Shared Memory Clustering

Lessons Learned

In working with both solutions, we havedeveloped the following takeaways:

NServiceBus is good solution for integrating custom .NET applications where you control the code. If you have to integrate with cloud or COTS solutions for which you don't control the code, be prepared to face the challenge of a lot of custom code to get messages onto the ESB.

Mule ESB is a good solution if you have a heterogeneous environment or know you will need to integrate with cloud or COTS applications in the future. A working Mule application can be developed through AnyPoint Studio with no Java code, Spring, Maven, or Groovy. In addition, you can minimize Java dependencies by engaging specific resources or outside help with basic setup/architecture, then leverage that architecture across configuration-based flows.