Architecture Guidance for Microsoft’s Integration & Service Technology Portfolio

by Gijs in ’t Veld, CTO, motion10

“By 2015, more than 25% of enterprises will directly use cloud application infrastructure services, up from less than 5% in 2011 (more than 50% will use PaaS technologies indirectly, embedded in SaaS, brokerage offerings and other solutions)” – Gartner Research

Integration & Service Technology Architecture: What & Why?

Integration is key in a best-of-breed application landscape, B2B (business-to-business) scenarios or service oriented architecture (SOA). Applications, functions and services need to exchange data in order to participate in business processes. In the early days, integration was done by creating peer-to-peer connections between applications using direct database interaction, exchanging proprietary import/export files or executing API (application programming interface) calls. Over time, this resulted in unmanageable and non-performing applications due to the well-known “spaghetti integration” dilemma. With the arrival of the Enterprise Application Integration (EAI) server, integration got a more sophisticated approach, but very often it still resulted in peer-to-peer connection spaghetti, but now with a broker (hub) in the middle. Especially with the acceptance of SOA, the ESB integration pattern was born: loosely coupled integration through an Enterprise Service Bus, preferably based on the principles of service design (by Thomas Erl), thereby creating a manageable and flexible abstraction layer. But... the risk of creating unmanageable and non-performing integration architectures is still there; it is said that the greatest competitor of integration middleware is the developer!

Traditionally, integration middleware (the common name for the group of products facilitating ESB, EAI, B2B) is hosted and operated by the company that also hosts the back-end applications that are used to support the company business processes. For some scenarios, mainly B2B, external providers are sometimes used to integrate the 3rd parties outside the company firewalls. This is where you currently see traditional VAN (Value Added Network) providers morph into cloud based B2B integration service providers.

Depending on the integration scenarios, patterns have evolved over the years to become standard integration patterns. A great resource that still applies is Enterprise Integration Patterns (Hohpe & Woolf, 2003), containing all possible integration patterns that are integration middleware agnostic.

The paradigm shift to cloud computing, brought by Infrastructure-as-a-Service (IaaS), Platform-as-a-Service (PaaS) and Software-as-a-Service (SaaS) will face companies with new challenges, such as:

- How do I integrate my back-end systems and services with multi-tenant SaaS applications?
- How do I create composite services that are orchestrated services provided by multiple SaaS applications or even SaaS vendors?
- Should I use integration middleware in the cloud (PaaS) to integrate my on-premise applications?
- What does hybrid integration architecture look like and how to I manage such an environment?
- How do I govern hybrid architectures?
With the move to mass-market, multi-tenant SaaS solutions, integration will become even more important than it is today, because now companies can start to deliver their value add to customers not only by building tailormade solutions anymore, but by leveraging smart combinations of cloud based offerings.

This paper will provide high level guidance on when to choose what Microsoft integration solution, how to combine them and what integration architecture should be applied to implement the new scenarios around PaaS and SaaS so you can create the appropriate future proof architecture today.

Figure 1 – Hub and spoke integration

Figure 2 – Spaghetti integration
On Premise Integration Middleware (BizTalk, SQL Server, Service Bus for Windows Server)

Integration software that is hosted on a company’s own servers or servers managed by a dedicated hosting provider, is called on-premise integration middleware. Microsoft mainly provides three products in that category, each used for different integration needs: BizTalk Server, SQL Server and Service Bus for Windows Server. Each is described in the following paragraphs.

BizTalk Server

Microsoft BizTalk Server is a very mature integration middleware product that has been around since 2000. Today, with the 2013 release, it provides a robust platform to implement an EAI or B2B integration layer and it is also used very often as an ESB at the core of service oriented architectures. Furthermore it contains full functionality to connect to Windows Azure to provide secure hybrid cloud architectures. With the 2013 release, full REST support has been added as well. The product contains facilities for transformation, routing, business process orchestration, Business Activity Monitoring (BAM) and business rules; all the ingredients to create a scalable integration layer are there.

More than 12,000 enterprises worldwide use the product today, among which there are very large implementations handling several million transactions a day. BizTalk is also used quite often as the middleware layer between SharePoint front-ends and multiple back-end systems and functions to provide service composition, transactions, scalability and monitor ability in Enterprise Portal scenarios. Although there have been rumors saying otherwise, it is important to note that BizTalk Server has a future roadmap, the next (R2) release will become available in 2014. It will for example provide JSON support.
SQL Server

SQL Server, more specifically SQL Server Integration Services (SSIS) is often used for data integration services, such as ETL (Extract, Transform and Load). For batch-oriented data level integration this is perfectly suited.

Apart from providing data integration services, SQL Server is also a very important piece of BizTalk Server and SharePoint:

- For BizTalk it is the main persistence engine for publish & subscribe, guaranteed delivery and long running business processes paradigms.
- For SharePoint it is the main persistence engine for storing documents and workflow states.

By means of SQL Server Analysis Services (SSAS) and SQL Server Reporting Services (SSRS) in combination with the BAM facilities in BizTalk and tooling such as Excel, PowerPivot and Visio Services surfaced in SharePoint it provides a very powerful operational BI stack that can help improve business processes. The latest version of SQL Server is 2012 and 2014 will be released shortly.

Service Bus for Windows Server

For light-weight on-premise integration it is possible to use Service Bus for Windows Server together with WF (Windows Workflow) and WCF (Windows Communication Foundation), which are basically all standard components in the .Net framework. Of course this is not on-premise integration middleware architecture as sophisticated in terms of integration capabilities, monitor ability and scalability like BizTalk Server, but for scenarios where there are no requirements for such extended capabilities and Quality of Service attributes (i.e. non-functional requirements), it suffices. With the introduction of Service Bus for Windows Server, Microsoft has already started the trend of embedding integration middleware in the Windows platform.

It is important to note that all of the BizTalk Server, SQL Server and Server Bus for Windows Server (including WCF and WF) integration solutions are built using Visual Studio based tooling, a familiar environment for developers in the Microsoft world.

By introducing Server Bus for Windows Server, Microsoft has already made a start with providing integration services as part of the operating system, both on-premises and in the cloud; Service Bus provides the same functionalities no matter where it is hosted.

Cloud Integration Middleware (Windows Azure Service Bus and Windows Azure BizTalk Services)

“With Service Bus, Microsoft has made significant strides towards delivering the ESB pattern at Internet scope.” – Microsoft 2011

“An Internet- scoped ESB should make it possible to integrate your on-premises ESB with your services running in the cloud.” – Microsoft 2011

Integration software that is provided as a (multi-tenant) service in the cloud, or running in a single-tenant fashion dedicated to one company is called cloud integration middleware. Microsoft mainly provides two main solutions for integration in the cloud: Windows Azure Service Bus and Windows Azure BizTalk Services, or your own virtual machines (most probably with BizTalk Server installed) running in the cloud on the IaaS platform (Infrastructure-as-a-Service). This chapter will focus on Windows Azure Service Bus and Windows Azure BizTalk Services only.
Windows Azure Service Bus (SB)

With the arrival of Microsoft’s cloud platform, Windows Azure, it has also become possible to build and deploy cloud integration solutions. Windows Azure is a PaaS offering and according to Gartner, “PaaS is the architectural center of the cloud”. Windows Azure is mainly an application platform in the cloud that can be used to host tailor-made applications at cloud scale. This means that companies using this technology will build their own solutions, to be deployed and run in the cloud. The company is not only responsible for architecture, design and development of the applications, but also for functional and technical maintenance and health.

Providing a platform in the cloud also means that it needs to incorporate integration capabilities. Enter Azure Service Bus. This Integration-as-a-Service offering (or, “Cloud service integration and composition” as Gartner calls it) has become an integral part of Windows Azure, making it possible to host and manage integration services. Currently it is possible to deploy relatively simple integrations only, using WCF based communication endpoints, messaging (queuing) and routing mechanisms. Service Bus can therefore be used as an abstraction layer between (cloud) applications and services, providing REST, SOAP and WS-* connection capabilities.

Windows Azure BizTalk Services (WABS)

Since the GA release in November 2013 of WABS it is also possible to leverage functionalities for processing B2B specific scenarios such as EDI (Electronic Data Interchange). Features include: Transform (for creating transformations between XML instances), TPM (Trading Partner Management) and bridges for specific communication and exchange protocols such as AS2 (secure exchange of B2B transactions over HTTP), FTP and X12 (EDI-FACT support will follow in a later release in 2014).

This clearly means that within a couple of releases from now (and the release cycles in the cloud realm are much shorter than for on-premise software; typically 3-6 months), Azure will provide sufficient integration capabilities (including workflow for process orchestration & composition and BPMN and BAM & BRE support) to cater for production-ready, full-featured integration solutions for: cloud-cloud integration and cloud-on-premise integration. And… all of this as part of the platform, as it should be.

On a last note: WABS can already integrate with the on-premises LOB adapter pack to provide hybrid integration with the most commonly used LOB systems like SAP and Oracle.

Hybrid Integration Solutions (Mixing On-premise and Cloud Architecture)

Many companies will keep on hosting parts of the application infrastructure in-house (or with a dedicated hosting provider) and some parts of it in the cloud. Cloud applications (multi-tenant SaaS or homegrown and deployed, single-tenant in PaaS) will be used for their (elastic) scalability, time-to-market and ease of management features. And let’s not forget the very important advantage of having Opex (operational expense) instead of Capex (capital expense) because you don’t own the software but pay-per-use. The on-premise applications will either be legacy applications that cannot be ported to the cloud because of their (lack of) architecture, or applications that have very high security or privacy requirements. This means that integration between on-premise and in-the-cloud applications will most of the times be necessary. Enter Hybrid integration solutions.

Integration of on-premise applications and services with cloud applications and services can leverage the facilities provided by BizTalk Server or Service Bus for Windows Server handling the on-premise integration requirements and Windows Azure Service Bus and Windows Azure BizTalk Services handling the cloud
integrations. A bridge between the two domains is provided by Windows Azure, using the relay services in the cloud and ACS (Access Control Service) for the security between the two worlds. By using Azure Virtual Network from within the on-premise integration layer, the on-premise layer creates a secure “gateway” with Azure Service Bus without undermining the DMZ (demilitarized zone) or other security rules.

**Federated ESB**

Probably the most important integration pattern for hybrid integration solutions is the Federated ESB pattern, where two (or more) ESBs are deployed: one on-premise and one in the cloud, thereby serving the two domains with the same approach. These ESBs are connected by either implementing a brokered ESB architecture (where one “master” ESB determines where to route requests, thereby offering bridge services) or a connected ESB architecture (each ESB having its own service registry, visible to all ESBs, thereby reflecting each service’s affinity for one of the ESBs). Either way, each ESB is responsible for connectivity in its own domain and routes service requests to other ESBs if these services are not hosted in its own domain.

IBM has said the following about these patterns: “Service specification and implementation are governed at the domain level, with a degree of centralized governance to enable cross-domain service reuse and interoperability.” (Source: IBM).

“The widely anticipated hybrid cloud pattern will likely be implemented using integration technology, rather than reliance on homogeneity across private and cloud data centers.” – Gartner 2011

![Figure 4 – Federated ESB](image-url)
What Will the Future of Microsoft's Integration & Service Technology Look Like?

Microsoft is dedicated to merging the integration and service technology features provided by BizTalk Server, Service Bus for Windows Server, WCF, WF, Windows Azure Service Bus and Windows Azure BizTalk Services into an architecture that has the following key capabilities:

- One workflow engine (instead of two: X/LANG in BizTalk and WF in .Net)
- One toolset for development, deployment and management of integration solutions
- One hosting “container”, available in public cloud, private cloud and on-premise
- Providing for low latency and high throughput / volume solutions
- Alignment of Service Bus and ESB across platforms
- Provide BAM and Business Rule Engine (BRE) across platforms

This will be a multi-year development effort, and in the meantime the cloud based integration middleware will further evolve to incorporate new features and functionalities that are most often based on technology or capabilities already available in the BizTalk Server or Service Bus. These features will be made available as re-usable .Net components that can be mixed-and-matched to create multi-tenant integration solutions for customers. The toolbox in the Visual Studio development environment will be extended with integration specific capabilities, such as: Transform, Route, Messaging, Identity & Access and Cache, but also features for debugging and testing. The run-time container that will host the integration solutions built with these components will provide built-in features for multi-tenancy, scalability, resilience and management & monitoring. It will not matter if the solutions will be deployed in the public cloud, a private cloud or on-premise; the tools are the same.

By doing this, Microsoft will provide an attractive integration and service technology platform that appeals to integration architects, specialists and the .Net developer community.

Future Proof Integration Architecture Guidance

Gartner has said the following related to this topic: “Users that delay the adoption of PaaS to sometime in the future, when the standards, leading providers and best practices are better established, should invest now in building expertise in service-oriented architecture (SOA), including its event-driven form. SOA is a bridge from the traditional computing in the enterprise data center to the hybrid model of computing, engaging enterprise data center and cloud resources.” (Source: Gartner). Gartner has made an excellent point, and here is why: SOA is the architecture paradigm that makes integration of disparate applications, functions and services possible by adhering to the following design principles (also see www.soapprinciples.com):

- **Reusability** – Services contain and express agnostic logic and can be positioned as reusable enterprise resources (think about the future uses and make them generic)
- **Autonomy** – Services exercise a high level of control over their underlying runtime execution environment (there are no other services making use of the underlying components)
- **Discoverability** – Services are supplemented with communicative meta data by which they can be effectively discovered and interpreted (if nobody knows about the existence of services, they will never be re-used)
■ **Composability** – Services are effective composition participants regardless of the size and complexity of the composition (break-up larger components into smaller, more manageable and composed ones)

■ **Loose coupled** – Service contracts impose low consumer coupling requirements and are themselves decoupled from their surrounding environment (contract-first development of services; don’t generate contracts)

■ **Abstractness** – Service contracts only contain essential information and information about services is limited to what is published in service contracts (the contract does not contain system level information on underlying systems)

■ **Standardized Service Contract** – Services within the same service inventory are in compliance with the same contract design standards (by using a uniform language and applying standards notations for creating contracts, discoverability, reliability and application of governance of services will improve)

■ **Statelessness** – Services minimize resource consumption by deferring the management of state information when necessary (keeping excessive state can undermine the availability and scalability of services)

When designing and developing new software, make sure you do it in a service oriented way and comply with the above principles whenever feasible. This also applies to modern REST-based integration. This will help build solutions that are highly usable, composable and portable.

Integration middleware can also be used to wrap functions in legacy applications in such a way, that they can adhere (as good as possible) to the above principles and thereby become candidates to publish their “services” on the on-premise enterprise service bus for consumption by composition services or other consumers, on-premise or through a federated ESB in the cloud. For these purposes, REST and OData are important supported capabilities as well. This way, investments in legacy keep on delivering value.

Furthermore, by applying standard integration patterns as described in Enterprise Integration Patterns (see Introduction), the chance that a manageable and, even more important, portable integration architecture will be created increases significantly. It is also vital to implement as little custom (coded) components as possible in order to make compatible solutions for future versions of integration middleware, such as BizTalk Server or Azure Service Bus and WABS.

Last but not least, Microsoft will make migration of artifacts between intermediate integration solution products and versions possible. Schemas used currently in BizTalk Server can most of the times be migrated to WABS without conversion. And for transformations, a migration tool for converting BizTalk transformation maps to the Azure based Transforms has been made available earlier last year. It can convert BizTalk maps almost always 100%, provided no custom code has been used.

The most important rule that applies though, remains: make sure your integration solutions indeed have an architecture based on standard patterns and best practices and little to no custom code; this will make the transition to a federated ESB much easier.

**Key Takeaways**

Integration architecture is a complex topic! With the arrival of the cloud computing era, companies today are faced with even more difficult architecture choices than ever before. Even in a “Microsoft unless” environment, there are still many decisions to make: how to integrate SaaS with on-premise applications, should I use cloud integration middleware to provide connectivity between my on-premise applications and functions, should I still invest in creating on-premise integration solutions, is service oriented architecture the right path to follow, will
an enterprise service bus help me address all requirements?

This article has given you (CIO, IT Manager or Architect) the most up-to-date facts on the availability and roadmap of Microsoft integration products and it has provided you with high level guidance on how to design future proof integration solutions with Microsoft technology. Hopefully it has helped to organize your thoughts on integration architecture in today’s “confusing” application landscapes and will prove useful now or in the

Gijs in ‘t Veld

Gijs in ‘t Veld is CTO at motion10, where he is specialized in Enterprise Integration and Portals. Since 2001 he has been involved with hundreds of integration projects world-wide on the Microsoft Business Platform. Gijs has been awarded Microsoft MVP 7 times in a row for his many contributions to the world-wide adoption of BizTalk Server and Windows Azure in the past decade and has been a Microsoft V-TSP (technical sales professional) Integration since 2007. He is also a technology advisor for the Microsoft Business Platform technologies Windows Azure AppFabric and BizTalk Server.

Contributions

■ Architecture Guidance for Microsoft’s Integration & Service Technology Portfolio
■ Service Driven SharePoint Enterprise Portal Design
■ Guidance for Integration Architecture on the Microsoft Business Platform