OpenLegacy
The Business End of Integration

Sales Battle Card

Integration Market Trends

Current market Trends

Currently the large enterprise integration market is dominated by SOA based ESBs. These technologies that has been around for over 15 years are still the most common way for large enterprises to approach their integration needs and are a considerable pain point when trying to embark on a digital transformation, but while these technologies are mostly recognized as outdated, still almost no major enterprise with substantial IT needs, have moved beyond them. There are several reasons for that:

Cost of change - ESBs have grown to become huge and complex platforms, not easily migrated

Lack of legacy integration - While some newer vendors claim to have legacy support, mostly it is very limited and depends on 3rd party connectors. For example JitterBit's and Mulesoft's offering for Mainframes (still today running 96 of the world's top 100 banks, 23 of the 25 top US retailers, and 9 out of 10 of the world's largest insurance companies) is limited to an underwhelming DB2 JDBC connector.

Cloud - A lot of the newer vendors are 'Born on the cloud' type companies. While this might be appealing in some cases, for larger enterprises in heavily regulated industries this can actually be a deterrent. These larger organizations would certainly like to have a path forward towards to cloud but they are not, by and large, planning on moving any of their core applications to the cloud any time soon. For these enterprises a hybrid & cloud agnostic approach is the most appealing one as it allows them to move at their own pace.

Nothing Changes - Last but not least, the drive to move from ESBs to other solutions such as IPaaS isn't very strong because at the end of the day most IPaaS solutions are just ESBs in the cloud. They are still a message-driven, asynchronous, flow-based, services-heavy, integration-team-requiring middleware for implementing SOA. If the only benefit you get from a migration is a nicer user interface and 500 connectors to facebook, twitter etc. than you rather stay with what IBM, SoftwareAG or Tibco has to offer and endure the costs,inflexibility and prolonged times to market.

In summary, currently the market is searching for solutions to address new needs and requirements while still integrating with existing platforms and processes. Currently this leads to an explosion of multiple integration platforms serving different segments of the organization and a reliance on services-heavy solutions. This is considered a problem for large enterprises who wish to consolidate and simplify their integration architecture (See key findings in https://www.gartner.com/doc/3174717/predicts--opportunities-integration-digital).

The OpenLegacy Advantage

OpenLegacy offers a completely new approach to enterprise integration, one which addresses all the needs of the modern enterprise trying to embark on a digital transformation. The OpenLegacy approach is a departure from the old Service Oriented Architecture (SOA) of the past, a fact which allows for implementations once considered impossible or impractical.

At a high level the OpenLegacy native APIs approach do away with the concept of a large anypoint to anypoint middleware such as ESBs, it does not accept that the integration channel needs to be agnostic of it's context and it certainly does not accept the need for flow process doing at runtime what could have been done at design time.

A way to think of the OpenLegacy approach is that instead of having a single integration middleware, OpenLegacy is a factory for multiple micro-integration-applications. each micro-application is a Java compiled code, in charge of a single API (or an API group), and is independently deployable and runnable.

Some differences between the traditional approach and OpenLegacy approach are:

- **Automatic API Generation** - With OpenLegacy's approach the APIs are created automatically by parsing backend assets. This is a huge advantage as it shortens the time needed to create a service dramatically.
- **Message based vs Model based** - OpenLegacy model-based approach does not require schemas or mappings. Instead of trying to map each message at run-time, a time-consuming and security-exposed process, OpenLegacy creates a Java based model for each API at design time. This means that the format of the message is integrated with the execution of the flow, providing for maximum performance and security
- **State** - With OpenLegacy it is very easy and natural to keep state vs the different backend systems, making a huge challenge in traditional integration projects simply disappear
- **Flexibility** - Dealing with core applications sometimes means working in non-standard proprietary ways. While traditional middlewares may have a wide range of customizations available, they are still limited by their design. OpenLegacy's approach allows for complete flexibility, since each API is an independent application which can be basically be treated as a Java application, while still being a first-class citizen on the platform.
- **Channel Specific Logic** - Another advantage of the OpenLegacy approach is the ability to introduce channel specific logic right at the API level without the need for additional application platforms.
- **Performance** - Since each API is a compiled java code automatically generated with caching abilities, performance is unparalleled by other solutions.
- **Deployment Agnostic** - OpenLegacy's APIs can be deployed anywhere with ease,. Be it a cloud, on-prem or any other future technology - if it runs Java it will run the OpenLegacy APIs. You do not even have to use the OpenLegacy server to run them (although it does give some additional management functionality).

Some examples of the things OpenLegacy is able to do which are not possible on other solutions using the SOA approach:

Dealing with multiple/unpredictable messages - One of our major customers (one of the largest banks globally) have an extremely proprietary system. The problem is not so much with the platform or the technology but rather with the core-application itself. It was written in a way which produces a very large variety of unpredictable messages. Vendors such as IBM, Tibco and Accenture have tried to solve this problem with existing tools but this is an extremely difficult problem in message based SOA products. With OpenLegacy it was a very easy problem to solve using concepts of dependencies and casting on the Java model

Distributed Transactions - We are in an engagement with a popular core banking application provider (>100 banks including Wells Fargo), they use us as an API integration platform but have a specific need to provide transactions spanning multiple platforms so that changes done on the mainframe will be rolled-back automatically the transaction fails to update a cloud based database for example. This is extremely hard to do on a SOA based approach since each service is indepent. New implementation of transactional web-services do exist but requires major changes to the backend applications. With OpenLegacy's stateful mainframe adapter it was very easy to offer a transactional solution spanning legacy and cloud.

Channel Logic - An insurance customer of ours, after creating a mobile application based on OpenLegacy's APIs, wanted to add the ability to attach pictures to a claim, so that customers reporting an accident, could just use the mobile device camera to take pictures of the damage and upload it. Of course their backend application does not yet support this functionality and would they be using a different integration platform they would have needed to implement it on their backend or create a new platform just for that use-case. With OpenLegacy they implemented this channel specific logic right on the platform using standard Java and deployed the solution in days.

Future Market Opportunities

As the market moves more and more towards digital services and cloud solutions, we believe that the need for a new, consolidated integration approach and architecture will grow. The problems associated with SOA integration: long TTM, complex and service-intensive projects with high TCO and high risk of failure, will drive customers to search for better solutions. We believe that native-API solutions such as OpenLegacy will be the new standard architecture for enterprise integration for legacy and non-legacy customers alike. As we expand our offering and product, we believe that we are very well positioned in a nexus of convergence where data and application integration will meet API management and cloud transformation. These shifts will create major opportunities and we plan on pursuing them with vigor.