White Paper

Appian and the Enterprise Service Bus

How an Appian-centric system can benefit from an ESB

By Mike West

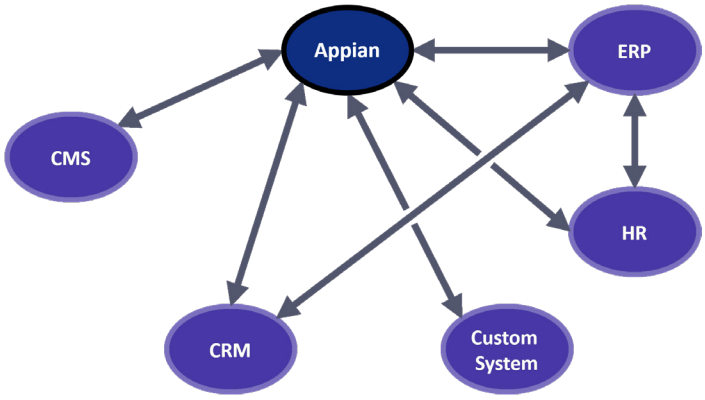
MACEDON TECHNOLOGIES

Table of Contents

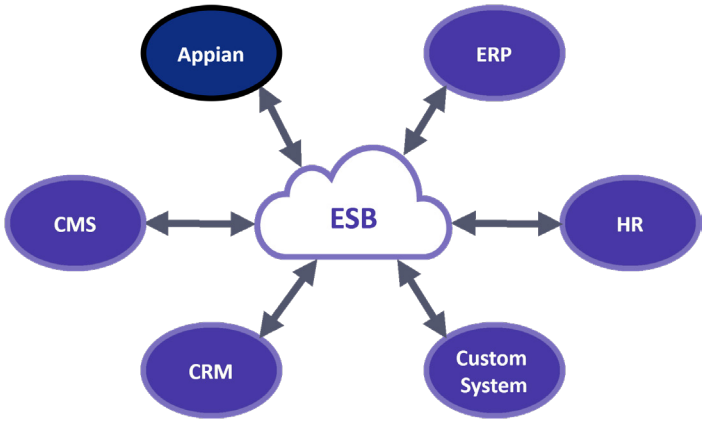
Problem Statement

Modern IT environments often have a diverse mix of custom and commercial off the shelf (COTS) software platforms to serve the business' needs.

In such an environment, each system needs to be able to easily share information with other systems in the environment. As the number of systems increases, this becomes a larger and larger problem, and leads to a very rigid configuration. If any system is updated or replaced, each of the others may require updates to continue successfully integrating.



Instead of this type of environment, we would like a more loosely coupled infrastructure in which each system must talk directly with a single intermediary system.



In such an environment, the ESB is the only system that needs to know the specific details of how to talk to any given node.

In addition, the ESB can offer guaranteed delivery, so that any client can send it a message and rely on the ESB to handle network or service reliability issues.

Preferred and Supported Integration Patterns

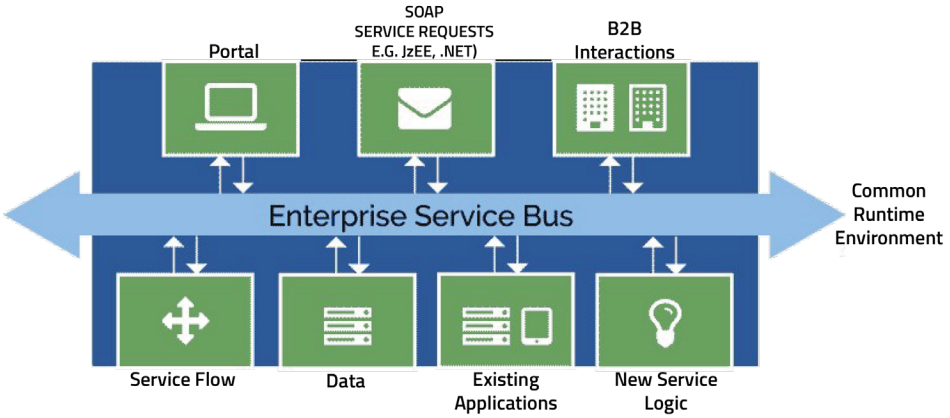
Appian supports different styles of integration out of the box. For a modern Appian system, the preferred mechanism of integration (for both send and receiving data) is to use REST-based APIs passing JSON. If the business logic

of the API is simple, setting up an endpoint in Appian can often be done in a few minutes. Likewise, establishing a REST connection to an external system and parsing the response payload can also be done quite easily. These are powered by Appian's fromJson and toJson functions that convert between JSON and Appian's native dictionary type.

In addition to REST APIs, Appian also supports SOAP over HTTP, both as a service and a client. This is supported natively, without needing to write custom low-level code. For other, more specialized protocols, Appian can essentially support anything that can be written in Java, using the Integrations SDK.

Services and service calls from Appian support a variety of mechanisms for authentication. When Appian is the client, it supports natively Basic (username/password) authentication, API Keys, AWS Signature Version 4, Google Service Accounts, as well as Oauth 2.0 (authorization code grant and client credentials grant). As a service, basic authentication and API keys are supported.

Given such a broad set of functionalities, together with Appian's traditional strength in Business Process Management, it might be tempting to use Appian as an ESB. Why not put Appian at the center of Diagram 2? There are a few areas in which a traditional ESB, such as Mule, Tibco, Boomi, or IBM's Websphere MQ offer a better solution.



ESB diagram from AINS [1]

Guaranteed Delivery

A production-quality Enterprise Service Bus will support guaranteed delivery of payloads between systems. This means the client system can 'fire-and-forget' data to the service system. The call to the integration exposed by the ESB completes immediately, and the ESB handles the delivery to the service system. The caller no longer needs to handle issues with uptime or network connectivity to the service. This is typically implemented by a retry mechanism for failed integrations. The ESB will, when unable to connect to the endpoint, call the service again after a few seconds, then after a few minutes,

then after a few hours, etc. This could in principle be reproduced by Appian, but it is not available out of the box, and would have to be implemented in a highly configurable way for the (presumably) large number of integrations.

Throughput

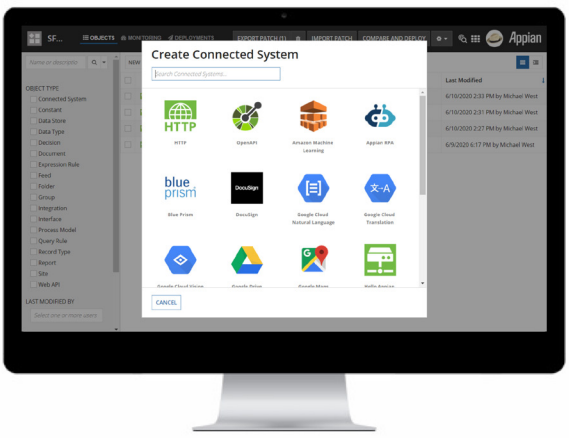
ESBs are designed for high throughput and scalability to support a high volume of data passing through the system by a large number of client systems. Appian, on the other hand, is designed for rapid application development, cross platform web and mobile interfaces, and record centric design. The need for responsive user interfaces and business processes drives an underlying system architecture that relies heavily on in-memory data caching. This caching is in tension with high data throughput. In particular, Appian would not be ideal for any ETL-type service.

Some Exceptions

In some cases, even when using an ESB, it may be the case that a direct connection from Appian to a service has advantages over going through the ESB.

One example is for services that use the Authorization Code Grant authentication mechanism. This is a protocol that allows for services that act 'as an end user', so to speak, as opposed to as a service account. When using an ACG integration, a user is presented with a link to the service system to explicitly allow the client system (Appian, in this case) to act on their behalf. This is accomplished by the service passing a token associated with the user back to Appian. This token is then cached internally by Appian in a secured way so that even a system administrator could not access another user's token. Because the user interacts with the service system, it would be difficult to achieve this over an ESB.

Another situation in which a direct connection may be more suitable is for a few specific services for which Appian provides 'connected systems'. A connected system is an Appian object that encapsulates logic for authentication as well as constructing service calls and parsing their results. They turn cod-



ing into configuration for integrations. These connected systems exist for DocuSign, SharePoint, and Salesforce. Since these are web-hosted services, these connected systems are designed to directly connect to the service.

Conclusion

On one hand, new resources with relevant expertise must be brought on to configure and develop the ESB platform. Depending on the ESB chosen, there can be license costs as well. From a project management perspective, having Appian talk to a given single system can involve coordination of three teams (Appian, ESB, and other system), instead of two. For these reasons, if the number of systems that must share information is not very large, an ESB may not be a good fit.

In a highly integrated environment with a large number of interacting systems, an ESB creates a more stable infrastructure, and allows solving communication problems in one spot, instead of for each pair of systems that must share information.

The principal benefits include:

1. The ESB must know how to talk to each system, rather than each system knowing how to talk to each other.
2. The ESB can offer guaranteed delivery of messages on a high volume of messages.

About the Author

Michael West is a Macedon Appian Enterprise Architect. He has been working on Appian projects since 2013. He currently lives in Austin, TX.

[1] "AINS Professional IT Services: Enterprise Service Bus." AINS
http://www.ains.com/enterprise-service-bus-esb/