Making Sense of The API Economy

By Joseph Gulla, Ph.D.
joseph.g.gulla@gmail.com

LINKEDIN profile at https://www.linkedin.com/in/joseph-gulla-ph-d-ba5b9515/

Abstract

In this paper, the author presents a 360-degree view of modern Application Programming Interfaces (APIs) that focus innovation and entrepreneurship often with a concentration on enterprise computing. The paper gives an introduction to the API economy then explains the classification of APIs with a particular focus on microservices. Leading API management software companies are identified and important API technologies are explored including REST and SOAP, as well as open source tools and open standards and protocols.

Introduction to the API Economy

According to Gartner (Welcome to the API Economy, 2016), the API economy is an enabler for turning a business or organization into a platform. How is this possible? According to Gartner, the API economy is a set of business models and channels based on secure access of functionality and exchange of data. So, to answer the question--new models and channels make this possible.

In new ways, APIs make it simpler to integrate and connect people, places, systems, applications and data. Many businesses are implementing API management products to make what they already have, often called systems of record, more readily available to new systems of engagement including mobile devices and cloud services.

Classification of APIs by Type

The author has devised a simple method of classification. O-APIs refers to the "older" types of APIs like those for access methods and performance management. Two examples or types of the older kind of APIs are summarized in Table 1.

Table 1. O-APIs

Category	Description	Example
A-type	Access Methods	The Queued Sequential Access Method (QSAM) was released in OS/360 offering device independence: to the extent possible, the same API calls are used for different devices (Queued Sequential, 2017). The Virtual Storage Access Method (VSAM) is in the same class as QSAM.

P-type	Application Performance	The Application Response Measurement (ARM) API is focused on application response measurement and has the function "arm_init" that is used to define an application. It must be made before any other ARM API calls related to that application like arm_start and arm_end (Application Response, 2017).

The N-APIs classification pertains to "new" types of API. New can best be classified as those that arose since the advent of the worldwide web. Three examples or types of the newer kind of APIs are summarized in Table 2.

Table 2. N-APIs

Category	Description	Example
D-type	Data access	Local Government, Climate, Ecosystems and Agriculture are examples of datasets available on data.gov. The data is available in many forms like .xls and .tar for human viewing or program use.
$-type	Accept online and mobile payments	PayPal payments & Square online and in-person payments are example of financial APIs.
M-type	Application program in the form of a microservice	These programs are created from a variety of API Management tools producing a new kind of application running on top of the legacy applications that is non-disruptive.

Special Focus on the M-type API

This type of API has come about since the creation of cloud computing. M-type means microservices, which is a kind of small data program. These small data programs are grouped into applications. These applications are used by companies to generate revenue, lower costs, improve efficiency and respond to competitive pressures. These programs are created in an integrated development environment (IDE) and they provide a connection between the legacy world (called the systems of record) and the new world of engagement These microservices APIs are surrounded by support capabilities like a management console, security, analytics and logging, that is used for governance. The most complete API Management programs have a full life-cycle approach that is needed because these microservices APIs are applications that need to be handled as company assets.

APIs and Microservices
There are two views of microservices. One view characterizes it as an architectural style where the other view uses the term microservice to describe the data programs created with API Management software products.

An Architectural Style
Through adoption of the microservices architectural style, companies use many very small modules, communicating using lightweight protocols that combine to provide a service. They are part of an application like order processing, supporting a business area like add a new customer, for a specific scenario like verify customer address. The microservices can be and are often used in multiple scenarios.

A Kind of Program
Many API Management software suppliers are providing an IDE and other key components like a management console and analytics, in support of the creation of small data applications they call microservices. These data applications can be used stand-alone or combined into applications by making important data and processes available in new ways without disrupting the system of record.

The Role View of APIs
Figure 1 depicts the role of APIs, which is to leverage existing enterprise assets to unlock your business capabilities. This view aligns with the Gartner vision and has this main theme—unlock your business capabilities by leveraging your existing assets. This is a compelling idea for many companies.

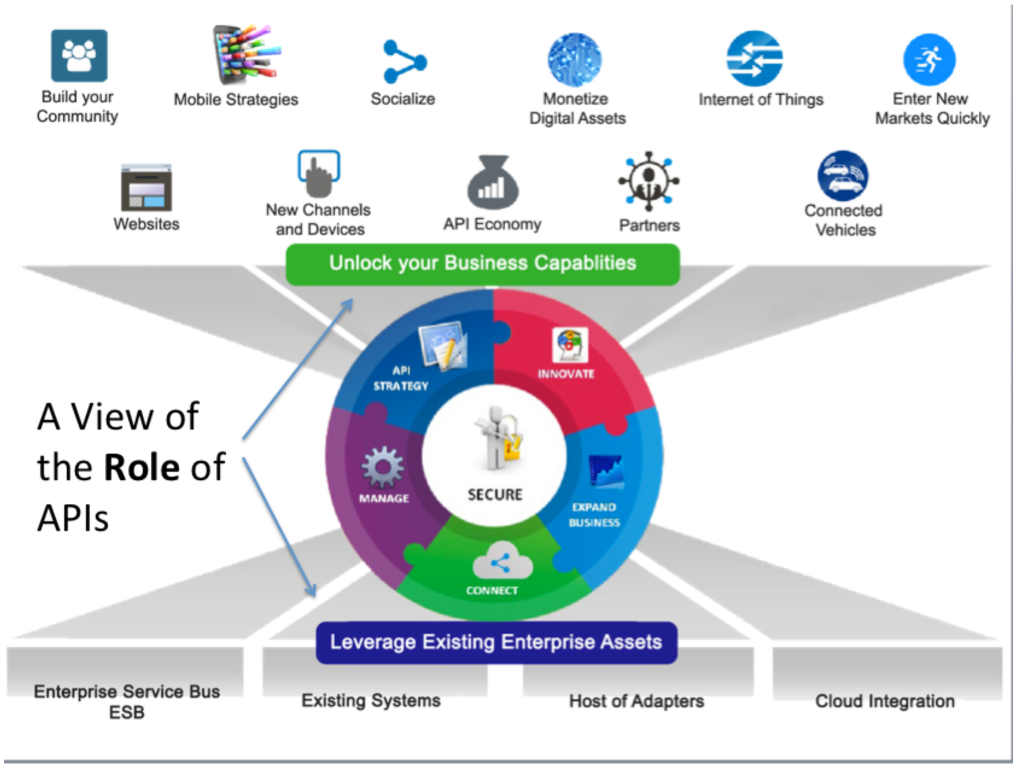
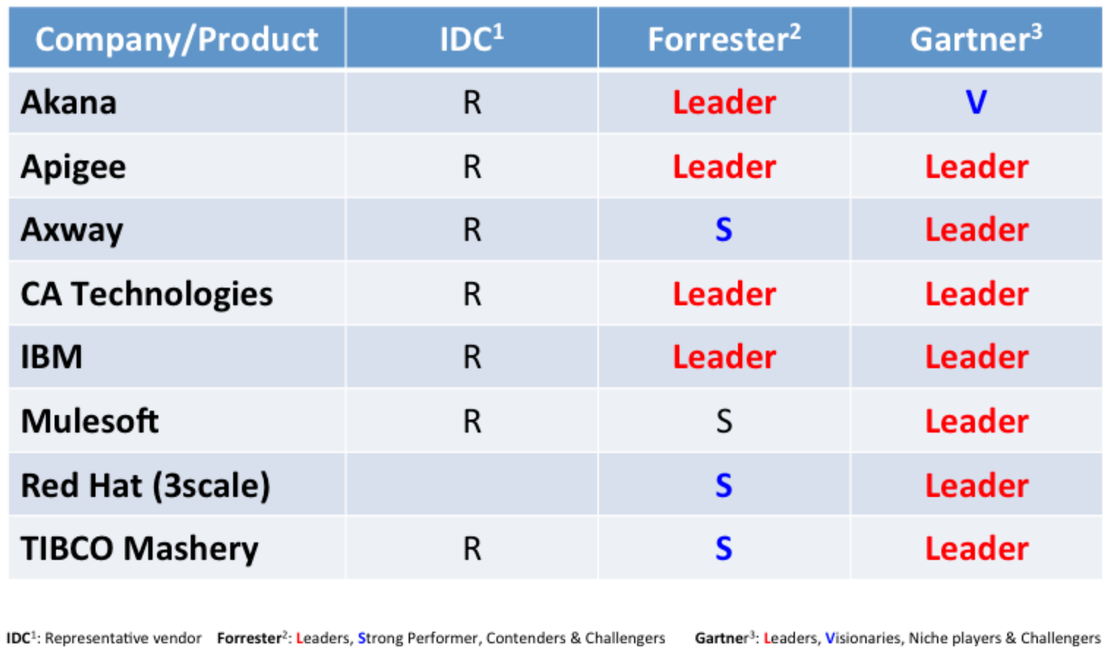


Figure 1. The Role of APIs (Fiorano API Management, n.d.)

Who are the Leading API Management Software Companies?

Figure 2, below, is a consolidation of the highly regarded API management products. Included are Leaders, Visionaries and Strong Performers identified by Forrester and Gartner. IDC uses the word Representational in its software taxonomy document.



Company/Product	IDC[1]	Forrester[2]	Gartner[3]
Akana	R	Leader	V
Apigee	R	Leader	Leader
Axway	R	S	Leader
CA Technologies	R	Leader	Leader
IBM	R	Leader	Leader
Mulesoft	R	S	Leader
Red Hat (3scale)		S	Leader
TIBCO Mashery	R	S	Leader

IDC[1]: Representative vendor **Forrester**[2]: Leaders, Strong Performer, Contenders & Challengers **Gartner**[3]: Leaders, Visionaries, Niche players & Challengers

Figure 2. Leading API Management Software Companies

How the API Management Systems are Organized to Get Work Done

The common components--IDE or developer portal, connectors, gateways, security, monetization, analytics and operations support tools—are common in the context of a build and run lifecycle. Figure 3, below from a recent BLOG post shows the features of many API products. Design Time includes Data Modeling, Interface Modeling and Registry & Repository whereas Run Time includes API Management Solution, Identity Stack, Monitoring, DevOps Tools, Logging, and Application Infrastructure (Broeckelmann, 2017).

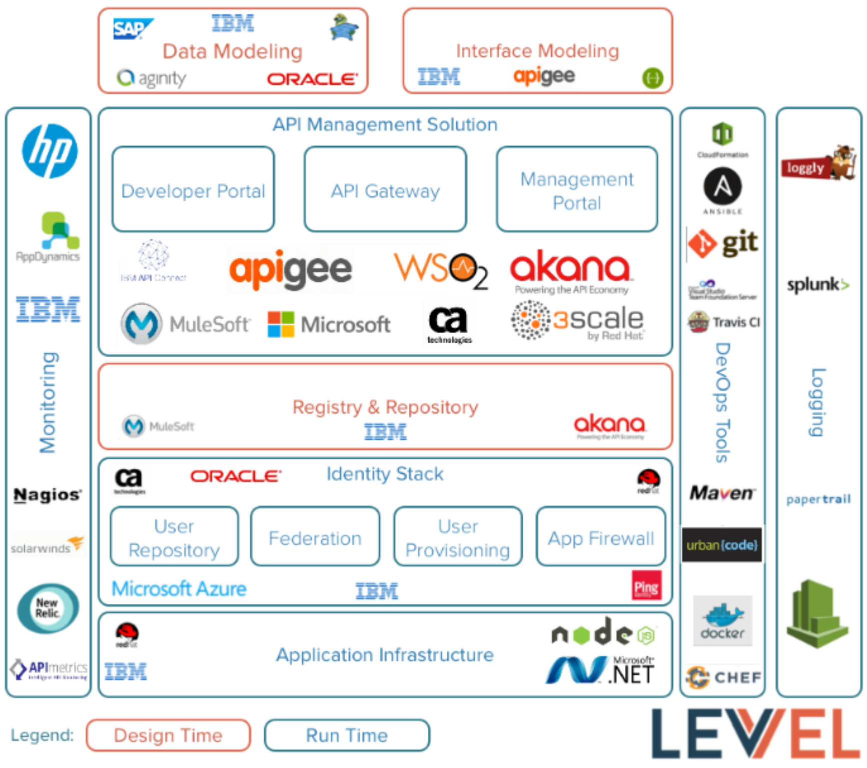


Figure 3. The Tools of API Management—The Full Stack

A Product Example -- Components of IBM API Connect

Rather than a logical view of API management software, lets look at a leading product's components. This is the actual list of components for IBM API Connect (API Connect Components, 2017). The list includes--

1. Cloud Management Console that provides an administrator interface mainly to manage/monitor (add, start, stop, delete, get usage report, etc.) servers and to register the components that will be used at runtime.
2. API Designer provides an API developer toolkit, installed on workstations to create and secure LoopBack applications. A LoopBack application is a Node.js application that supports the creation of REST APIs based on data model definitions.
3. Management Server is the API developer's interface to manage and secure their existing APIs.
4. Developer Portal is for application developers where published APIs will be referenced, described and can be tested.
5. API Gateway is used to enforce APIs security or functional policies like JSON schema validation and JWT generation. All registered applications send their requests to the API Gateway to access the APIs.
6. Liberty Collective are server clusters containing one Liberty controller and multiple Liberty members. Liberty collectives are only used if there is a LoopBack application or Microgateway to run. It constitutes the Node.js runtime environment. Liberty servers can be run on Linux on z.

Most API Management products have gateways although their function isn't always exactly the same. They also have servers to run and manage the APIs and facilities for design and development of APIs. They often interface with an Enterprise Service Bus, if it is present, allowing companies to make use of their existing integration and SOA investments.

Importance of RESTful Web Services and Other Internet Technologies

Many API products use Web services like REST and SOAP. Roy Fielding is the father of REST as it was the subject of his dissertation called Architectural Styles and the Design of Network-based Software Architectures (Fielding, 2000).

Here are a few key points from the abstract of his dissertation--"REST emphasizes scalability of component interactions, generality of interfaces, independent deployment of components, and intermediary components to reduce interaction latency, enforce security, and encapsulate legacy systems. I describe the software engineering principles guiding REST and the interaction constraints chosen to retain those principles, contrasting them to the constraints of other architectural styles. Finally, I describe the lessons learned from applying REST to the design of the Hypertext Transfer Protocol and Uniform Resource Identifier standards, and from their subsequent deployment in Web client and server software."

Figure 4, below, makes it straightforward to see some of the main points of REST. REST is a client/server architecture allowing for scalability. The server can interact with many clients at the same time because it is the client's responsibility to handle context. Not storing context on the server is one element of REST's architectural style.

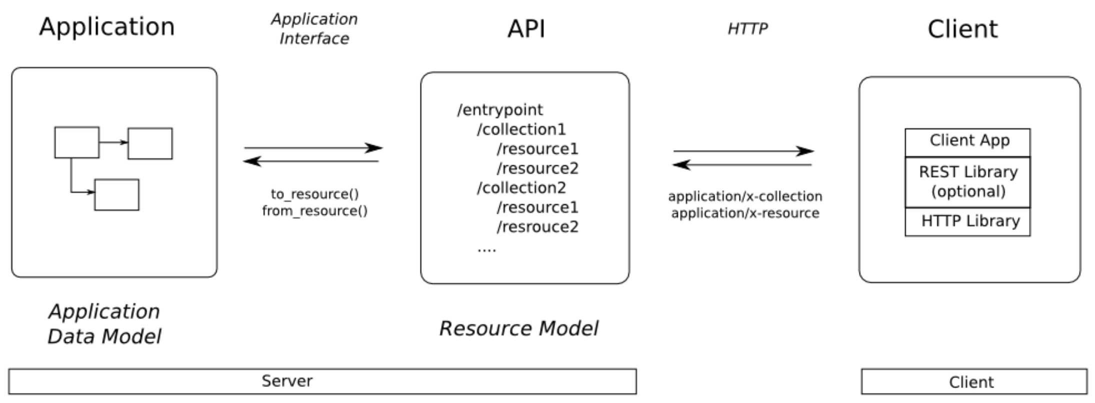


Figure 4. RESTful Architectural Style (Rocheleau, n.d.)

XML and JSON with REST

XML and JSON come up when discussing REST. They are simply ways of serializing data. XML is more flexible and with many standards designed around, some feel that it is too complex and long-winded. JSON is more straightforward and defines a few basic structures in simple ways. This makes it easy to use for informal data structures (JSON, REST, SOAP, WSDL, 2013).

SOAP versus REST

SOAP and REST are often compared but they are quite different. SOAP is a protocol whereas REST is an architectural style. REST APIs access a resource for data whereas SOAP APIs perform an operation. REST permits many different data formats like plain text, HTML, XML, and JSON whereas SOAP only uses XML. With SOAP and REST, security is handled differently. SOAP requires more bandwidth whereas REST requires fewer resources depending on the processing that is needed to support the API. SOAP has an advantage over REST in that SOAP has Atomicity, Consistency, Isolation and Durability, called ACID, which is a set of properties of database transactions. These characteristics help ensure accurate and recoverable database updates (SOAP versus REST, n.d.).

Role of Other Open Protocols, Standards and Tools

Some API management products are distributed as open source but many use open source components for logging, security, and other key support functions. The usefulness of open source protocols, standards and tools is as building blocks for these API products. The brief list below in Table 3, is taken from an internal document called OpenLegacy Terms and Definitions. It contains a sample of open protocols, standards and tools used by OpenLegacy API management.

Table 3. Open Protocols, Standards and Tools used by OpenLegacy

Item	Description
Apache Log4j	A reliable, fast and flexible logging framework written in Java.
Trail File	An XML representation of in/out screens.
oAuth	Open protocol to allow secure authorization for REST APIs, web, mobile and desktop applications.
EhCache	Widely used open source Java distributed cache engine.
Angular	Open source development platform for web applications.
Freemarker	Open source, Java-based template engine.
Templates	A structured format, created by Freemarker, into which data is entered when generating entities.

Special Focus on API with Enterprise Computing

In 2015, IBM announced z/OS Connect Enterprise Edition, a strategic API gateway into z/OS. This gateway is a configurable, high throughput interface into CICS, IMS, DB2 and WebSphere Application Server. This product is used to make APIs that utilize data from CICS and IMS applications while requiring no changes to the application's underlying COBOL or PL/1 code.

Since z/OS connect is a gateway, the most important thing it does is help transform applications written in one architectural style to a different one. It is very well documented so you can understand exactly what to do even if the tasks are not always native to your skillset. Figure 5 shows the inputs and outputs handled by z/OS Connect (IBM z/OS Connect, 2015).

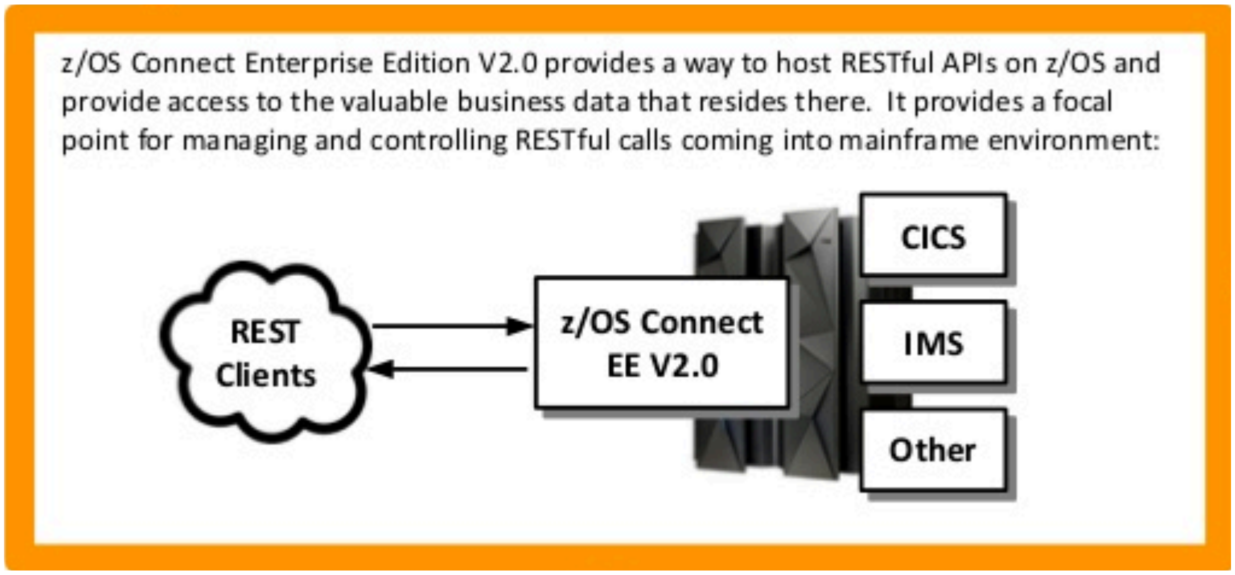

z/OS Connect Enterprise Edition V2.0 provides a way to host RESTful APIs on z/OS and provide access to the valuable business data that resides there. It provides a focal point for managing and controlling RESTful calls coming into mainframe environment:

Figure 5. The Role of z/OS Connect EE (Excellent 2 Pager, 2016)

Steps to Create an API for a CICS Transaction

There are four main task areas for creating an API using z/OS Connect (Creating an API, 2016):

1. Generate Bindings for the Application

Using a supplied JCL job stream, generate the bindings, data definitions like a commarea copybook, which will allow z/OS Connect to call your CICS application. This generates a Service Archive File (SAR) file.

2. Import the SAR File into the Tooling

This allows the z/OS Connect tooling to display the fields that are part of your application interface. The first step is to create a new z/OS Connect EE API project. To do this select File – New Project. In the list scroll down and expand 'z/OS Connect Enterprise Edition' and select z/OS Connect EE API Project. Click next. The detailed actions are easy to follow in the Creating an API Article referenced above.

3. Map the API

Define the Uniform Resource Identifiers (URIs) that make up the API, you can then map fields from the URIs to the fields within the application interface.

4. Deploy the API

Once your API is ready you need to deploy it. The z/OS Connect EE tooling has the ability to deploy your API directly to CICS. This is done through a service in the runtime that is listening on the same HTTP port from which your API will be executed.

Conclusion

It is interesting and useful that APIs have been reinvented and brought forth in a completely new way. The old APIs, access methods like QSAM and VSAM, are vital and still in use. The new APIs have found their use in the data-sharing setting to save costs and improve efficiency.

The microservices APIs are supplementing applications in many businesses to satisfy needs unmet by existing applications and systems. The microservices APIs are making their impact by providing powerful levels of integration, in an application context, without the need for changes to the existing systems of record.

Microservices API programs are not without their challenges. Just like conventional application assets, you design and build the API programs then they are deployed, managed, made highly available as necessary, backed up and eventually retired. These disciplines must be given the proper level of attention in order for the APIs to reliability carry out their purpose.

References

API Connect Components. 2017. Accessed in May 2017 at https://www.ibm.com/support/knowledgecenter/en/SSMNED_5.0.0/com.ibm.apic.overview.doc/capim_overview_apiconnectcomponents.html

Application Response Measurement. 2017. Accessed in May 2017 at https://en.wikipedia.org/wiki/Application_Response_Measurement

Broeckelmann, R. 2017. The Tools of API Management—The Full Stack. Accessed May 2017 from www.linkedin.com/pulse/tools-api-management-full-stack-robert-broeckelmann?trk=v-feed.

Creating an API from a CICS application. 2016. Accessed June 2017 from https://developer.ibm.com/mainframe/docs/getting-started/how-to-expose-a-cics-service/creating-an-api-from-a-cics-application/

Fielding, R. 2000. Architectural Styles and the Design of Network-based Software Architectures. Accessed May 2017 from https://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm

Fiorano API Management. n.d. Accessed May 2017 from http://hkwiseco.com/fiorano-api-management-2/

Getting Started with IBM API Connect Concepts and Architecture Guide. 2016. Accessed May 2017 from http://www.redbooks.ibm.com/redpapers/pdfs/redp5349.pdf

IBM z/OS Connect Enterprise Edition V2 Technical Overview. 2015. Accessed May 2017 from https://www.slideshare.net/UkRobJones/zos-connect-enterprise-edition-v2000-technical-overview

JSON, REST, SOAP, WSDL, and SOA: How do they all link together. 2013. Accessed May 2017 from https://stackoverflow.com/questions/16626021/json-rest-soap-wsdl-and-soa-how-do-they-all-link-together

Queued Sequential Access Method. 2017. Accessed in May 2017 at https://en.wikipedia.org/wiki/Queued_Sequential_Access_Method

Rocheleau, Jake. n.d. The Basics of REST and RESTful API Development. Accessed in May 2017 at http://www.hongkiat.com/blog/rest-restful-api-dev/

SOAP vs. REST: A Look at Two Different API Styles. n.d. Accessed May 2017 from https://www.upwork.com/hiring/development/soap-vs-rest-comparing-two-apis/

Tommaseo, L. 2016. Excellent 2 Pager on z/OS Connect. Accesses May 2017 from https://www.slideshare.net/LuigiTommaseo/excellent-2-pager-on-zos-connect-ent-edition

Welcome to the API Economy. 2016. Gartner Research. Accessed May 2017 from http://www.gartner.com/smarterwithgartner/welcome-to-the-api-economy/