enactor

Enactor.Headless

A tools-driven engine for all channels

About Headless Commerce

Headless Commerce - what do they mean?

Headless Commerce is a term used to describe when the front-end presentation layers of applications, such as websites and Apps, are separate from the back-end and share a common set of services and data-structures, all managed via APIs.

The “headless” aspect, i.e. an application with no User Interface, simply means that there is no front-end that comes with the platform for customer interaction. The application acts as a common layer accessed via web service calls from disparate applications that rely on the application being at least part, if not all, of their back-end functionality.

Why is this desirable?

Historically the most well-known web platforms in the market have had very tight coupling of the front-end and back-end. This has meant that they have been poor at facilitating bringing in other channels to market, even with the APIs that are available.

Tight-coupling also means that Retailers have not had the benefit of being able to experiment with new channels on their own. In spite of the powerful capabilities within web technologies being relatively easy to access, they are still dependent on their supplier’s more monolithic web platforms in order to expedite process change.

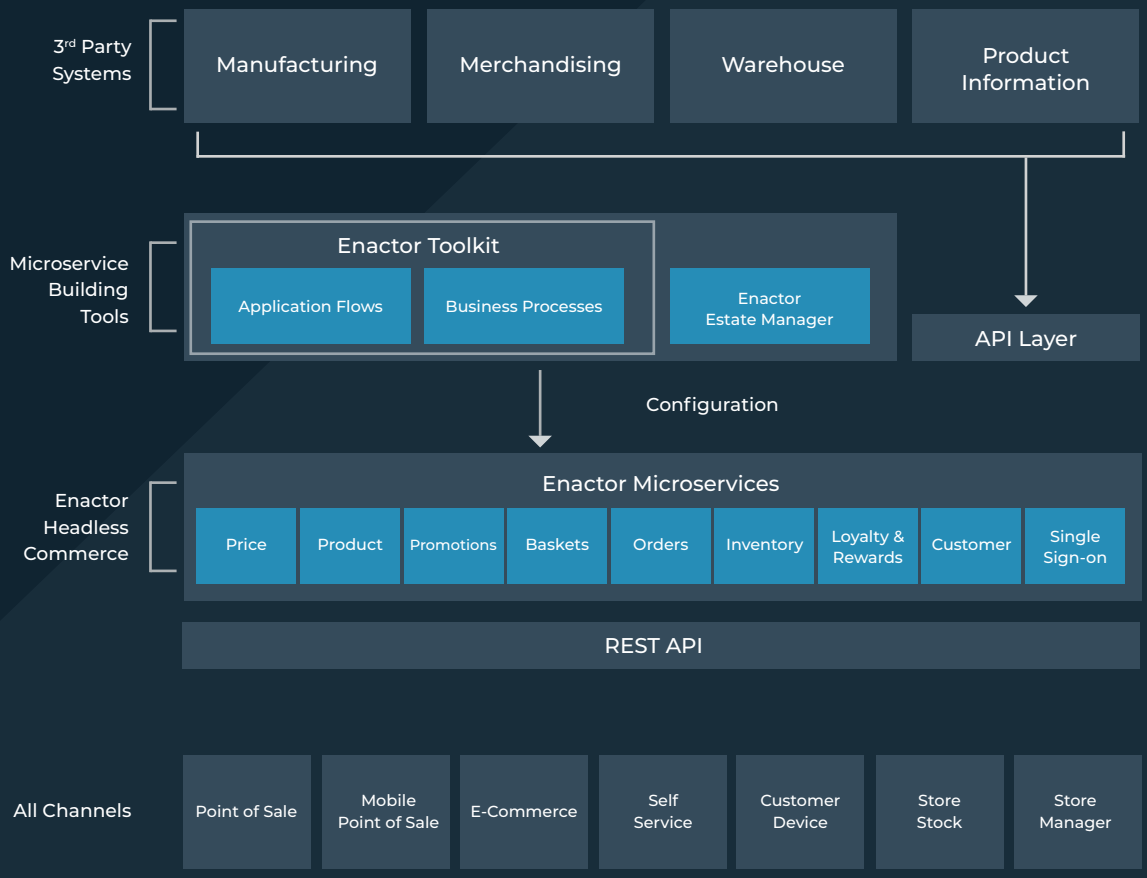
This desire for increased flexibility of the front-end has also been compounded by the rise in Single Page Applications. Most older style, monolithic web platforms have come packaged with traditional Content Management Systems for delivering multipage sites. But with the rise of agile JavaScript Frameworks for web UIs there is a gradual shift away from CMS’ and Multipage frameworks altogether.

The result of this shift is a back-end accessible only through APIs, to better facilitate the “plumbing” aspect of e-commerce. Services for elements such as the basket, product, prices and imagery for example are now delivered via web service calls as opposed to the internal workings of a web platform.

Headless Commerce enabling Unified Commerce

This Headless model enables a way of providing a platform for exceptional Unified Commerce experiences. By having a common set of places and data-models to execute commerce functions across all channels, consistency via all channels is much easier to achieve. In addition, on the front-end there is more control over, and focus on, the customer experience. With the focus shifted from selling to buying, it can be about supporting how the customer wants to buy as a priority over and above how IT infrastructure allows the retailer to sell.

Enactor offering so much more....



Much more than just a database and APIs

At Enactor we know it's not enough to just have APIs that maintain lists of software entities, that share many of the same names as microservices, causing confusion for example around including lists of products, prices, orders etc.

Headless Commerce is very much about delivering genuine functionality to the business, crucial at this level when it comes to large scale, complex retail operations. After all, software is much more than just a database and UI. A good example of business value created with a rich, functional microservices layer lies in the area of promotions.

Typical promotions engines can't keep up with requirements that change and flex depending on consumer behaviour. When a retailer has thousands of SKUs, and many more thousands of purchase combinations impacted by hundreds of product promotions, a strong configurability around the e-commerce layer is crucial. In practice this could mean that promotions are dynamically managed and assigned to a transaction in the front end, to provide the best possible price for the customer based on all the options available.

When the above is multiplied across thousands of transactions, it becomes clear why the functionality at the front end is more important than just simply "passing" lists of information between the front end and the back end via an API. If the latter were the case, the processing speed would slow to a crawl during business periods such as black Friday.

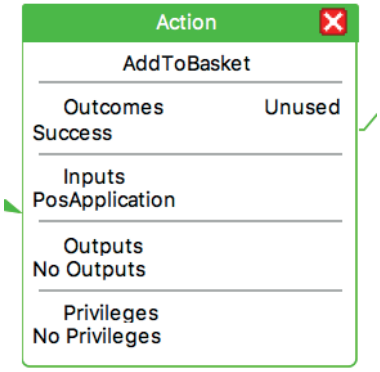
Tooling for Microservices is critical

Enactor has been building applications with "microservices" for many years – before the term even existed. We know that microservices do provide flexibility, but they still require a layer of tooling in order to manipulate them in order to get the most from them. As you can see from our architecture the groups of microservices listed are themselves made-up of smaller microservices, connected together and managed by our various tools.

How the Microservice tooling works

Super-granular Microservices

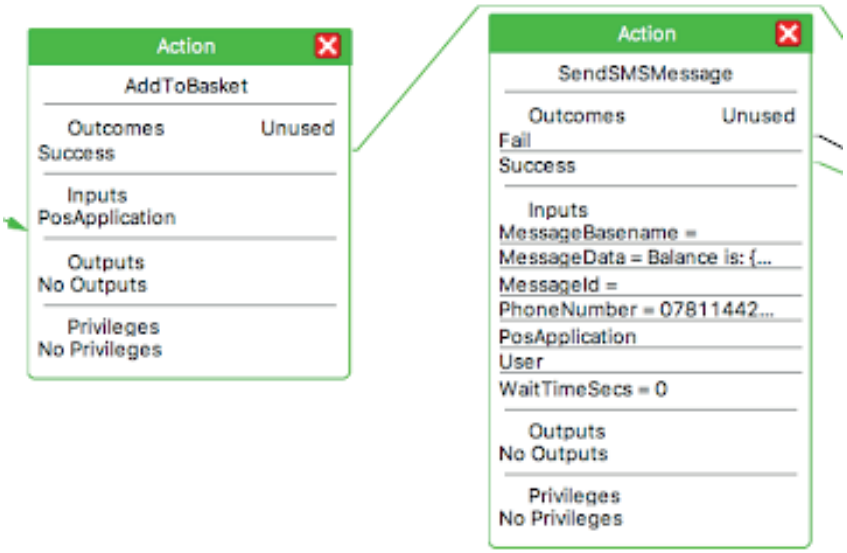
In Enactor there are literally thousands of microservices – probably more than any other platform of its kind. That's because the Framework enforces an incredibly granular level of microservice wrapping. Each Java class is automatically wrapped as a microservice so that it can be invoked either locally or remotely over a network.



The Framework enforces that each microservice defines its inputs, outputs and outcomes in invocation. When building applications we refer to each one as an "Action". All of them are inherently re-useable and, on-created, are stored in our resource library.

Micro-Process based Applications

In order to build applications, we stitch these small microservices together into what we call "Processes". Each process can be remotely and locally invoked over a network and are all therefore microservices in themselves. Therefore all Enactor products whether a; Headless microservice such as the basket service, distributed applications such as the point-of-sale, or central tools including Enactor Estate Manager, have all been created in this way.



Much Richer and more flexible than “Black-boxes”

Headless Commerce and our Microservices

In a Headless microservices architecture there are groups of services and APIs which provide the back-bone to the entire commerce architecture. Typical Headless Commerce providers will simply offer a collection of services, with the key elements being: Product, Prices, Promotions, Customer, Order etc.

But to other platforms these web services are simply black boxes. They either offer little functionality or internally struggle with limited application architectures. Major enterprise retailers don’t choose black-box natured systems for large-scale mission-critical systems so why should Headless Commerce be any different?

Purely because the Enactor Microservice Architecture is so granular, and the process driven nature of it, the flexibility of microservices is driven to within Headless Services as well as outside of it.

Functionality and configurability

What’s important to remember is that the Enactor platform also brings genuine functional richness. Every function is a microservice, meaning that all of our elements are available as a service. The Enactor platform is inherently richer in functionality than most on the market and each group of functions in Enactor is also inherently configurable.

The primary benefits of this approach are; complete enablement of unified commerce, ability to support large scale processing of very complex processing and the ability to change business processes quickly without hard coding.