SOLUTION BRIEF
API-Led, Event-Driven Architecture

Adopting a Distributed API & Event-Driven Architecture

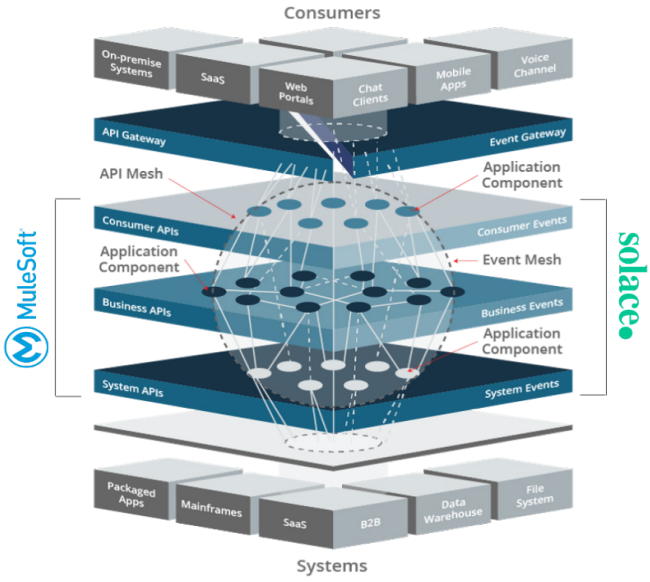
The key to doing business in real-time is establishing a modern digital backbone that enables better and faster decision making to deliver business agility and reduced cost. The modern digital backbone is underpinned by a cloud-native infrastructure that can support a distributed API and event-based architecture. This distributed architecture is heavily based on an interconnected network of APIs and Events. APIs are used for exposing business functionality securely in a consistent manner that is consumable by internal and external entities in a self-service capacity. They are mostly synchronous and stateless in nature. Events, on the other hand, are inherently asynchronous and represent emission of key state changes within and outside the application. These changes can be subscribed to by various consumers, and if the change impacts them, they can take actions in accordance to their expected behaviour.

This is where events complement APIs perfectly, and together, they can be combined to solve almost every communication problem within or outside an application in a loosely-coupled manner and thereby enable real-time business. Event-driven architecture (EDA) is a software architecture paradigm promoting the production, detection, consumption of, and reaction to events. EDA is common in microservices, cloud native, IOT, event-driven Integration, streaming to AI/ML. The Publish-Subscribe (Pub-Sub) concept relies on events - "producers" describe the event they are publishing with event tags, known as topic taxonomy and "consumers" attract events using patterns and wildcards - this is true decoupling. Broadcasting events from the publisher (decoupling) allows additional uses of events later. This is key to agility.

Adopting event-driven technology cuts across the entire enterprise IT landscape of organisations. It's important to remember that it's not a replacement for existing thinking, but a way of getting value from the events that are happening in an organisation. We can think of becoming event- driven as a 3 step process.

- Firstly we have events already occurring in our systems. Publishing these events as they happen liberates the events from their respective data silo's and allows innovation decoupled from the existing applications.
- As we modernise and digitise more of the platform we can extend the digital backbone into cloud and modern microservices -based applications. Events also act as a buffer to isolate legacy applications from bursty modern applications.
- Finally we can extend participation to partners, customers and other LoBs allowing real time interaction and visibility.

Our Cloud-Native Architecture Blueprint provides a best of breed, architecture leveraging market-leading platforms - Solace enabling the distributed event-mesh and MuleSoft Anypoint Platform managing the application network - complementing each other perfectly to enable the real-time business.



Cloud-Native Architecture Blueprint

Conceptual Architecture

RUBICON RED

Application Components

Application components are the actual modules built out by application teams and they help in dividing the logical application into discrete functional blocks interacting with each other. Application Components expose their functionality in the form of APIs. They also publish events occurring within them for consumption by other Application Components. Application Components often have to interact with APIs of other Application Components and subscribe to relevant Events published by them.

Systems

Most cloud native applications do not exist in isolation. They are built on top of existing systems and have bi-directional interaction between them. These systems may be an organisation's existing assets such as legacy mainframes and packaged ERPs, or SaaS solutions. These are primarily the systems of record, which change remain fairly stable in terms of functionality, over time. Systems such as bespoke databases and enterprise data warehouses also fall within this category. These systems of record serve key business processes by implementing valuable business functionality.

System APIs

With systems of record in place, we need a way of unlocking the value of these important assets for consumption by other modules within cloud-native applications. System APIs are meant to serve this purpose. They act as a thin wrapper around the end systems to expose key business functionality for access in a managed, secure and consistent manner. They adhere to the same API design standards that are laid out across the entire organisation.

System Events

System Events represent the significant events generated when either changes made in the end system need to be bubbled up for consumption by interested subscribers, or when important changes need to be registered within the end system in an asynchronous manner. This covers both the events that originate out of end systems as well as events that result in changes being reflected inside the end systems.

Business APIs

While System APIs are primarily wrappers around end systems, Business APIs are more focused on higher-value, system-agnostic business functionality. These are the APIs that truly realise holistic business processes, which may span across multiple systems. These APIs are perfect candidates for exposing the application's functionality to the outside world.

Business Events

Complementary to Business APIs, Business Events represent the significant business events that need to be emitted or consumed from the business layer. The business layer is the layer that is truly business-led in nature and is not designed to cater to the needs of any particular consumer nor is it linked to any particular end-system. It is a purely agnostic layer and hence serves the meat of the value offered by a cloud-native application.

Consumers

Consumers are the entities that consume the application and reap the business value from the application. Consumers could well be end systems themselves who use the functionality provided by this cloud-native application to in turn realise their own functionality. Any system, internal or external, web apps, mobile apps, external API consumers and such come under this category.

Consumer APIs

Consumers often have specific requirements and cannot readily consume APIs exposed by the business or the system layer. For example a mobile consumer needs tailor-made dat ato render on a specific screen. We can design a special API layer, fully catering to the requirements of the consumer, that feeds off APIs exposed by the other layers but massages information to make it more consumable. These APIs are often owned and created by the owners of the consumer application itself. For example, the developer who creates a mobile app might create a suitable Consumer API as well. Different stakeholders can effectively contribute independently to build out logical modules. The only requirement is to enforce standards around how data is consistently exposed in the form of APIs and Events, and where this information needs to be registered, discovered and self-served from.

Consumer Events

These are events that are published in a way that can be consumed by multiple interested consumers. These can also include rare scenarios where bi-directional communication is needed, in the sense that a cloud native application may also need to open up asynchronous channels to consumers for them to post events which the application can then use to take appropriate actions within it. For example, there could be an asynchronous response to an event published to an end-consumer, and, due to technical constraints, the only possible option might be to open up a response event channel for it to send its response.

Best of Breed Platform

API Mesh/Application Network

The individual application components need to come together and interact with each other to realise the entire functionality of the cloud-native application. Intercommunication between a component and APIs exposed by other components is an obvious requirement, and this is where we need an API mesh or application network to enable interaction between different components while freeing up the API developers from common concerns such as service discovery, throttling and background retries. The key role of the application network is to facilitate communication between different APIs within the application.

Another important role of the application network is to act as a central point of management and observability. Since all communication between API components can be monitored by the mesh, it now enables the mesh to analyse traffic and report valuable metrics for the entire distributed application. It can show performance bottlenecks, overloaded APIs, geographical distribution of API requests and other information.

MuleSoft's Anypoint Platform is the world's leading application network platform for full API lifecycle management APIs and integration of any system across clouds. Typical use cases include Event Driven Integrations, IoT connectivity, Microservices, ETL, SaaS Integration, and Legacy SOA replacement. MuleSoft provides exceptional business agility to companies by connecting applications, data, and devices, both on-premise and in the cloud with an API-led approach. By leveraging Anypoint Platform, companies can accelerate their digital transformation by connecting and exposing data held in their legacy systems, proprietary platforms, and custom data stores and do so in an organized, secure, and manageable way.

MuleSoft customers can realise a microservices architecture using Anypoint Platform and the API-led approach. These microservices leverage the Mule runtime engine, and become a part of the customers application network. For any existing non-MuleSoft microservice, Anypoint Service Mesh, allow organisations to extend the benefits of the application network, allowing them to:

- Empower innovation teams to build with polyglot technologies that best align to their skillsets.
- Accelerate microservice adoption with discovery and re-use of non-Mule services.
- Maintain flexibility across services with a network that is built for change.

API Gateway

The boundaries of a cloud-native application are quite flexible and dependent on organisation context and how it is structured. Given the distributed and modular nature of cloud-native applications, it may be easy to imagine it as an infinitely growing entity. However, this will not be the case and various purely business-led factors will come into play when deciding the logical boundary of such an application.

Once this boundary is ascertained, we can conceptualise two types of traffic flowing through the application at any time. 'East-West' traffic, as it is popularly called, is the flow of information within the APIs of the same application. This traffic flows purely for the reason that it allows the application to function properly by effective intercommunication between relevant application components. The other type of traffic, popularly known as 'North-South' traffic, is where data flows into the application from outside the application boundary. This traffic needs to be monitored and governed a lot more stringently, as it represents the external ecosystem of the application and hence consumers who are beyond the line of control of the application. This is where we say that all such traffic needs to go through an API Gateway component, which is the single gate through which any API traffic can flow into the application from outside.

The API Gateway can enforce application-wide security policies, rate-limiting controls for metered access, provide smart caching for faster response times and more. It also acts as a management pane for analysing all external traffic flowing into the application.

MuleSoft's industry-leading API management platform provides end-to-end, enterprise-grade security, including a high-performance API gateway component. The API gateway points to the backend APIs and services that you define and abstracts them into a layer that Anypoint Platform manages. Consumer applications invoke your services. APIs route to the endpoints that the gateway exposes to enforce runtime policies and collect and track analytics data. The API gateway acts as a dedicated orchestration layer for all your backend APIs to separate orchestration from implementation concerns. The gateway leverages the governance capabilities of API Manager, so that you can apply throttling, security, and other policies to your APIs.

Best of Breed Platform

Event Mesh

As a perfect complement to the API-driven application network, the Event Mesh represents a distributed set of event brokers that host the channels over which all events are exchanged between the different application components. The asynchronous protocols supported by the mesh may need to be quite vast to cater to the needs of different kinds of systems and consumers. In these cases, more than one messaging technology can be used to realise the holistic Event Mesh. Similar to APIs, the Event Mesh also plays a significant part in observing traffic as it flows within the distributed application.

Solace's PubSub+ platform provides a comprehensive way to create, document, discover and stream events from where they are produced to where they need to be consumed – securely, reliably, quickly, and guaranteed. A Solace PubSub+ powered event mesh spans hybrid, multi-cloud, and IoT environments (locally, regionally, globally) and provides intelligent and dynamic event/message routing with built-in WAN optimization, hierarchical topics and wildcards and support for multiple open standard protocols & APIs.

Event Broker

An Event broker sits between event producers and Event consumers, routing and filtering the necessary event data between parties. The Solace Event Broker is a modern message broker available as an appliance, software and cloud-managed service. It efficiently streams events and information across cloud, on-premises and IoT environments. The "+" in PubSub+ means it supports a wide range of message exchange patterns beyond publish/subscribe, including request/reply, streaming and replay, as well as different qualities of service, such as best effort and guaranteed delivery. All deployment options offer the same functionality and management experience.

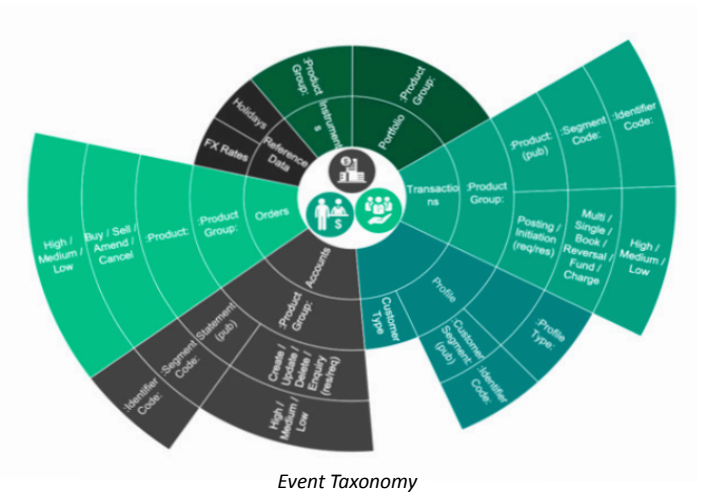
Event Portal

An event Portal provides visibility and governance across a distributed Event Mesh and facilitates re-use and conceptually similar to an API Portal. The Solace PubSub+ Event Portal is a single place to design, discover, catalog, visualize, share, secure and manage all events in your ecosystem. It allows you to:

- Visualize, optimize, and govern event flow across your distributed enterprise.
- Design events & event-driven apps, with best practices built-in.
- Discover and catalog events from across the business.
- Fully automate CI:CD pipelines via it's rich API & support for AsyncAPI.

Event Taxonomy

Topic routing is the lifeblood of an event-driven architecture. Topics are metadata of events; tags of the form a/b/c – just like an HTTP URL or a File Path – which describe the event. The event broker understands topics and can route events based on who subscribed to them, including wildcard subscriptions.



Event Taxonomy

In Summary

In almost every industry today there is a need to do business in real-time, where speed and precision are the new value-chain drivers. Those organisations that can quickly modernise their existing environments with a cloud native infrastructure that embraces API-enabled and event-driven concepts will be best placed to repond to real-time events, making better decisions, faster providing the agility to respond faster to customer demands, supply chain interruptions or opportunities for innovation - all at reduced cost.

Talk to Rubicon Red today and discover first hand how we can help you build a roadmap that identifies and delivers the events and APIs that are key to enabling your real-time business and implements an event-driven architecture that will drive your next phase of growth.

www.rubiconred.com

1300 799 959