Event Based Architecture Introduction

In this section we present a small set of linked patterns that represent the fundamental building blocks of an Event Driven architecture. An Event Driven approach is an important (some would say essential) extension to the services approach that supports decoupling services in a way that is not just compatible with a cloud approach, but in many cases, absolutely required when you are pursuing a Hybrid Cloud approach. For more information on Why an Event Driven approach can help, see this sidebar.

As we are advocating the use of an Event Driven Architecture within a cloud architecture, most likely within a Microservices approach, it's worth discussing briefly when an event-based or request-response approach is more appropriate. There are no hard and fast rules, but here are some considerations:

A component should use a request-response (HTTP/REST) approach to send a service request when:

- It knows exactly what service it wants to have run
- It wants it run the service exactly once
- It needs to know when the service completes successfully
- It wants to receive the results of the service invocation

A component should use an Event-driven approach to announce an event when:

- It wants to notify many potential consumers that might be interested in an occurance
- It doesn't need to know what consumers may be interested in an event
- It doesn't need to know how each consumer may react to an event, or indeed expects that different consumers may react differently.
- The communication is one-way: The announcer doesn't want to wait for the consumers to react, indeed doesn't know when they've all reacted, and isn't interested in the results from the reactions

In this style of interaction, the announcer sends a notification and then is unconcerned with the outcome.

This set of tightly related patterns describes the basic issue of how to construct an Event Driven Architecture from component parts. The patterns discussed are:

- Event Driven Architecture extends the services concepts of components and loose coupling with a new concept of event coupling to further enable evolution of families of applications, in which new components can be added to the family, enriching the services provided and the contexts triggering use of those services, without altering interactions between preexisting components and applications in the family.
- Event Backbone provides the shared medium allowing multiple Event receivers to obtain events from multiple Event providers without having to know the identity or physical location of their event coupled partners; the Event Backbone also insulates Event receivers and producers from awareness of the network topology and message transport protocols used to receive, move and deliver events.
- Event Sourcing allows you to represent complex application state by scanning over an event stream.
- Event Joiner helps you tie together related events that form a logical stream of events in order to process the elements in the stream as a group or to deliver them individually to an appropriate processing context where they can be handled.
- Event Deriver can build new events from current and old events based on comparisons to previous event data, including temporal information.
- Event Aggregator helps you to avoid flooding your Event Backbone with low-level events while allowing other applications dependent upon the results of those events to function.
- Event Topic Registry provides a way for decoupled event producers and consumers to share essential metadata about events. If metadata sharing was not possible, event producers would not be able to make event consumers aware of events and event messages, and event consumers would have no way to discover and subscribe to pertinent events.
- Event Consumer decouples components by allowing different components (microservices) to only register for and receive those specific events they are interested in.

The relationships between most of the patterns in this section are shown in the diagram below:

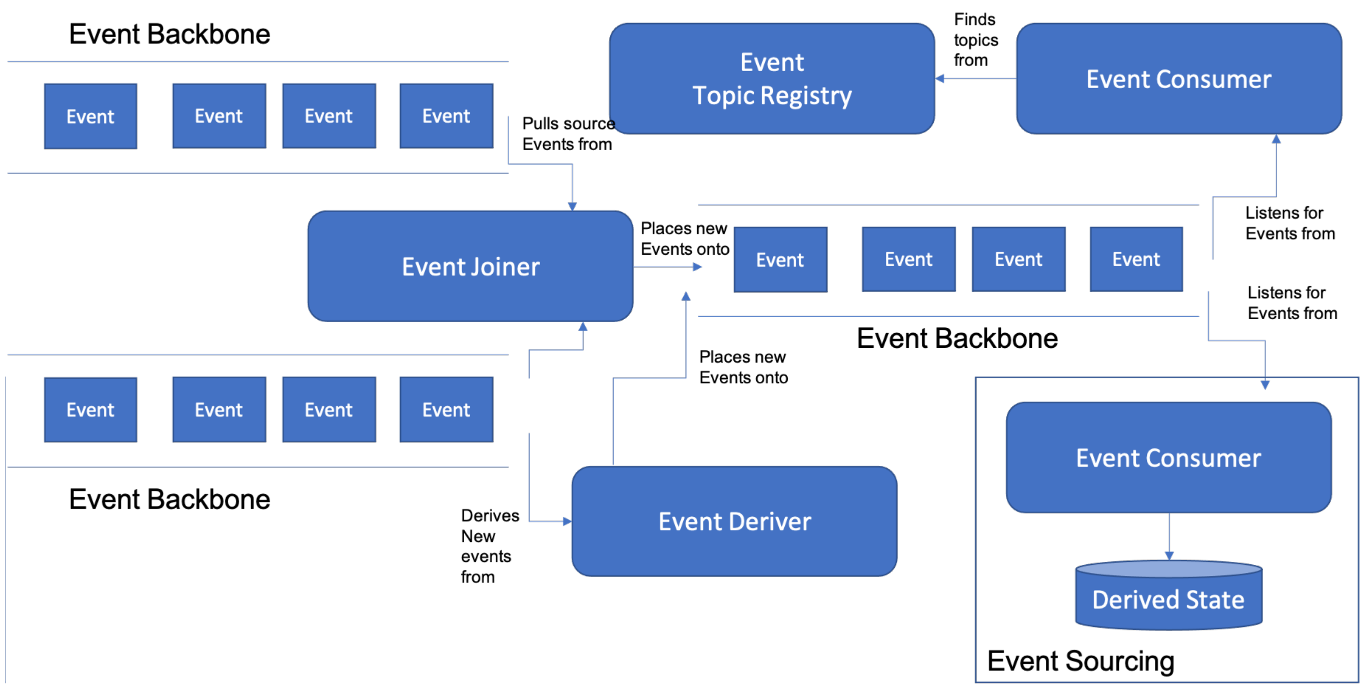

TABLE OF CONTENTS