IDC

SPOTLIGHT

Sponsored by: SAP

Event-driven automation is quickly becoming a mainstream architecture for data processing, orchestration, and situational awareness. Companies are leveraging it to rethink their approach to integration for more resiliency, speed, and flexibility.

Integration Reimagined: Preparing for the Event-Driven Automation Revolution

September 2022

Written by: Shari Lava, Research Director, Automation, and Maureen Fleming, Program Vice President, Intelligent Process Automation

Introduction

Businesses increasingly recognize they need to move faster and become more situationally aware — not just internally but also externally with customers and across their ecosystems. Situational awareness is the ability for people and systems to understand the current state and quickly react or adapt as necessary.

Executives may be keenly aware that scalability problems during peak periods will cause loss of business. They may be aware of order cancellations when inventory systems are inaccurate or of lost opportunities when their lead time is too slow. They may also notice increasing competition from digital upstarts and competitors that combine artificial intelligence (AI) with real-time signals to offer new types of innovative services.

Additionally, there is a much better understanding that providing excellent customer service means customers expect to be informed about status changes that impact the delivery of their orders or execution of a service.

Organizations trying to solve these issues increasingly understand that they gain an economic advantage when they can identify a problem immediately at the root cause. They also understand the competitive benefits of faster cycle times. Today, speeding up cycle times, operating with more resilience, and systematically becoming more customer aware mean that an organization requires a different type of information flow that is built around events.

AT A GLANCE

KEY STATS

While recognition of the importance of event-driven architectures is still growing, early adopters are quickly taking full advantage of this technology. According to IDC research:

- By 2023, nearly two-thirds of companies plan to deploy some event brokering tools, with over half having already started.
- Of the companies that have implemented event-driven tools, 55.8% have implemented only a few use cases, meaning event-driven automation is not yet a core integration architecture principle, which may create rework in the future.

An event is a state change (for more information, see the Definitions section). Physically turning a light switch from off to on is an example of a state change. In streaming services, hitting the pause button and then later resuming are two state changes that allow us to be entertained and not frustrated. In applications, a state change may be a new order, a change in an order, or a cancellation. It may be the addition of one new unit of inventory or the reduction of inventory caused by a purchase.

While there is nothing really new about an event, changes in how events are collected, propagated, and used can transform an organization from one that is slow to capture, understand, and respond to problems and opportunities to one that is more predictive and proactive about how it does business. Event-driven design is the primary model for much of the innovation that is commonly associated with digital-native successes. Elastic cloud infrastructure and digital services such as turn-by-turn navigation, ride sharing, food delivery, and modern logistics emit events that are then used to make and update predictions such as when more or less capacity is needed, when to turn when driving, how long it will take to arrive at a destination, and when an order will arrive. These innovation-based services invest in managing expectations while building systems that manage outcomes. Their competitive advantage is in being fully situationally aware.

Event-Driven Design Is Both More Critical and Much More Approachable

Asynchronous event-driven design and an event-driven architecture were first used in the late 1980s with the introduction of digital messaging technology that would produce events as data and publish them. Systems interested in receiving these events would subscribe to them. Publish and subscribe (PubSub) became a foundational integration pattern for high-speed, high-volume mission-critical systems, such as automated financial trading and execution systems and for straight-through processing used in payment processing. Because these ultra-low latency systems were traditionally expensive to build and maintain, organizations narrowly adopted PubSub for specific use cases.

Over the past decade, the use and value of events began to be reconsidered for several reasons, including:

- New types of ephemeral data were event based, and there was value in collecting the events. Examples are Twitter feeds, mobile messages, and sensor data.
- The event logs of databases contain valuable data at an atomic level that can be continuously collected.
- Events contain time stamps. That means events can be collected from a variety of distributed sources, aggregated, correlated, organized into a time-sequenced order, and broadcast to subscribing systems.
- Time-series analysis, correlation, and machine learning can be applied to time-sequenced events to make recommendations and predictions and support decision making.
- Major application providers began to output their data as events.

Resolving the Problem of Utilizing Events at Different Speeds

The speed of the technology used to support PubSub was too fast for many use cases. Organizations wanted to speed up their processes and analytics capabilities, but fast is a relative term. For example, hourly collection is faster than daily collection, but subsecond collection may be too fast. By contrast, a service generating near-real-time interactive recommendations by continuously collecting and correlating events from several sources requires the fastest possible delivery speeds.

Apache Kafka was an early open source project that solved this problem by collecting log data events at speeds set by the team supporting a project. Kafka collected events from the log data of databases. It also received events that were streamed by data providers. All these events were collected and stored in time-order sequence into immutable topics — or distributed log files. Systems that needed these events would either subscribe to a topic for continuous delivery or replicate the event data in batch.

This approach accelerated the use of events, elevated the importance of events, and made it easier for organizations to utilize event data at different rates of speed, which gave organizations time to gradually adapt their systems to operate at higher speeds. As a result, PubSub and event-driven architecture are more widely adopted.

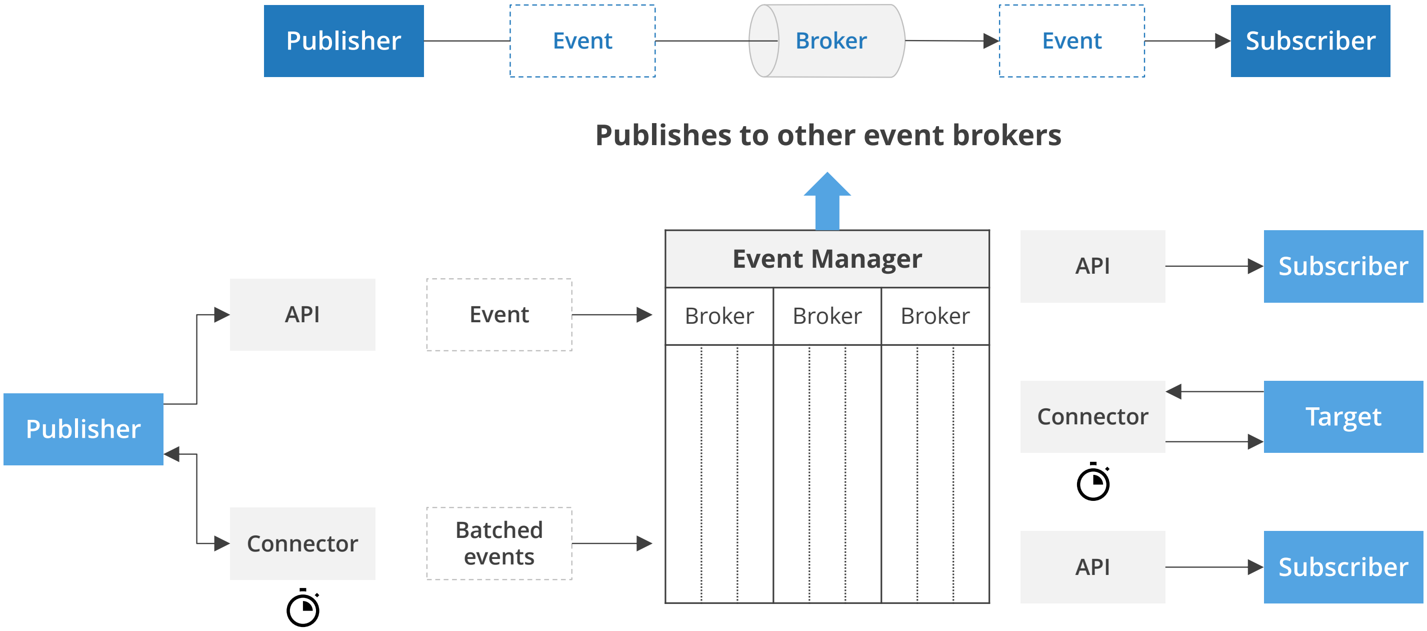
Emergence of Event Brokers and Event Meshes

The technology that originally supported PubSub was messaging middleware that collected messages temporarily into queues. With the introduction of new types of PubSub technology from open source providers, including Apache Kafka, Apache Pulsar, and Redis, along with newer commercial offerings, the category has been modified to support storing or caching event data. This replaces the need for queuing while making the data permanently accessible. "Event" was elevated, and the category is now called "event brokers." The category also supports all types of deployment models, including software, SaaS, and private cloud hosting of event broker software. As Figure 1 shows, event brokers also become event producers. Event brokers can subscribe to the topics of other event brokers. This creates diverse types of event brokers and broker resiliency.

In Figure 1, the event broker shown at the bottom of the illustration (Broker 1) is an event manager that supports different speeds. The event broker shown at the top of the illustration (Broker 2) is focused on high-speed delivery of events. The events may pass through or may be cached by Broker 2.

All these technologies support reliable delivery of events and have callback mechanisms to acknowledge receipt of events along with a retry mechanism when acknowledgement is not received. Efforts are made to ensure that events delivered are not duplicates.

FIGURE 1: ***Event Meshes Publish to Each Other***



Source: IDC ,2022

When Broker 1 communicates with Broker 2, a mesh is formed. Event meshes are responsible for delivering events across distributed systems. Different broker participants in a mesh may have different roles. For example, one broker may be added for capacity reasons, another may be added to support low-latency requirements, and still another may filter for specific types of events, such as an order. The order may be captured by an implementation of Apache Kafka as part of a larger collection, and another broker in the mesh may subscribe to orders to broadcast to all applications that are listening to that event broker for new orders. Meshes also support both parallel processing and sequential processing.

Convergence of Event Brokers and iPaaS

An iPaaS typically consists of APIs, API management, and connectors to databases. It supports sequential orchestration where a transaction can be processed straight through across the different applications involved with the processing. An iPaaS also maintains a library, or catalog, of prebuilt APIs and connectors that can be dragged into the development environment and configured. Since an iPaaS supports integration, mapping and transformations are enabled to ensure source data will be converted to the target data structure.

REST APIs are a critical asset in an iPaaS, underpinning both request-style commands, such as GET, and push-style commands, such as POST. Push style means an API supports the delivery of an event, which is often used to trigger an action. REST APIs support one-to-one interactions. AsyncAPI is an emerging specification for defining an API that supports one-to-many asynchronous interactions. AsyncAPI, in essence, supports event producers and event subscribers, like an event broker. AsyncAPI provides a mechanism for packaging PubSub into a callable service. The problem is that event brokers are better at handling PubSub, but customers are far more likely to want to utilize their iPaaS catalog to invoke the PubSub service.

As events grow in importance and more urgency is placed on operating at faster clock speeds, it is logical that there will be a convergence of event brokers with iPaaS through partnerships, OEM relationships, or acquisitions, especially as events and PubSub become dominant for high-speed situational awareness.

Becoming Event-Driven Doesn't Mean Saying Goodbye to APIs and Connectors

APIs are a core part of integrating data from events into the company, including both REST-based APIs that use request/response patterns and the emerging AsyncAPI protocol that more natively supports PubSub patterns more frequently used by events. APIs are also important in themselves for automatically kicking off subsequent actions in processes, including pushing or pulling data from external sources. In addition, data on demand, web applications, monetization of data/services, and participation in marketplaces will continue to rely on APIs as a front end to events.

An enterprise iPaaS is key to getting all these technologies to work together. It ensures events and the resulting data are processed correctly, making integrations more fault tolerant with less bandwidth demand. The iPaaS system also is important to maintain consistent mapping and proper data transformation when syncing data across the enterprise, including data from event streams. The integration platform must work seamlessly with event brokers and/or event meshes to respond to events, coordinate resulting data updates across systems, and handle any exceptions in processing.

Benefits

Companies that use event-driven automation in their integration mix unlock several key technical and business benefits that both provide new insights and make data across the company more reliable.

Business Benefits

- **Incorporate new data.** Tapping into unleveraged data streams, whether from internal or external sources, can augment organizational insights, providing a more holistic picture. When data streams are incorporated properly and in a timely manner, it leads to faster and better business decisions by employees throughout the company.
- **Increase response time.** Real-time processing of business events makes it easier to respond faster with every customer, employee, and partner interaction in two ways. First, it enables broader automation, reducing wait times and manual interventions between process steps. Second, it ensures responses are always fueled by the most up-to-date data, regardless of the application being used.

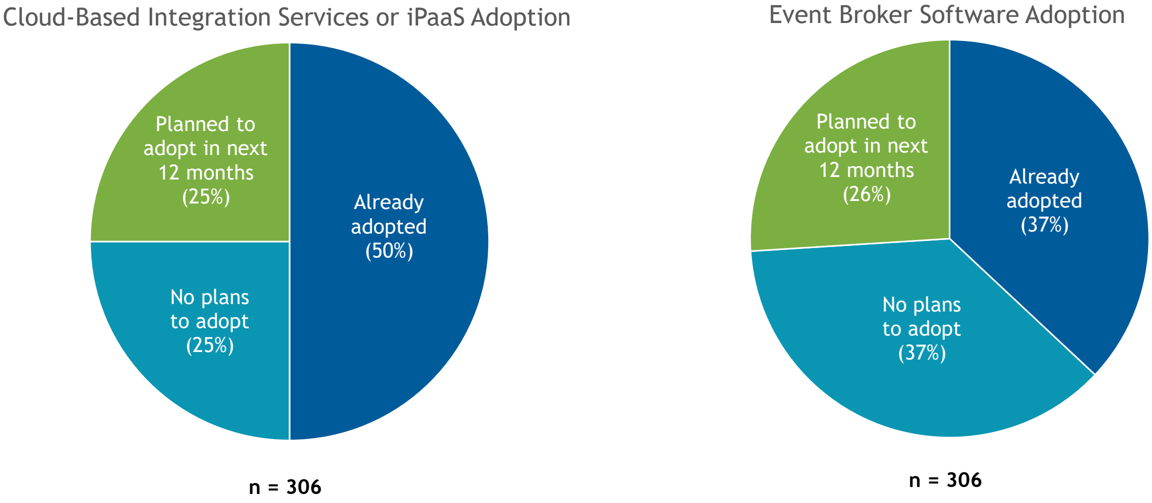
Technical Benefits

- **Integrations become decoupled.** Integrations are broken down into smaller components, making it easier to reuse components in other integration patterns and applications. This reduces the overall administrative burden on IT to manage point-to-point integrations, creating a more flexible architecture.
- **Better analytics are enabled.** Companies with multiple SaaS solutions from multiple providers can use events as part of their business process integration/orchestration as well as to consolidate data changes in a central data store for analytics, bringing about a convergence of process and data integration specifically with SaaS applications.

Trends

Cloud integration tools are already much more widely adopted than event brokering tools (see Figure 2). This may indicate that companies are unaware of how events can and should be part of a holistic integration architecture to support new patterns and solve business needs better.

FIGURE 2: ***Adoption of Cloud Integration Tools Versus Event Broker Software***



Source: IDC's U.S. Application Integration and Automation Survey, December 2021

Furthermore, nearly 75% of companies that have already adopted event broker tools did so more than a year ago, and only about half have implemented beyond the initial use case, according to IDC research. This behavior can indicate an uncoordinated integration strategy and may hamper the benefits that can be unlocked by using event-driven automation with integration.

Considering SAP

SAP Integration Suite, advanced event mesh is a new service capability within SAP Integration Suite, which is part of the Business Technology Platform (BTP). It is intended to complement the existing Integration Suite offering to support emerging integration patterns that disseminate information across the enterprise in a scalable, resilient, and real-time fashion, enabling the business to operate with the most current information. It includes additional cross-cloud, on-premises, and edge deployments of event brokers intended to support complex, hybrid infrastructure.

The AsyncAPI, transaction and replay support, and visualization of decentralized event streaming can also make it easier to deploy and manage event stream processing.

SAP's event mesh capabilities are meant to help organizations achieve seamless business process automation, accelerate the connectivity of business systems, and progress more rapidly on their integration modernization journey.

Challenges

A key challenge of using event mesh and brokering is the limited developer experience with architecting event-driven automation, which can limit the speed of delivery and hamper the realization of benefits. Developers are well versed in using APIs and developing stateful web applications, but developing event-based and stateless automation requires updated skills.

In addition, the advanced event mesh is a newer offering. While new, it aligns with the rapidly growing adoption of events and event-driven design, bringing an important capability to SAP Integration Suite that can more natively respond to business events produced from SAP core solutions. In general, SAP Integration Suite is likely to be leveraged by existing SAP customers and more likely to be leveraged by those on cloud-based products.

Conclusion

Companies that are modernizing their integration architecture need to carefully consider how event-driven automation can enhance the speed and reliability of integration and better prepare the business for a faster, more responsive future. Organizations need to ensure that any integration tools deployed also include this capability to better future proof their technology investment.

Definitions

AsyncAPI: An open specification for creating an asynchronous application programming interface that supports one-to-one and one-to-many interactions, in essence supporting PubSub integration patterns

Enterprise integration platform as a service (EiPaaS): A set of integration capabilities delivered as a service consisting of development and testing, integration management, and runtime capabilities (With EiPaaS, there is no software locally installed on the developer's machine, and all management of cloud and on-premises integration is controlled and monitored in the cloud.)

Event: A data state change or status change (Events are represented as data including header information about the event, the event data, and a time stamp.)

Event-driven automation (EDA): An approach to building systems that includes event-driven technologies that publish and process events and deliver messages using event brokers, event stream processing, and messaging (Event-driven automation is often used interchangeably with event-driven architecture.)

Event broker: A middleware application, service, or appliance that transmits events between producers and consumers of events (Unlike traditional messaging brokers that are on premises, event brokers run in multiple deployment models, making them a useful part of the integration infrastructure for hybrid environments.)

Event mesh: A networked collection of event brokers that routes events dynamically across a distributed computing environment (An event mesh can help prevent overloading a particular broker during peak loads to create resiliency and redundancy while reducing latency.)

Industry ecosystem: A collection of participants in the same industry that partner to create goods and services for an end customer (A classic example is all the supply chain partners needed to manufacture a good that eventually is sold to an end customer. Ecosystems can go beyond manufacturing to include many industries, such as banking, professional services, and healthcare.)

Integration pattern: A design for interacting with and connecting distributed systems and data (These patterns are treated as assets that can be technologically enabled and reused. Individual integration patterns are building blocks that can combine with others to form a more comprehensive pattern.)

Publish and subscribe (PubSub): The backbone design pattern for event-driven systems (Event-driven design is used to build systems that can detect, consume, and react to events. This type of architecture is core to the development of high-speed, highly scalable systems. Event-driven architecture is high speed because the design is associated with rapid delivery of events. It is scalable because the publisher role can broadcast to any number of subscribers, with every part of the interaction operating independently of each other. PubSub is a type of asynchronous broadcast pattern supporting distributed computing where a publisher broadcasts events and any authorized consumer may subscribe to them. Events can be stored temporarily in queues or caches or permanently in topics.)

About the Analysts

Shari Lava, *Research Director, Automation*

Shari Lava is Research Director, Automation within the AI and Automation Group. Shari's core research coverage includes the fast-changing automation and API management software space. Based on her background in the enterprise applications and data integration, her research also includes an emphasis on understanding the market for and adoption of key technologies that are integral to business productivity and success. Shari works remotely in Toronto, Canada.

Maureen Fleming, *Program Vice President, Intelligent Process Automation*

Maureen Fleming is Program Vice President for IDC's Intelligent Process Automation research. In this role, she focuses on a portfolio of technologies used by enterprises to speed up, drive cost out of, and support a customer-centric approach to business operations. She especially focuses on the convergence of AI, machine learning, and automation and how that combination changes the economics and benefits of process improvement.

MESSAGE FROM THE SPONSOR

Build Modern Situation-Aware Applications Using SAP BTP with the New Event Mesh Capabilities of SAP Integration Suite

SAP provides a complete enterprise integration platform-as-a-service including event mesh capabilities. Build situational-aware applications using event brokers that natively respond to SAP applications as well as non-SAP applications that can be deployed cross clouds and on-premises in distributed environments. By building situation-aware applications, you can respond faster to customers, employees, and partners as well as handle peak loads without impacting operations.

Learn more about SAP Integration Suite, an SAP BTP solution that connects and automates your business building upon SAP's business process expertise to bring together comprehensive integration capabilities and best practices to accelerate and modernize integration, helping you innovate and achieve seamless business processes efficiently across your heterogenous IT landscape and business network.

IDC Custom Solutions

The content in this paper was adapted from existing IDC research published on www.idc.com.

IDC Research, Inc.
140 Kendrick Street
Building B
Needham, MA 02494, USA
T 508.872.8200
F 508.935.4015
Twitter @IDC
idc-insights-community.com
www.idc.com