

# Integration in 2024 and beyond

A view on integration in 2024 and beyond by the A-Team

May, 2024, Version 1  
Copyright © 2024, Oracle and/or its affiliates  
Public

## Purpose Statement

This document provides an overview of the integration software domain in 2024. It is intended to help you understand the integration pillars and overall domain in 2024 and into the future.

## Disclaimer

This document is for informational purposes only and is intended as an opinion on our view of the state of the industry in the integration domain. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described in this document remains at the sole discretion of Oracle .

## Revision History

DATE	VERSION	REVISION
May 2024	1	Initial publication

# Table of Contents

<b>Purpose Statement</b>	<b>2</b>
<b>Disclaimer</b>	<b>2</b>
<b>Revision History</b>	<b>2</b>
<b>Table of Contents</b>	<b>3</b>
<b>List of Figures and Tables</b>	<b>5</b>
<b>A. Overview</b>	<b>6</b>
The Structure of This Document	6
About the Authors	6
<b>B. Three Main Pillars of Integration</b>	<b>6</b>
Application Integration	6
Data Integration	7
Process Integration	7
Other Areas: RPA—Robotic Process Integration	7
The Three Pillars and Our Focus	8
<b>C. The History of Integration and the Trajectory in 2024</b>	<b>8</b>
History	8
Manual Integration	8
Middleware	9
Messaging and Busses	9
Low Code	9
AI as a Helper	10
AI Driven	10
<b>D. Application Integration—A Deeper Dive</b>	<b>10</b>
Approaches to Application Integration	10
Integration Platform (iPaaS)	10
Build It Yourself	12
<b>E. Process Integration—A Deeper Dive</b>	<b>13</b>
Key Components in Process Integration	13
Upcoming Developments in Process Integration	15
Oracle Process Automation Cloud (OPA)	15
<b>F. Trends and Themes in Integration</b>	<b>15</b>
<b>Deployment Trends</b>	<b>16</b>
Prebuilt Integrations	16
Features in a Modern iPaaS Platform for Prebuilt Support	16
Advantages of Prebuilt Integrations	17
Disadvantages of Prebuilt Integrations	17
Hybrid and Multi-Cloud Integrations	18
Introduction	18

Hybrid Cloud and Multi-Cloud Environments	18
Challenges	18
Capabilities	19
Benefits	19
<b>Development Trends</b>	<b>20</b>
AI and Machine Learning	20
Introduction	20
AI Assistants and Generative AI	21
Some Tooling Examples Where AI is Being Used	21
Advantages of Using AI Tooling	21
Disadvantages of Using AI Tooling	22
Solution Consideration	22
API Driven Integration Development	22
Introduction	22
API Monetization	24
Advantages of API First Integration	24
Disadvantages of API First Integration	24
Solution Considerations	25
No-Code and Low-Code Development Environments	26
Introduction	26
Capabilities of Modern iPaaS Platform to Support Low Code or Drag & Drop Features	26
Advantages of No-Code and Low-Code	26
Disadvantages of No-Code and Low-Code	27
Solution Considerations	27
Event-Driven Architecture	28
Introduction	28
Publish-and-Subscribe	29
Streaming (Produce/Consume)	30
Advantages of an Event-Driven Architecture	30
Disadvantages of an Event-Driven Architecture	31
Solution Considerations	31
Batch Architecture	32
Introduction	32
Tooling Examples	32
Advantages of Batch Architecture	33
Disadvantages of Batch Architecture	33
Solution Considerations	33
Data Integration vs. Application Integration When Considering Batch Processes	33

Business to Business Integration (B2B)	34
Introduction	34
Advantages of Using iPaaS for B2B and Healthcare	34
Disadvantages of Using iPaaS for B2B and Healthcare	35
Solution Considerations for iPaaS B2B Capabilities	35
Solution Considerations for iPaaS Healthcare Support Capabilities	36
<b>G. A Case Study—Fusion Applications Social Network Integration</b>	<b>36</b>
Case Study Overview	36
Project Introduction	36
Problem Statement	37
Objectives	37
Methodology	38
Solution Overview	39
Relationship With the Whitepaper Trends	39
In Summary	40
<b>H. Conclusion</b>	<b>41</b>

## List of Figures and Tables

Figure 1 - The progress of integration	8
Table 1 - Build or Buy Analysis for Prebuilt Integrations	18
Figure 2 - The progress of AI as it develops	20
Figure 3 - Microservice Architecture vs. Monolithic Architecture	23
Table 2 - Low-code vs. Middleware analysis	28
Figure 4 - Pub/Sub diagram	29
Figure 5 - Streaming diagram	30
Figure 6 - Batch architecture diagram	32

## A. Overview

This paper is meant to provide a broad, 10,000-foot view of the world of integration in 2024 and beyond. We'll be exploring some trends in the industry as well as our view of the current trajectories in integration technology. We include a contemporary use case study to help explore some of the decisions and choices that can be made to accomplish a good integration solution today.

### The Structure of This Document

This document is structured to present a summary on integration technologies, a brief history of integration, with some look forward to new developments, a slightly deeper dive on our specialist integration pillars, then some trends within the integration world and finally a deeper dive on a contemporary integration use case.

### About the Authors

We are the A-Team, an Oracle SaaS expert knowledge team, specializing in integration technologies over the past 20 years or so. As Oracle veteran consultants, we obviously understand and acknowledge our biases regarding the industry. However, we hope this document will prove of some utility even if you are not interested in Oracle products directly.

## B. Three Main Pillars of Integration

As integration has evolved, the industry has, *in general*, broken it down into three core pillars: **Application Integration**, **Data Integration**, and **Process Integration**. They are related and often work together, and products can often address integrations in two or all three pillars, but typically, each serves a different primary purpose. There are also other areas that are sometimes labeled independently, though we usually consider them as part of one of the primary pillars here.

### Application Integration

Application Integration entails linking various software applications, services, or systems to facilitate smooth collaboration. Within the realm of cloud computing, this process encompasses the integration of both SaaS applications and a combination of SaaS and on-premise applications, amongst many integration scenarios, including hybrid, multi-cloud, and traditional on-premise solutions.

#### Focus

The primary focus of Application Integration is to facilitate communication and interaction between disparate applications (cloud and on-premise), allowing them to share data, functionality, and processes.

#### Components

Application Integration typically involves middleware, iPaaS (Integration Platform as a Service), APIs (Application Programming Interfaces), message brokers, and other integration technologies to enable communication and data exchange between applications.

#### Objectives

The main objective of Application Integration is to streamline business processes, improve operational efficiency, enhance collaboration, and provide a unified user experience across multiple applications.

#### Examples

Examples of Application Integration include integrating CRM (Customer Relationship Management) systems with ERP (Enterprise Resource Planning) systems, connecting e-commerce platforms with inventory management systems, or linking a website with a payment gateway.

## Data Integration

Data Integration involves combining data from different sources or formats to provide users with a unified and consistent view of the data.

### Focus

The primary focus of Data Integration is on harmonizing data from disparate sources, ensuring its consistency, accuracy, and accessibility for reporting, analysis, and decision-making.

### Components

Data Integration encompasses processes such as data extraction, transformation, and loading (ETL), data replication, synchronization, and virtualization to integrate and manage data from various sources. These things can be achieved using tools such as ETL tools and iPaaS.

### Objectives

The main objective of Data Integration is to provide a unified, accurate, and consistent view of data across the organization, enabling informed decision-making, better analytics, and reporting.

### Examples

Examples of Data Integration include integrating data from multiple databases or data warehouses, consolidating customer data from different sources into a single view, or aggregating sales data from various retail stores.

## Process Integration

Process Integration involves orchestrating and automating typically human-driven business processes across different systems or applications to achieve seamless end-to-end workflow execution, incorporating both human and automated elements.

### Focus

The primary focus of Process Integration revolves around the seamless integration and enhancement of business workflows, prioritizing streamlined execution and reducing the need for manual intervention. Nonetheless, Process Integration primarily targets workflows that involve human participation, thereby often leading to the management of long-running processes.

### Components

Process Integration often involves workflow automation tools, BPM (Business Process Management) systems, or integration platforms to model, automate, and monitor business processes.

### Objectives

The main objective of Process Integration is to improve operational efficiency, reduce errors, enhance collaboration, and accelerate time-to-market by automating and optimizing business processes.

### Examples

Examples of Process Integration include filling out loan applications that involve approvals, automating order fulfillment processes spanning multiple systems, integrating customer onboarding workflows across various departments, or synchronizing inventory management processes with supply chain partners.

## Other Areas: RPA—Robotic Process Integration

Robotic Process Integration is at the “join” between Application Integration and Process Integration. The goal is to automate legacy systems that are typically intended to be human-operated, and APIs are not feasible. For us, it is generally considered to be part of the Process Integration pillar.

## The Three Pillars and Our Focus

To summarize, while Application Integration focuses on connecting disparate applications, Data Integration focuses on harmonizing data from different sources, and Process Integration focuses on orchestrating and automating business processes across systems. Together, these integration approaches play a crucial role in enabling organizations to operate more efficiently, improve collaboration, and adapt to changing business needs. The evolving landscape sees vendors and companies approaching these integration pillars in a very individual fashion, blurring the lines between them. Frequently, the concepts and products associated with these pillars are interchangeable, underscoring the interconnection and complexity of modern integration.

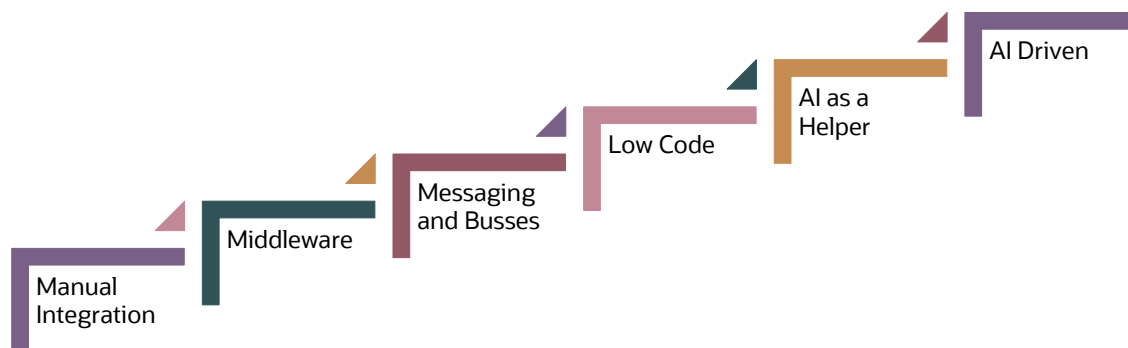
The focus of this whitepaper is trends, concepts, vision and use-cases in **Application Integration** and **Process Integration**. A deep dive into Data Integration is a broad topic of its own and a different white paper.

## C. The History of Integration and the Trajectory in 2024

### History

Integration as a software domain has followed a clear trajectory over the past 30 years. Starting in the Data Integration world, early Data Integration systems sought to try and merge multiple emerging data sources into a cohesive unit using strategies such as ETL (extract, transform, load). Subsequently, as interconnectedness increased, we saw the rise of application and Process Integration, with APIs being developed explicitly to facilitate inter-operation between applications at the application layer rather than the data layer. The trajectory has been clear, and new technologies continue pushing the integration domain forward.

Figure 1 - The progress of integration



### Manual Integration

Manual integration is the oldest approach to integration. Bespoke point-to-point integration solutions have existed since two applications wanted to exchange data and have been implemented in a variety of programming languages. These early integration solutions were typically more focused on Data Integration than true Application Integration.

Modern manual integration has changed dramatically with the rise of cloud-native and cloud-first trends in the wider software development industry. Manually built integrations can leverage and follow all the best practices in wider software development and, as such, usually leverage either containerization platforms such as Kubernetes to create container-based deployments or serverless infrastructures to provide fully cloud-native serverless implementations. These technologies can be triggered automatically on demand or as part of a wider infrastructure as a service solution, so they can be as “online” and responsive as any other Application Integration solution.



## Middleware

Application Integration was truly born with the rise of middleware platforms able to handle the nascent internet technologies of 25 years ago such as SOAP. Platforms such as Oracle's SOA Suite brought together many approaches to Application Integration under a single middleware server, able to handle the demands of the emerging pillars of Application Integration, as demonstrated in this excerpt from marketing for Oracle SOA Suite some 15 years ago:

### **Modularity**

Enable organizations to develop and manage services and composite applications in a modular fashion. This modularity supports reusability of components, making it easier and more cost-effective to design, deploy, and maintain applications over time.

### **Standards-based**

Industry-standard protocols and technologies ensure interoperability between disparate systems and applications. This approach reduces vendor lock-in and facilitates easier integration with existing IT infrastructure, regardless of the technology or platforms in use.

### **Efficiency and Scalability**

Designed to improve the efficiency of IT operations and the scalability of IT systems, middleware supports high-performance execution of services and composite applications, enabling organizations to handle growing volumes of transactions and data with minimal impact on system performance.

### **Management and Security**

Including comprehensive management and security features to monitor, manage, and secure services and composite applications. This includes capabilities for service governance, lifecycle management, monitoring, and security policy enforcement, helping ensure that deployments remain robust, secure, and compliant with regulatory requirements.

Middleware is still highly relevant in 2024—most cloud-based technologies are based on an integration middleware platform of some kind. Additionally, you can easily deploy and scale a middleware solution onto modern cloud architectures such as Kubernetes if you need to access their capabilities. If you don't want or need to control the middleware platform for yourself, most cloud vendors offer an integration platform that is based on a middleware software solution hosted directly from their cloud.

## Messaging and Busses

The rise of middleware facilitated the consolidation of much integration into singular software platforms. This saw the rise of “point-to-point” integration, then the event bus came along, and everyone jumped aboard. These days we see a hybridization of many approaches—buses based on cloud streaming platforms, point-to-point and hub-and-spoke “fan-out” styles.

## Low Code

With the rise of middleware, there was a strong desire to push towards more accessible solutions for developing the integrations themselves. Low-code solutions (long ago, some called these and similar low code platforms “4th generation programming languages”) have often been mixed into an overall middleware platform solution—drag-and-drop development accessible to so-called “citizen developers,” such as business analysts have been the mainstay of integration development for many years now. Most cloud-based integration solutions offer a browser-based integration development environment that is based on low-code principles. We explore some of the capabilities of low-code solutions in the section *No-Code and Low-Code Development Environments* below.

## AI as a Helper

AI helpers, or “co-pilots,” are a newly emerging field in software development, where the developer leverages some kind of domain-specific large language AI model to generate software development artifacts in some fashion.

The problem of data mapping has been persistent throughout the history of integration development. It is easy for a person to understand and recognize that “DOB” and “Date of Birth” are matching fields and should be mapped to one another. It is much more challenging for development environments to handle this automatically during development.

With large business objects with multiple nested hierarchies, the challenge gets complex and requires a lot of both domain knowledge and skill to correctly implement, with potentially disastrous consequences when a critical field is missed, for example. Arguably, most integration in the modern age really boils down to managing these complex object mappings between disparate products and domains. Automation has long been available, but it has typically used simple string pattern matching, which is weak and error prone. AI helpers are starting to emerge that are using the power of tools such as Large Language Models to try and interpret and generate mappings and other integration artifacts using AI trained on domain specific knowledge.

These have the potential to dramatically simplify the process of developing a complex integration.

## AI Driven

The future of most software development seems to be that the software should ultimately be able to “develop itself”—what this looks like is yet to be seen, but there are many future speculations on the evolution of AI tools and co-pilots to get to the state that you don’t really “develop” software anymore, you would have a conversation with an AI, and it would end up outputting a software artifact complete and ready to go.

## D. Application Integration—A Deeper Dive

Application Integration refers to connecting and automating interactions between stand-alone components in a larger application system. In contrast to business Process Integration, the systems being integrated are typically computer applications or similar components.

## Approaches to Application Integration

### Integration Platform (iPaaS)

Integration Platform as a Service (iPaaS) is a fully featured integration platform suite. It provides several key features to allow for comprehensive integration solutions. Much of this document will discuss these features and how modern integration solutions can leverage them.

#### Key components of iPaaS

##### Cloud-Based Service

The migration to the cloud has seen most iPaaS services migrate to a cloud-based service—historically, you would run a service on premises, but the advantages of cloud-hosted iPaaS are significant and have, for the most part, supplanted on-premise solutions.

##### Standards-Based

iPaaS is typically based on industry standards such as REST and SOAP. BPEL was historically a standard for implementation. However, the specification for BPEL has stagnated, and that is no longer a common offering.

##### Connectivity and Adapters

iPaaS platforms will typically provide a wide gamut of connectivity options, allowing for connections to a wide variety of backend services in a standardized manner, with transformation capabilities built-in and resilience

to backend issues as part of the standard connectivity. Often, these connectivity options will be available via an on-premise proxy agent as well, allowing for iPaaS to reach into an on-premise application without having to expose the on-premise application to the internet or punch through a VPN or other network connection.

### **Auditability and Traceability**

iPaaS platforms will offer comprehensive audit and tracing systems, allowing the efficient examination in detail of how integrations are performing and what is doing what.

### **Control Plane and the Runtime**

A key feature of almost all modern iPaaS is some sort of **control plane**. Control planes are usually the core server component of an application or even a Process Integration platform. They allow for control of the behavior of a system at **runtime** through various UI or API interactions. A well-featured control plane should have:

- A properly secured UI for access.
- A properly secured API for controlling the runtime.
- Means to query the current state of integrations.
- Means to view the recent or even long-term history of an integration instance.

### **Scaling, Elasticity, and Resilience**

A control plane and its associated runtime are usually the drivers of integration. To deal with a varying flow of demand, **elasticity** is typically a requirement of the control plane. Elasticity is the ability of the control plane to change its size, to cope with the current load imposed upon it. There are varying strategies for changing the size of a control plane:

- Using standard cloud scaling mechanisms:
  - Deploying an extra compute instance to accommodate the new load and removing it when the load requirement is gone.
  - Deploying additional CPU and memory capacity to accommodate.
- Migrating to a new instance with the new capacity requirement.

The size change, ideally, would be driven automatically by the control plane, based on parameters of the current load, however, this could have side effects like additional costs, which means that manual scaling might be more desirable—either by manually over allocating capacity to avoid “ramp-up” times, or by under allocating to save costs at the price of performance or potentially integration failures. It should be noted that automated ramp-up is typically accompanied by an automatic ramp-down—to preserve resources when the load decreases.

### **Disaster Recovery**

Typically, disaster recovery for iPaaS would be the capability for the integration platform to persist when a primary hosting site goes offline for whatever reason (power failure of the data center, natural disaster, etc.). This kind of capability is desirable to be built into the control plane and platform to allow all the integration systems to fail over to an alternative site, either fully automatically or via a manual process as part of a wider **business continuity plan**. There are usually business factors in play - an integration platform is only as operational as the targets it is integrating, so disaster recovery, e.g., a data center power outage, might involve failing over not just the integration platform but also all the integrated services. You should consider control plane disaster recovery in the wider context of a **business continuity** or **business recovery plan**.

## Upcoming Developments in iPaaS and Application Integration

### AI Helpers

AI helpers are rapidly becoming part of Application Integration. For more details, see the *AI and Machine Learning* trend discussion below.

### Oracle Integration Cloud

Oracle Integration Cloud (OIC) is Oracle's primary product offering in the iPaaS Application Integration pillar. It offers many of the key components listed above and is considered a strong offering in the integration domain by independent industry experts.

### Build It Yourself

For more advanced development teams, Application Integration can be accomplished by using standard development tools and practices, often leveraging flexible cloud-based deployment solutions, such as serverless or containerization. This is obviously a much more complex proposition for most, but it can fit in better with a wider overall development team.

### Serverless Platforms

Serverless is a somewhat emerging development paradigm whereby a developer, by adhering to a simple API, can create a code deployment artifact that is deployed into a cloud-hosted infrastructure elsewhere and triggered on demand. It gets its name from the fact that no server is apparent, though it can be considered to behave as if it is deployed on a more traditional server-hosted solution. Typically, serverless infrastructure involves the backend creation of a container specification, along with some rules that cause the creation of the container and triggering of the contained code, such as when an HTTP call is received matching a URI format for example. (There are many ways a serverless container can be triggered, but the most common by far is the HTTP trigger).

### Serverless and Integration

An integration driven by some kind of triggering action, such as an API call or an Event, could easily be implemented as a serverless “function” of some kind, whereby the implementation invokes whatever associated integrated services in response. This approach is surprisingly accessible and meets many of the standard needs of an iPaaS platform with little of the associated overhead. It obviously requires developer skills and ongoing maintenance, which can be a definite drawback.

### Containerization

Containerization has been a standard cloud deployment model for some years now, with Kubernetes becoming the de facto standard in containerized deployment infrastructure. If you want ultimate control over everything, packaging your custom-built custom code into a Kubernetes cluster is unparalleled.

### Containers and Integration

There is little specific about containers and integration. Many legacy on-premise integration platforms can be deployed into a Kubernetes cluster, providing an unorthodox path from on-premises to the cloud if that is required. Alternatively, your cutting-edge AI-generated integration code can be packaged and deployed into Kubernetes as well. There really are no limits when it comes to containers.

## Advantages of Build It Yourself Integration Solutions

### Customization

There is unparalleled customization from building it yourself. Every aspect can be controlled to an exacting degree.

### Performance

It is unlikely that any iPaaS platform will match the performance of a bespoke point-to-point integration built in Rust, C++, or another high-performance language.

## Disadvantages of Build It Yourself Integration Solutions

There are many, many disadvantages to build-it-yourself solutions:

### Development Costs

The bespoke Rust point-to-point integration will require a highly skilled developer familiar with the domain specifics of all the integration endpoints, which is unlikely to be low-cost.

### Maintenance

The bespoke Rust implementation will need to be maintained because all services change over time, and your integrations must accommodate those changes in some way.

### All the Usual Stuff You Get for “Free” With iPaaS

Scalability, auditing, disaster recovery, and elasticity will all need to be considered, configured, and developed.

## E. Process Integration—A Deeper Dive

Application Integration refers to connecting and automating interactions between stand-alone components in a larger application system. When one of these stand-alone components is human, a different type of software product is typically needed to support the use case. This type of product has typically been called Business process management (BPM), Human Workflow, or Business Process Automation. We use the term **Application Integration** for integration scenarios that only include system-to-system. We use the term **Process Integration** or **business Process Integration** for integration scenarios that include both systems and human beings.

Enterprises need business Process Integration to automate and streamline human-related business operations. Business process use cases are prevalent in all aspects of any enterprise. Process products have been used as a stand-alone process platform and as extensions to enterprise applications.

A process platform also allows processes to interact with external systems. So, it bears many similar features to integration in supporting system-system integrations. The main capability of a process platform, of course, is system-human interaction. Process platform products usually offer basic features such as a low code development environment in a web-based IDE and a standards-based process construct (e.g., BPMN). Like any other application platform, process platforms have been evolving through the years based on new customer demands. The following are some of the more advanced capabilities that are often key to a successful process implementation.

## Key Components in Process Integration

### Forms

HTML-based forms are the main user interface for interacting with process instances. Complex business processes often involve complex data structures such as nested arrays. Process forms should be constructed to represent a complex process data structure. In addition, the data set can be large, for ease of user, forms should support dynamic interactive behavior such as conditionally hiding or displaying data depending on user entries.

### Form Rendering

Form rendering typically refers to the capability that a single definition of a form can be viewed differently in different communication channels and devices, such as in computer browsers, email, mobile devices, and messaging applications. For completed manual tasks in a process, form rendering may include the capability of printing completed forms into PDF or other formats. This is often a requirement for record-keeping and non-repudiation.

### **Process UI Page Integration**

Business users prefer a single-entry portal to multiple disjointed applications for conducting their daily business activities. Process UI integration allows process workspace pages to be integrated into other web pages, allowing business users to work with multiple applications more coherently.

### **Customizable Calendar**

Calendar is an important feature in business processes. A customizable calendar allows businesses to define their work calendar. It also allows individual business users to create their work and time-off schedules. These capabilities in a calendar are essential to the flexibility of task assignments.

### **Hierarchical Organizational Structure**

A clearly defined organization hierarchy enables a process platform to create more sophisticated task assigns, reassignments, and escalations based on every business' unique organizational structure.

### **Human Task Life-Cycle Management**

Human task life cycle management encompasses task creation, assignment, reassignment, update, cancellation, and completion. Role-based task assignment is a basic feature in a process platform. Complex business processes often require more comprehensive features. More advanced features include hierarchical task approval and escalation based on organizational structure, group task assignment, and group task completion.

### **APIs for Managing Human Tasks**

The availability of APIs for managing human tasks allows process capabilities to be integrated into other applications. It also allows developers to extend a process platform by developing additional process functionality they require. Some examples of such extensions may include bulk task approval and custom task search capability.

### **Batch Operations on Human Tasks**

Batch operations on human tasks allow a business user to select multiple assigned tasks and act on them at once rather than individually. The feature greatly improves efficiency in working with a large number of processes.

### **Structured and Unstructured Processes**

In a structured business process, steps and transitions between steps are clearly defined. However, in some scenarios, transitions between steps are more complex and may require a large number of conditions to determine the next step. Using a structured process construct to implement complex transition rules can lead to an unmanageable process application. In these scenarios, unstructured processes may help reduce the complexity. An unstructured process leverages the concept of stages to group one or more tasks. Rules can be defined to govern the transitions from one stage to the next. Milestones can be added to unstructured processes to indicate the progress of process instances. Unstructured processes can simplify complex processes and improve business process visibility.

### **Business Rule Services**

Business rule services allow a process platform to externalize complex conditional tests and, thus, improve process modularity and manageability.

### **Alter Flow by Manual Intervention**

Manual alter flow allows a user to manually change the flow of the running process instances. Such alteration may include skipping the following steps or going back to a previous step. This feature can be essential in managing business process exceptions.



### Process Versioning

Process versioning allows multiple process versions to be active in the same runtime environment. Business processes can be long-running. The process versioning feature gives process administrators the ability to create business process upgrade strategies. It ensures minimum interruption to existing business process instances due to upgrades.

### Process Correlation

Process correlation is a technology feature that allows developers to create decoupled processes and enable decoupled processes to have a one-on-one message exchange. This feature can simplify process implementation in many use cases.

### Analytics for Process Visibility

Collectively, business process instances embed some critical information about business operations. Providing visibility into the business process instances is key to the success of large process implementations. Out-of-the-box analytics support of a process platform includes basic UI for high-level process view in pre-defined charts and tables, and the ability to search for process instances based on process instance metadata as well as payload data. Equally important for a process platform is to be able to export a customizable set of process instance data into external systems so that they can be processed by other applications.

### External Service Integrations

Integrating with external services extends a process platform's reach into new areas. RPA (Robotic Process Automation) has been commonly used to automate existing manual UI-based applications. The data that an RPA tool extracts from legacy applications can be used to trigger business processes. In turn, business processes can feed data into legacy applications via RPA. Combined with the emergent AI-based services such as document processing services, a process platform can automate a wide range of manual tasks previously not possible.

## Upcoming Developments in Process Integration

### AI helpers

The evolution of AI helpers is changing all facets of integration and Process Integration is no exception. For more information see the *AI and Machine Learning* trend discussion below.

### Ad-hoc Human Tasks

This is an emerging feature in process platforms. Ad-hoc human tasks are created via an API. They are stand-alone and live outside of a process. Though not widely used currently, this feature should extend the reach of human tasks into new use cases.

## Oracle Process Automation Cloud (OPA)

OPA is Oracle's current offering in business process automation. It supports most of the features mentioned above and is adding new features in every future release. In addition, OPA supports out-of-the-box integration with OIC. Many Oracle SaaS modules, such as HCM, provide out-of-the-box integration with OPA as an extension mechanism for enterprise applications. These features make OPA a key differentiator from other process platforms.

## F. Trends and Themes in Integration

There are several common trends and themes across modern integration solutions. Some are part of the ongoing trajectory of integration as a domain, others are interesting new facets of integration, but they generally fall into two *arenas*: *Deployment Trends* and *Development Trends*. Deployment trends are focused more on how to take your integration and "deploy" it to become a production service available for use. Development trends are more focused

on the ways you develop your integration, from how the code is developed to paradigms in software development itself.

## Deployment Trends

Hosting, deploying, and generally running your integration has many facets. Here, we will review two integration deployment-focused trends.

### Prebuilt Integrations

### Hybrid and Multi-Cloud Integrations

---

## Prebuilt Integrations

### Introduction

In the ever-evolving landscape of digital business, where organizations embrace a multitude of diverse and specialized applications to address their unique business needs, managing the seamless communication and data flow between these applications has become imperative. This section delves into the strategic concept of **prebuilt** integrations.

Enterprises find themselves relying on an array of applications to streamline operations, manage data, and enhance business efficiency. However, the inherent diversity of these applications often results in siloing of data and fragmentation of workflows. To address this challenge, a robust integration architecture is essential, acting as the bridge to span these gaps and create a cohesive IT ecosystem.

At the forefront of addressing these integration needs is the concept of ready-to-use integrations, commonly known as **prebuilt** integrations. This encompasses readily available integrations that can be implemented without the need for extensive customization or development effort. It includes both prebuilt integration solutions and prebuilt API connectors, with the integration process typically involving a drag-and-drop low-code implementation for predefined processes.

### Benefits

Enterprises benefit from **prebuilt** integration solutions, pre-packaged integration frameworks meticulously designed to handle a standard integration task specific to an industry or use case. These solutions come equipped with built-in features and functionality directly from the vendor, empowering organizations to integrate systems and applications rapidly and seamlessly without the burden of extensive development, configuration, or customization. Upon implementation, these solutions stand ready for immediate utilization, providing organizations with on-the-fly integration capabilities to address common business needs. A fully featured Integration Platform as a Service (iPaaS) platform should offer a vast library of **prebuilt** integrations tailored for standard use cases across various business applications.

## Features in a Modern iPaaS Platform for Prebuilt Support

### Marketplace

A key capability of iPaaS platforms is to provide a marketplace for publishers and consumers. A marketplace provides a means to connect potential consumers of prebuilt integrations with the developers and publishers who are building them.

### Monetization

Another key capability is the ability for publishers and developers to monetize the artifacts they produce. Developing and publishing prebuilt integrations takes time and effort, so there needs to be a means to



reward those who create prebuilt integrations. A platform should have capabilities to help creators monetize the assets they publish to the marketplace.

### **Template-Based Integration Workflows**

iPaaS platforms enable organizations to leverage prebuilt integration templates or workflows. These templates encapsulate best practices and proven integration patterns, empowering businesses to kickstart integration projects quickly and efficiently.

### **Flexibility and Customization**

While iPaaS platforms can provide template-based integration workflows leading to prebuilt integrations, they should also provide capabilities to customize these prebuilt integrations for organizations to align with their specific requirements. A prebuilt is unlikely to cover 100% of a use case.

## **Advantages of Prebuilt Integrations**

### **Accelerated Implementation**

By leveraging prebuilt integration components, organizations can significantly reduce the time and effort required for integration projects. This enables faster deployment of new applications and features. Prebuilt integrations enable organizations to achieve faster deployment of their required integrations. With pre-configured settings and connectors, businesses can save valuable time and effort that would otherwise be spent on developing and implementing complex integration solutions from scratch.

### **Cost-Effective Solutions**

Prebuilt integrations minimize the need for custom development, leading to cost savings in terms of both time and resources. Organizations can allocate their budget more efficiently, focusing on innovation rather than repetitive integration tasks.

### **Improved Agility**

The modular nature of prebuilt integrations allows organizations to quickly adapt to changing business requirements. New applications can be seamlessly integrated without disrupting existing workflows, enhancing overall business agility.

### **Reduced Maintenance Overheads**

As prebuilt integrations are standardized and tested, ongoing maintenance becomes more straightforward. Updates and changes to integrated applications can be managed more efficiently, reducing the risk of downtime and operational disruptions.

## **Disadvantages of Prebuilt Integrations**

### **Missing Customization Options**

Prebuilt integrations need to provide all the customizations you require for your integration scenario. If some critical customization is missing, avoiding the prebuilt entirely may make the implementation significantly easier. However, this is often a difficult assessment to make during a tight development deadline.

## Build or Buy Analysis for Prebuilt Integrations

Table 1 - Build or Buy Analysis for Prebuilt Integrations

	Prebuilt Integration	Custom Integration
<b>Cost</b>	<ul style="list-style-type: none"> <li>Lower costs</li> </ul>	<ul style="list-style-type: none"> <li>Higher Costs</li> </ul>
<b>Resources</b>	<ul style="list-style-type: none"> <li>Lower resource requirement</li> </ul>	<ul style="list-style-type: none"> <li>Higher resources requirement</li> </ul>
<b>Timelines</b>	<ul style="list-style-type: none"> <li>Faster Implementation and time to delivery</li> </ul>	<ul style="list-style-type: none"> <li>Slower implementation and time to delivery</li> </ul>
<b>Maintenance</b>	<ul style="list-style-type: none"> <li>Easy to maintain and upgrade</li> <li>Integrations are managed by Vendors so less need of dedicated teams</li> </ul>	<ul style="list-style-type: none"> <li>Manual updates requiring time, money and resources</li> <li>Integrations managed by customer</li> </ul>

## Hybrid and Multi-Cloud Integrations

### Introduction

Modern businesses find themselves invested in both on-premises and cloud, requiring iPaaS solutions that can seamlessly bridge diverse ecosystems. Additionally, vendor lock-in is still widely considered an anti-pattern in business, so iPaaS systems should be able to connect between cloud providers. Therefore, **hybrid and multi-cloud** connectivity with iPaaS has become a pivotal strategy for organizations aiming to optimize their IT infrastructure, enhance flexibility, and streamline business processes. For this, the iPaaS systems must possess the agility to connect with applications dispersed across the digital realm. This section delves into the challenges, capabilities, and benefits of iPaaS **hybrid and multi-cloud** strategy.

### Hybrid Cloud and Multi-Cloud Environments

#### Hybrid Cloud

Refers to the combination of on-premises infrastructure with cloud services. It allows organizations to leverage the benefits of both private and public clouds while maintaining control over sensitive data.

#### Multi-Cloud

Involves the use of multiple cloud service providers to meet specific business needs. This strategy mitigates vendor lock-in, enhances redundancy, and provides access to a broader range of services.

### Challenges

There are several key challenges in providing **hybrid and multi-cloud** solutions for both product vendors and businesses requiring this capability.

#### Diverse Ecosystems

Organizations often face challenges in integrating diverse applications, systems, and data sources spread across hybrid and multi-cloud environments.

#### Data Silos

Siloed data in different clouds can hinder collaboration and decision-making processes.

#### Security and Compliance

Ensuring secure and compliant data transfer between on-premises and cloud environments is a top concern.

## Capabilities

An Application Integration offering that supports **hybrid and multi-cloud** requires several key capabilities; otherwise, an effective implementation is unlikely.

### Connectors and Adapters

Connectors, also known as Adapters, are specialized software components provided by the iPaaS system that provide a predefined interface to interact with a specific application or service. They encapsulate the intricacies of API calls and data exchange to a remote service, either a cloud service or other application, simplifying the development of integrations.

IPaaS systems typically have a broad suite of connectors or adapters for many different cloud systems, from databases to SaaS applications to social platforms and more.

### Prebuilts

Prebuilt integrations can simplify the connection between different applications and cloud vendors but are not a *necessary* requirement of **hybrid and multi-cloud** integration.

They are particularly useful for commonly used software applications, such as popular CRMs, ERPs, or other SaaS cloud services, where prebuilt connectors are readily available. For more refer to the *Prebuilt Integrations* trend discussion above.

### On-Premise Connectivity Without Network Configuration Changes

IPaaS systems need to have capabilities to connect to on-premise applications and data. This needs to be accomplished securely without requiring the customer to change their network configuration to allow the cloud to access the on-premise components or expose the on-premise components to the Internet in any typical fashion.

The details of how this is accomplished are typically vendor-specific, but hybrid cloud is typically very challenging without such a capability.

### Security

As iPaaS facilitates connections with various cloud vendors, it is crucial for its security features to seamlessly integrate with diverse security mechanisms. Additionally, the iPaaS system should support functionalities such as token exchanges, identity propagation, OAuth, and similar mechanisms.

## Benefits

The benefits of using an iPaaS platform to deliver a **hybrid or multi-cloud** solution are several. It can be worth the additional development effort.

### Connectivity

iPaaS acts as a middleware, providing connectors, adapters and prebuilt integrations that simplify the connectivity between diverse systems and applications.

### Flexibility

iPaaS allows organizations to adapt to changing business requirements by providing a scalable and flexible integration solution.

### Automation

iPaaS automates workflows and data flows, reducing manual intervention and improving operational efficiency. IPaaS in multi-cloud Allows a process orchestration to be executed anywhere. One step can be performed on-premise and another in the cloud even if they are part of the same orchestration.

### Single View

A hybrid or multi-cloud iPaaS should offer a single visual view of a process that may be distributed across several clouds and on-premise platforms, reducing the management complexity and improving the audit trail presentation of the process in a multi-cloud environment.

## Development Trends

Developing your integration solution is critical to the success of an integration project. The means to develop integrations are manyfold. Here, we will look at some existing and emerging trends in integration.

**AI and Machine Learning**

**API Driven Integration Development**

**No-Code and Low-Code Development Environments**

**Event-Driven Architecture**

**Batch Architecture**

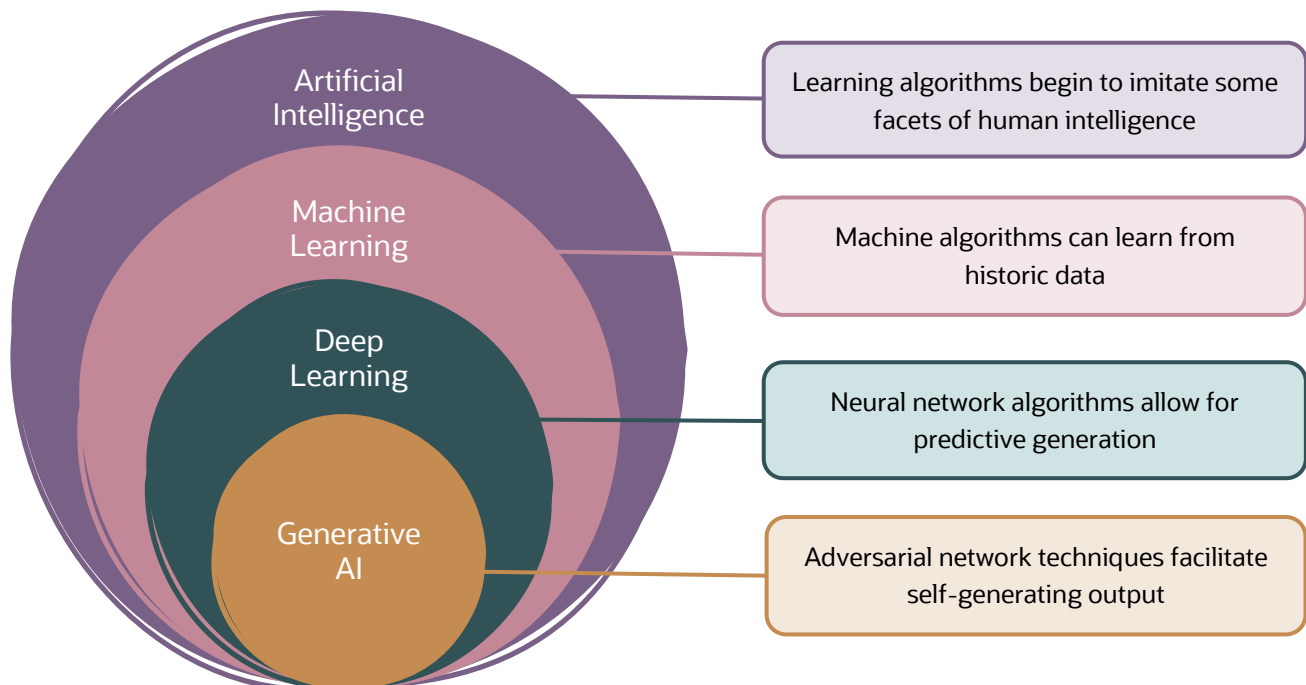
**Business to Business Integration (B2B)**

## AI and Machine Learning

### Introduction

Artificial Intelligence (AI) is an emerging technology across the development industry. It uses advanced compute models to try and replicate or even improve upon human intelligence in some fashion. It has become rapidly adopted across the software industry to accelerate many facets of software development. It is especially potent at enhancing efficiency and decision-making in the presence of significant data volumes. It is being rapidly adopted across diverse

Figure 2 - The progress of AI as it develops



industries like manufacturing, retail, customer service, transportation, agriculture, and supply chain. The prevalence of AI is expected to grow further, bringing transformative change to the industry.

Generative AI is a novel emerging capability, able to create new content by extrapolating from its training data. This technology enables organizations to summarize documents, construct tables, generate meaningful text, produce code, and synthesize ideas, offering a new level of creative and productive capabilities.

## **AI Assistants and Generative AI**

Much effort is being focused on incorporating AI assistance into software products like iPaaS to assist in the development process. Future developments include the capability to generate whole integrations from simple descriptive statements without coding.

### **AI and ML-Assisted**

Integration platforms started incorporating AI and ML to assist developers in the design and implementation process. This involved providing suggestions and guidance based on common patterns, reducing the manual effort required. Machine learning algorithms could analyze historical data and recommend optimal mappings, transformations, or error-handling strategies. For example, Dell Boomi Suggest recommends automatic mapping between source and target nodes with high, medium, and low confidence scores; we believe the OIC Recommendation Engine also does that. The AI assistant can guide developers to relevant documentation in response to their queries.

### **AI First**

The AI first approach takes integration platforms to the next level by designing, managing, and orchestrating integrations with learned intent. AI is not just an assistant but a key player in the integration development lifecycle.

Generative AI, LLMs, and agent frameworks can significantly contribute to an integration platform when adopting an AI first approach by bringing intelligence and autonomy to the integration process development process.

LLMs can enhance natural language understanding within the integration platforms. This can be utilized for interpreting and processing natural language queries, facilitating more user-friendly interaction with the integration platform.

An agent framework allows for the creation of modular AI agents that specialize in different aspects of integration, such as creating connections, data mapping, transformations, event handling, or error resolution. Each agent encapsulates specific AI capabilities, making it easier to integrate and update individual intelligent components within the broader integration process.

## **Some Tooling Examples Where AI is Being Used**

- OCI Data Science for development, deployment, and monitoring of machine and deep learning models.
- OCI Generative AI for hosting large language models.
- OCI Generative AI Agent service for question-and-answer related platform based on documents.
- Oracle Digital Assistant as a virtual assistant.

## **Advantages of Using AI Tooling**

- By leveraging AI capabilities, integration platforms can become more intelligent, adaptive, and user-friendly, ultimately streamlining the integration process and delivering enhanced value to organizations.
- AI and ML use cases for integration platforms can significantly improve efficiency, decision-making, and overall performance.

## Disadvantages of Using AI Tooling

### Extreme Cost

AI tooling requires a large amount of costly computing hardware. Currently, much of this cost is not visible to AI consumers as the market grows. This cost will eventually start getting passed on to consumers, and costs will greatly increase.

### Exposing privileged data

AI learning models work best when fed a specific data set. They tend to echo back their learning data in unexpected ways. A custom-trained language model may risk exposing privileged data to unexpected users.

## Solution Consideration

AI is being deployed to help with a lot of possible solution use cases. Some examples:

### Design Integration Flows

An AI assistant can be trained on numerous integration flows and data points. By giving input as “*Design Ship Confirm Integration from WMS to Fusion Inventory*,” an AI assistant could design an integration process by creating the project, connections, lookups, and integration code for a shipment confirmation flow. This is an AI first approach.

### Intelligent Data Mapping and Transformation

An AI assistant can suggest the data transformation mapping based on trained historical data during development and suggest if any attribute of an existing API is changed.

### Chatbots and Virtual Assistants

An AI assistant can answer questions about the integration platform by implementing *retrieval augmented generation*. Furthermore, it can also suggest solutions for known issues.

### Anomaly Detection

The integration platform can implement machine learning models to detect anomalies in activity streams. This can help identify unusual patterns or outliers, signaling potential issues that need attention.

### Real-time Analytics

The Integration platform can implement real-time analytics using ML models to gain insights into data as it flows through the integration platform. This can enable timely decision-making and actions based on current information.

### Automated Testing and Quality Assurance

Implement AI-powered testing frameworks that can automatically generate test cases, simulate real-world scenarios, and identify potential issues in integration workflows. This accelerates the testing process and ensures the robustness of integrations.

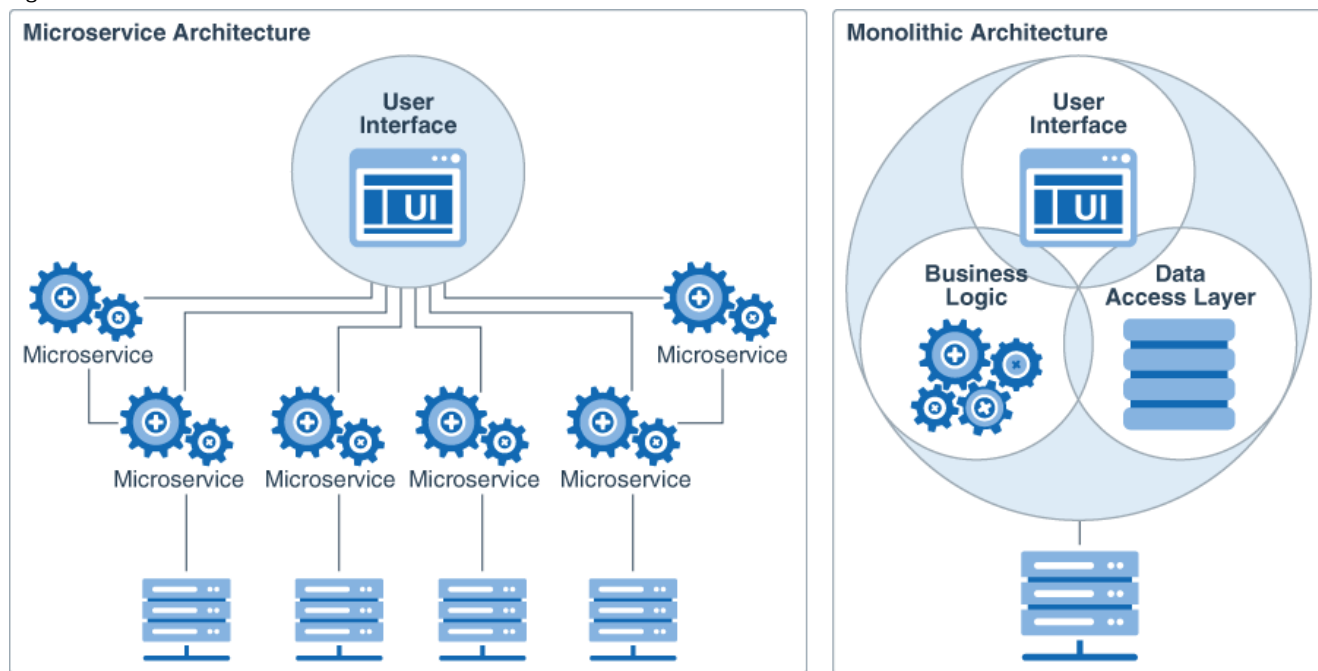
---

## API Driven Integration Development

### Introduction

API integration is the method of connecting two or more applications using APIs for exchanging data and performing actions. APIs (Application Programming Interfaces) are sets of protocols and standards that allow different software applications to communicate with each other. API-based integration allows applications to interact in real-time. They can help integrate applications and services using diverse technologies to effectively integrate, helping attain greater agility and flexibility.

Figure 3 - Microservice Architecture vs. Monolithic Architecture



APIs are essential to microservices-based application development. API-led integrations gained huge popularity due to the explosion of microservice-based architectures and the “cloud-native” development paradigm.

A microservice architecture is an approach to developing applications comprising small, independent services, each running in its own process and owning its own data. Services communicate with each other using lightweight mechanisms, typically REST APIs. REST APIs are designed to use HTTP protocol mechanisms and are very popular since they are simple to implement and require little specialized software.

### API First Development

API first development is a development approach focused on defining, developing, and testing APIs early in the development cycle. APIs are well-defined using API specifications (for example, Open API or Blueprint).

API governance practices, such as defining and applying standards and using API style guides, can help with the development and deployment lifecycle of API-based integrations.

### API Management and API Gateways

API management and gateway platforms provide tools for developing, designing, monitoring, testing, securing, and analyzing APIs for organizations. Features provided by API management platforms include:

- **Security**
- **Rate limiting**
- **Throttling**
- **Analytics, insights**
- **Monitoring**—API monitoring
- **Discovery**—a developer portal for API discovery



## API Monetization

API monetization refers to businesses charging developers or consumers for access to their APIs. This allows businesses to generate revenue using their business assets. At its core, a platform will require integrating with a billing provider to enable monetization.

The monetization models could be of various types, some of which are:

- **Subscription model**—monthly or annual recurring fee-based.
- **Pay-as-you-go model**—metered by usage volume.
- **Data monetization**—where consumers pay by access to unique datasets.
- **Transaction based**—per successful transaction - example payment gateways.
- The pricing could also be tiered based on usage limits, premium features, supportability, and other features provided.

## Advantages of API First Integration

### Stateless

Suitable for synchronous and one-way stateless integrations.

### Scalable

Independent services using stateless restful APIs allow for independent scaling.

### Flexible

Ability to add destroy and modify individual services in a highly available manner and ensuring zero downtime.

### Heterogeneous

Polyglot services using different technology stacks can be effectively integrated using API-based integration due to underlying common standards.

### Highly Available and Fault-Tolerant

API integration allows for load balancing between redundant instances of individual services. Any faults can be localized and resolved, hence minimizing the blast radius of failure.

### Independent Scaling

It is generally easier to provide high availability if services are containerized and can scale independently (for example, using a Kubernetes-based container engine).

### Possibly Lower Cost and Reusable

Services with defined API contracts can be reused by various applications, reducing duplication and, hence, development and maintenance costs.

### Gateway

Use of an API Gateway to front APIs for desired features such as rate limiting, throttling, monetizing, security, and simple transformations to be applied across an organization's APIs.

## Disadvantages of API First Integration

### State

APIs can be a challenge for modeling stateful workflows as APIs are inherently stateless.



**ACID**

ACID transactionality (an all-or-nothing guarantee in a sequence of API interactions) will require additional design or tooling support.

**Lack of Transactions**

Lack of transactionality risks data inconsistency between systems. These inconsistencies may require additional batch processes for reconciliation and to restore eventual consistency.

**Proliferation**

API proliferation among services can lead to multiple hops and increased latency.

**Troubleshooting**

Troubleshooting failures involves traversing through multiple systems' monitoring consoles. This can be quite complex for clustered services.

**Bulk Processing**

API integrations are rarely suitable for bulk transactions, where it is better to consider other patterns like batch integration.

## Solution Considerations

There are many facets that should go into considering API first integration approaches and development.

**Complex Monitoring**

Transactions or flows span multiple services, which could be polyglot or diverse in technologies.

**Complex Error Handling**

As the number of services or APIs increases, there will be a corresponding increase in points of failure, requiring more complex error handling.

**Context Tracking**

The integration platform provides end-to-end tracking using a common context (for example, ECID).

**Orchestration**

To ensure ACID transactions or SAGA-style eventual consistency, either the platform should provide a SAGA orchestrator or this must be designed by integration developers.

**Coupling**

Tight coupling between applications (in contrast to event-driven architectures).

**One-way**

Inherently suited for synchronous and one-way communication. True asynchronous style needs additional capabilities like events or callbacks.

**Automatic Publication and Documentation**

Ability to autogenerate and publish APIs for consumption.

**Control**

The ability for throttling, rate limiting, response caching, and load balancing to be considered during product choice.

**Monetization**

API monetization capability provided by the platform.

## No-Code and Low-Code Development Environments

### Introduction

Before the era of cloud computing, enterprises relied on a middleware integration solution for managing integrations among different on-premise applications, itself installed on-premise. Low-code solutions were often a feature of these integration platforms; however, the low-code development environment was typically a separately bundled client application, often only available on Windows.

On-premise middleware, which was limited to orchestrating on-premise applications, struggled to integrate cloud-based SaaS applications. With the rapid evolution of cloud computing, there was a growing need for integration solutions that could orchestrate on-premise systems with cloud-based applications.

This growing need led to the development of low-code browser and cloud-based integration platforms. These integration platforms are modern solutions designed to simplify and accelerate the process of creating integrations using prebuilt connectors and visual interfaces between on-premises systems and cloud applications. Furthermore, these platforms offer support for integrating services across multiple cloud providers. Low-code platforms provide a range of tools and features that allow users to create custom solutions using a drag-and-drop interface without requiring extensive coding knowledge.

### Capabilities of Modern iPaaS Platform to Support Low Code or Drag & Drop Features

#### Visual Development

These platforms offer visual development environments with drag-and-drop interfaces, enabling users to design integrations through graphical representations of workflows and data mappings. This visual approach reduces the need for extensive coding.

#### Pre-Built Connectors

Low-code integration platforms typically come with pre-built connectors and adapters for popular applications, databases, services, and technology. These connectors simplify the integration process by providing out-of-the-box connectivity, reducing the need for custom development.

#### Data Mapping and Transformation

Users can visually define data mappings and transformations between different systems. This capability allows for seamless translation of data formats and structures, ensuring compatibility between integrated applications.

#### Monitoring and Analytics

These platforms offer tools for monitoring integration performance, tracking data flow, and analyzing key metrics. Monitoring dashboards provide insights into the health and efficiency of integrated processes.

#### Continuous Integration and Deployment (CI/CD) Support

Many modern low-code platforms offer integration with CI/CD pipelines. This enables automated testing, version control, and deployment of integration solutions, ensuring a seamless and continuous integration process.

### Advantages of No-Code and Low-Code

#### Accelerated Development

These platforms' low-code nature significantly accelerates the development process, allowing users to create integrations with greater speed than traditional coding methods.

**Cost-Effectiveness**

Low-code platforms empower a broader range of users, including business analysts and citizen integrators, to participate in integration projects. This can lead to cost savings by reducing the need for hiring specialized developers and minimizing the time and resources required for integration projects.

**Flexibility and Adaptability**

The visual design approach allows for easy modifications and adaptations to integration workflows. Organizations can quickly respond to changing business requirements without lengthy development cycles.

**Consistency and Reusability**

Low-code platforms often include reusable components and templates, promoting consistency in integration design. This reduces redundancy and ensures that best practices are consistently applied.

**Easier Maintenance and Updates**

The visual nature of low-code platforms simplifies the understanding, maintenance, and updating of integrations. Changes can be made visually, reducing the complexity associated with traditional coding.

**Disadvantages of No-Code and Low-Code****Optimization**

In modern low-code systems, changing the approach to optimize performance is often difficult or impossible. For example, controlling the flow of a loop might be optimized by understanding the specifics of the data, but the low-code system generates in a singular way that is not performance-friendly.

**Customization**

In a low-code environment where everything is completely controlled for you, deeply customizing the flow is challenging or impossible.

**Incorporation Within a Wider Ecosystem**

Low-code and no-code environments generally offer an all-in-one experience, and incorporating an integration solution built in such an environment with a wider software ecosystem can prove challenging, as it can be difficult to create and recreate an artifact in coordination with the wider ecosystem.

**Lifecycle Management**

Progressing a no-code or low-code built solution through the phases of a typical software lifecycle (development, staging, production) can present a challenge, like incorporation in a wider ecosystem, above, for similar reasons—the creation of a portable or reproducible point-in-time artifact for progression can prove a challenge.

**Solution Considerations**

When considering no-code or low-code solutions the following are things to consider.

**Cost-effectiveness**

Evaluate the iPaaS solution's pricing model. Some platforms charge based on usage, while others have a subscription-based model. Consider your budget and choose a solution that offers the best value for your organization. Also, consider buy vs. build, as in whether it's cost-effective to build ground up using building blocks or to buy iPaaS products.

**Prebuilt Connectors**

Evaluate if the prebuilt connectors are helpful for your integration needs. If not, this is about the cost-effectiveness above.

Low-Code vs. Middleware Analysis

Table 2 - Low-code vs. Middleware analysis

Low-Code Platforms		Middleware
<b>Visual Development and Configuration</b>	These platforms provide visual development environments where users can design integrations using graphical interfaces. Configuration settings, data mappings, and process flows are created visually, reducing the need for manual coding.	Traditional middleware solutions often require manual coding and configuration through scripts or code-based configurations. This process can be more time-consuming and error-prone compared to visual development.
<b>Prebuilt Connectors and Templates</b>	These platforms come with prebuilt connectors and templates for common applications, databases, and services. These pre-configured components streamline the integration process and contribute to automation.	Middleware solutions may lack a comprehensive library of prebuilt connectors, requiring developers to manually configure connections and adapters for each integrated system.
<b>Deployment Process</b>	The user-friendly interfaces of these platforms simplify the deployment process. Users can deploy integrations with a few clicks, reducing the complexity and time required for deployment.	Middleware solutions may involve more complex deployment processes that require careful coordination and configuration.

Event-Driven Architecture

Introduction

Event-driven architecture (EDA) uses events to communicate between services. This is a common approach for asynchronous real-time or near-real-time integration between services in a decoupled fashion.

EDA is also widely used in modern applications using microservices for decoupled integration.

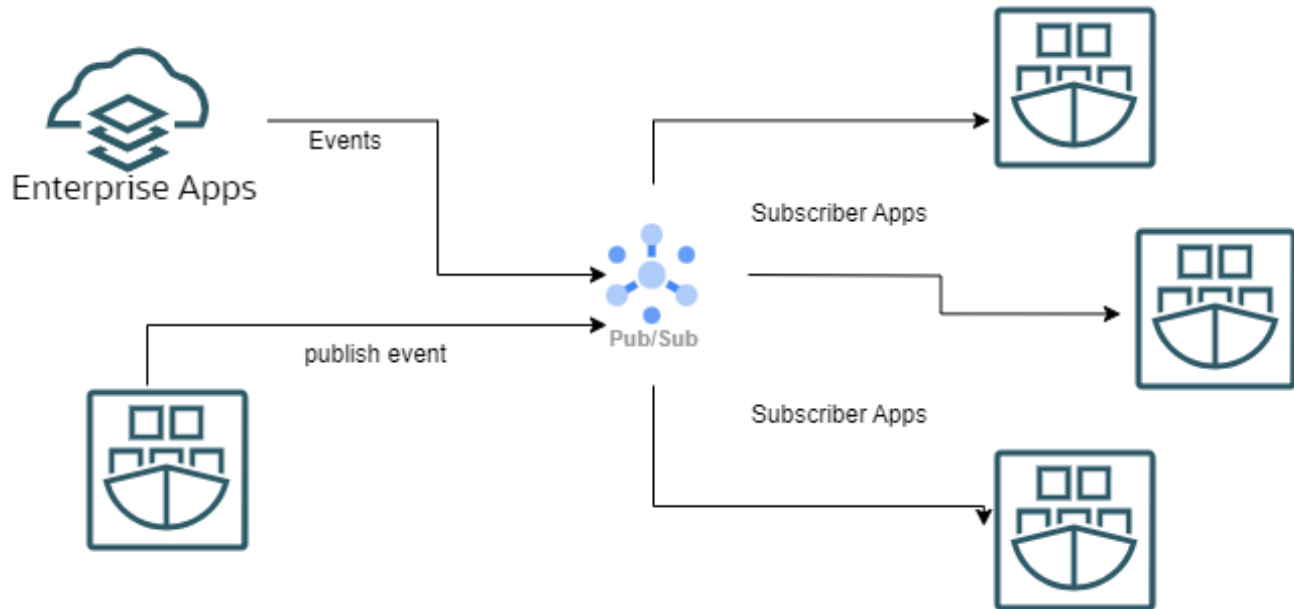
Events usually signify milestones or anomalies in business processes. Examples include the creation of a product item, the end of a business transaction, the abandonment of a shopping cart, a change in sales order status, or IoT device signaling. Events may require processing to accomplish a specific business process and are handled using event-driven integrations.

Based on the event delivery mechanism, EDAs typically follow either a Publish-and-Subscribe (Pub/Sub) or Streaming approach.

## Publish-and-Subscribe

The **publish-and-subscribe** model uses a middleware messaging platform with **topics (events)** that individual event consumers **subscribe** to. Event producers can **publish** events to those event topics, and the messaging middleware takes care of delivering the even to all **subscribers**. Typically, events are delivered once to a subscriber (this is often configurable, however).

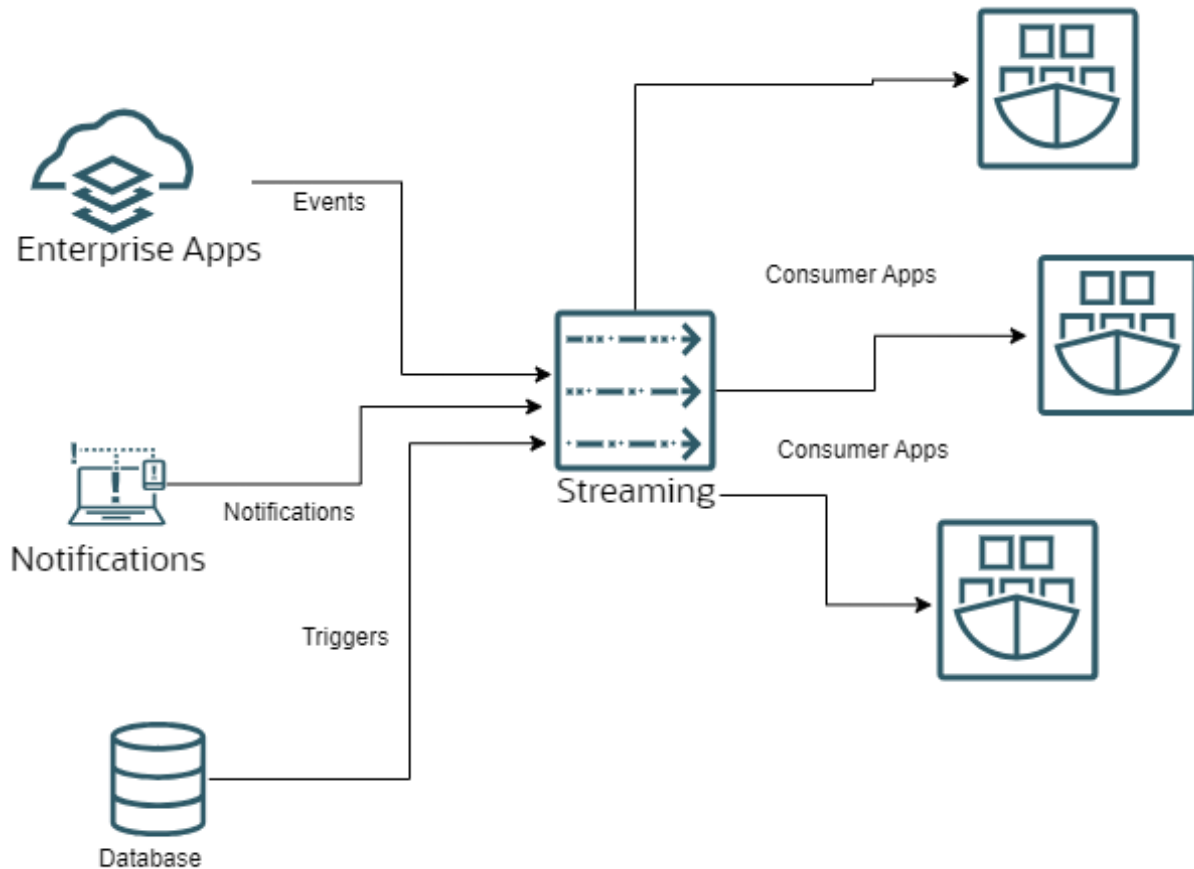
Figure 4 - Pub/Sub diagram



## Streaming (Produce/Consume)

Streaming involves a streaming platform, where events are written to logs which are ordered within partitions. Consumers do not subscribe to a stream but can read from any offset in a stream partition. The consumer typically keeps track of where it has reached in the stream. The streaming platform will typically have a retention period for messages.

Figure 5 - Streaming diagram



## Advantages of an Event-Driven Architecture

### Real-Time Processing

Event-driven integration is ideal for real-time and near-real-time processing.

### Asynchronous

Provides asynchronous communication.

### Fan-Out

It is ideal for fan-out scenarios where multiple services are interested in a single event and can independently process the event in parallel.

### Decoupled

Ideal when integrating heterogeneous services.

### Dynamic

Subscribers can be added at any time.

**Scalable**

Scale up and down based on event volume as triggers. This depends on the integration platform or solution's agility and deployment architecture, which must be able to scale up elastically on demand.

**Highly available**

Multiple instances of consumers can be added to avoid a single point of failure.

**Disadvantages of an Event-Driven Architecture****Sequencing**

Sequence and ordering of events can be lost, especially in distributed clustered services.

**Storage**

It may require large persistence requirements if business processes require larger retention periods (for example, OCI streams retain messages for 1 week, after which they are lost).

**Backlog**

Events buildup caused by consuming systems outages can impact end-to-end performance as consumers may not be able to process bursts of events.

**Transactional**

Due to asynchronous and stateless nature, end to end event-based integrations cannot easily provide ACID or other transaction-like guarantees, which can lead to data inconsistency between services.

**Reliability**

It relies on the integrating systems' ability to generate events reliably (e.g., avoiding duplicate events, handling event generation failures).

**Solution Considerations**

Some things to consider when thinking about implementing an event-driven architecture:

- Monitoring and tracking across multiple components—the platform's ability to provide metrics that can be ingested to provide end-to-end tracking.
- Guaranteed message delivery vs. eventual consistency.
- Certain platforms can provide controls for *exactly-once* and *at-least-once* event delivery mechanisms.
- Complex event processing relies on the Integration tool's ability to handle global transactions and consistency across multiple events.
- One example is support for the SAGA pattern, where a SAGA orchestrator coordinates local transactions and invokes compensations to handle failures.
- Event persistence—Service limits and constraints on persistence capacity can be a deciding factor when choosing the integration platform. Constraints could include:
  - Retention period.
  - Payload size limitations.
- A platform that supports throttling to safeguard end systems can effectively manage scenarios like slow consumers and event backlogs.

## Batch Architecture

### Introduction

Batch architecture refers to batch processes used for communication and data synchronization between systems. It is a method of automating and processing multiple records collected over time as a single group. This pattern is particularly suited for periodic or end-of-cycle processing of data. They usually involve processing large volumes of data in batches and are suited for non-interactive, non-real-time, and long-running interactions. Batch integrations are inherently asynchronous, with batch processes being scheduled at periodic intervals or triggered manually on-demand. Batch architectures are also extensively used for Data Integration, where large volumes of data are processed between applications.

### Examples

Some common examples of batch use cases are:

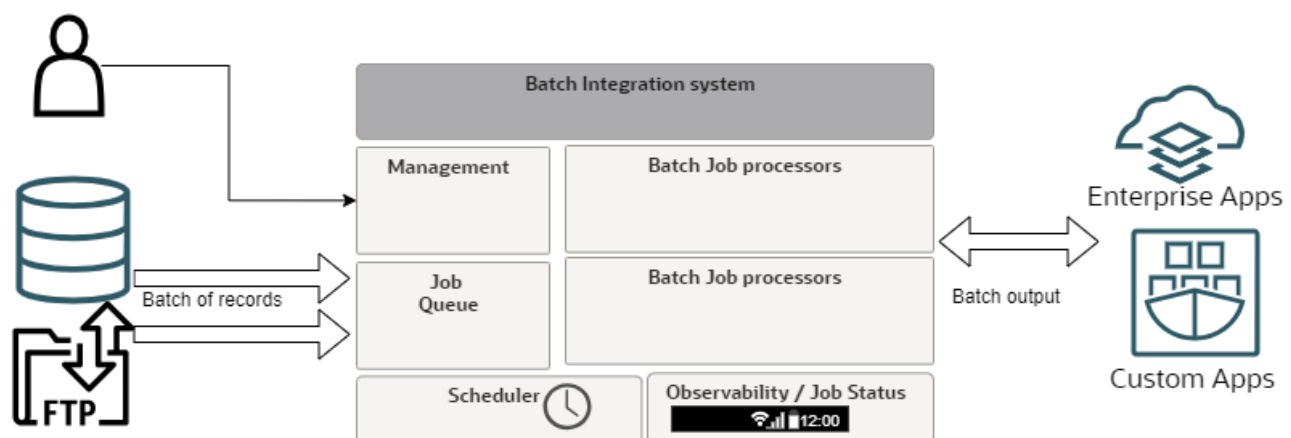
- Monthly bill generation.
- Payroll processing.
- Supply chain fulfillment tasks.
- Data extract and periodic reports generation.
- End-of-day reconciliation of failed transactions between systems.

### Tooling Examples

- Oracle Integration supports batch processing using various file, FTP, and stage operations and provides an inbuilt scheduler for scheduling long-running batch integrations.
- Fusion SaaS—BI Publisher Extracts and Enterprise Scheduling Service (ESS) jobs.
- Other examples include:
  - AWS Batch.
  - Spring Batch.

Common components of batch processing systems include a scheduler, data sources, job executor, job observability, and job queues.

Figure 6 - Batch architecture diagram





## Advantages of Batch Architecture

- Batch integrations are suitable for processing data in large batches.
- They can provide high performance and throughput if sub-batches can be processed independently and in parallel.
- Reconciliation-type integrations required to run overnight and during non-business hours can be modeled as batch jobs.
- Batch integration is also suitable for bulk retry jobs of failed real-time integrations.
- Batch processing can be combined with events. For example, a file upload event can automate triggering a file processing batch job.

## Disadvantages of Batch Architecture

- Batch processing is not suitable for real-time integration between systems.
- Synchronous integrations cannot be modeled using batch processing.
- Complex processing with external dependencies makes batch processing unwieldy.

## Solution Considerations

- The integration platform's support for various data stores makes batch processing flexible, such as a database, file system, FTP server, object storage, and streams.
- The batch tool's ability in scheduling is useful to automate triggering of end of cycle processes.
- Error handling capabilities, such as reprocessing a subset of transactions that failed in a batch and the ability to correct and reconcile failures in batch processing, can be crucial to ensuring data consistency.
- The tool's ability to callback or notify when batch processing completes can enable further downstream automation.
  - For example, Oracle Fusion ESS batch processes can generate callbacks when a batch completes. These callbacks can be integrated with event-driven integrations for automating further processing.
- Batch size restrictions and performance—check for service limits, which can be constraints on large batch processes.
- Parallel processing within the batch—the platform's ability to process sub-batches in parallel can provide better performance and throughput.
- Real-time status of job execution—observability of batch systems.
- Ability to abort and retry batches—manageability and control plane features for batch.
- Ability to configure batch size and frequency based on throughput requirements—configurability to tune batch performance and meet business SLAs.
- Ability to set priorities for certain workloads over others can be important for meeting SLAs.

## Data Integration vs. Application Integration When Considering Batch Processes

As the name suggests, **Application Integration** is used to integrate different applications so that they can work together to automate business processes. For example, this could follow event-based, API-based, or even batch integrations. Whereas **Data Integration** is mainly focused on moving batches of data from one source to another. Typical Data Integrations perform Extract Transform Load (ETL) operations from source to destination to keep data in sync between them.

The line between batch Application Integration and Data Integration could be blurred in many use cases since both process bulk data. Here is an example to differentiate them. A batch integration could be used to consolidate all errored orders from an Order Management System at midnight and process them to a Fulfillment System in batches. Whereas a Data Integration would typically extract all the orders from the Order Management System, transform them to remove personally identifiable information (PII), and load them into a Reporting System, which would then be used to generate daily reports and analytics insights. While the batch Application Integration process performs bulk transaction processing, the Data Integration process is primarily tasked with bulk data transfer.

---

## Business to Business Integration (B2B)

### Introduction

B2B (business-to-business) integration has long been a common use case for Application Integration. Healthcare is a rising use of B2B, with platforms often tailoring B2B capabilities for healthcare-specific uses.

### Advantages of Using iPaaS for B2B and Healthcare

#### Seamless Interoperability

Supporting B2B standards in iPaaS ensures seamless interoperability between different systems and applications used by business partners. Whether it's EDI (Electronic Data Interchange), XML, JSON, or other industry-specific protocols, iPaaS enables organizations to exchange data effortlessly, regardless of the underlying technologies.

#### Accelerated Onboarding Process

Adhering to B2B standards simplifies the onboarding process for new partners. iPaaS platforms provide pre-built connectors, templates, and mappings for commonly used standards, reducing the time and effort required to establish connections and exchange data with trading partners. This accelerated onboarding process translates into faster time-to-market and improved business agility.

#### Reduced Development Costs

Organizations can significantly reduce the costs associated with custom integration development by leveraging standardized formats and protocols. iPaaS platforms offer a library of pre-built connectors and transformation tools that abstract the complexities of B2B integration, minimizing the need for custom code and specialized expertise. This cost-effective approach enables businesses to allocate resources more efficiently and focus on innovation rather than reinventing the integration wheel.

#### Enhanced Data Quality and Compliance

B2B standards define clear guidelines for data formatting, validation, and error handling, ensuring the integrity and accuracy of exchanged information. iPaaS platforms enforce these standards through built-in validation rules and data transformation capabilities, reducing the risk of data errors, discrepancies, and compliance violations. This enhances data quality, fosters trust among trading partners and mitigates regulatory compliance risks.

#### Scalability and Flexibility

Supporting B2B standards in iPaaS empowers organizations to scale their integration capabilities as their business grows and evolves. Whether it's accommodating new partners, expanding into new markets, or integrating additional systems and applications, iPaaS platforms provide the scalability and flexibility needed to adapt to changing business requirements. This agility enables organizations to seize new opportunities, enter new markets, and stay ahead of the competition.

**Improved Visibility and Control**

iPaaS platforms offer comprehensive monitoring, tracking, and analytics capabilities that provide real-time visibility into B2B transactions and workflows. Organizations can monitor the status of data exchanges, track performance metrics, and proactively identify issues or bottlenecks in the integration process. This visibility enhances control over B2B operations, enabling organizations to optimize processes, resolve issues promptly, and meet SLAs (Service Level Agreements) effectively.

**Disadvantages of Using iPaaS for B2B and Healthcare****Complexity of Integration**

B2B and healthcare ecosystems often involve a multitude of applications, systems, and stakeholders. Integrating diverse systems with varying data formats, protocols, and standards can be complex. While iPaaS simplifies integration tasks, configuring and managing integrations across different entities and environments may still require significant effort and expertise.

**Support for All B2B Document Standards**

Many iPaaS support commonly used business exchange documents like EDI or HL7. If a specific B2B document standard is needed that is not supported by the iPaaS platform, then specified B2B Gateway software needs to be adopted.

**Data Security and Compliance Concerns**

B2B and healthcare industries deal with sensitive data such as personal health information (PHI) and financial records. Using iPaaS may raise concerns about data security and compliance with regulations like HIPAA (Health Insurance Portability and Accountability Act), GDPR (General Data Protection Regulation), and other industry-specific standards. Ensuring that the iPaaS platform meets stringent security requirements and complies with relevant regulations is essential.

**Solution Considerations for iPaaS B2B Capabilities**

B2B integration involves the exchange of data between businesses, typically facilitated through electronic data interchange (EDI), APIs, or other integration methods. iPaaS platforms streamline B2B integration by providing prebuilt connectors, data mapping tools, and workflow automation capabilities. This enables healthcare organizations to efficiently exchange data with partners such as insurers, suppliers, and regulatory agencies.

**Key Features of iPaaS Platforms Supporting B2B Integration****Connectivity**

iPaaS platforms offer a wide range of connectors and adapters to seamlessly integrate with various B2B systems and protocols, including EDI, AS2, SFTP, and more.

**Data Transformation**

These platforms facilitate the transformation of data between different formats and standards, ensuring compatibility between disparate systems used by different organizations.

**Workflow Automation**

iPaaS platforms enable the automation of B2B processes, such as order processing, invoicing, and inventory management, reducing manual effort and improving operational efficiency.

**Monitoring and Management**

Advanced monitoring and management capabilities allow organizations to track the status of B2B transactions in real-time, identify bottlenecks, and ensure compliance with service level agreements (SLAs).

## Solution Considerations for iPaaS Healthcare Support Capabilities

Compliance with healthcare standards is essential for ensuring the security, privacy, and interoperability of patient data. iPaaS platforms support a variety of healthcare standards, including HL7, FHIR, DICOM, and HIPAA, among others.

Below are the capabilities that iPaaS should provide to support healthcare standards integration:

### HL7 and FHIR Integration

iPaaS platforms provide native support for HL7 and FHIR standards, allowing healthcare organizations to exchange clinical and administrative data seamlessly.

### DICOM Integration

For medical imaging systems, iPaaS platforms offer DICOM support, enabling the integration of Picture Archiving and Communication Systems (PACS) with electronic health record (EHR) systems and other healthcare applications.

### HIPAA Compliance

iPaaS platforms implement robust security measures and data encryption techniques to ensure compliance with the Health Insurance Portability and Accountability Act (HIPAA) regulations, safeguarding patient data during transit and storage.

### Interoperability

By supporting healthcare standards, iPaaS platforms facilitate interoperability between disparate systems, enabling healthcare providers to access and exchange patient information securely across different care settings and organizations.

## G. A Case Study—Fusion Applications Social Network Integration

### Case Study Overview

This case study explores a new Oracle project that aims to connect Oracle Fusion Applications with social collaboration platforms, addressing the need for improved enterprise collaboration and communication. The study outlines the process from identifying integration challenges to implementing a solution that leverages cloud platforms and low-code environments. It aims to demonstrate how modern integration strategies can be applied to enhance organizational efficiency and collaboration. The study sets the stage for a deeper dive into the specific challenges, techniques, and outcomes of the project while also reflecting some broader themes of the 2024 integration landscape.

The focus of this case study is on the integration of Oracle Fusion Applications and Oracle PaaS for middleware with a designated social collaboration platform. However, the integration strategies and methodologies discussed are designed to be widely applicable. Depending on specific needs and available technologies, this framework can be adapted to interface with various other enterprise applications, middleware layers, and social platforms. This adaptability demonstrates the potential for universal application across different technological environments, highlighting the scalability and versatility of the integration solutions deployed in this project.

### Project Introduction

The need for effective collaboration across diverse platforms has become increasingly evident in the current enterprise landscape. Organizations are looking for innovative ways to enhance communication around critical business processes (aka, Business Objects) like sales opportunities, purchase orders, purchase requisitions, contracts,

and sales orders. This challenge has prompted the exploration of integrating social platforms with core business applications to improve collaboration.

Recognizing this need, the A-Team embarked on a strategic initiative to bridge the gap between Oracle Fusion Applications and the dynamic world of social collaboration. This initiative led to the formation of the Oracle Fusion Applications Social Network Integration (FASNI) project. The following case study offers a comprehensive overview of the FASNI example, detailing the journey from identifying the challenge to implementing a solution that significantly enhances organizational collaboration and communication.

## Problem Statement

The investigation for enhanced collaboration within the enterprise sector has illuminated a pivotal gap: insufficient integration between core business applications and contemporary social collaboration tools. The lack of this integration has surfaced several technical and operational challenges that impede effective communication and collaboration around critical Business Objects:

### Communication Method Inefficiencies

Current communication methods, notably email and standalone messaging platforms, need to support the agile and interactive dialogue essential for quick decision-making in business operations. These traditional methods introduce delays, negatively affecting operational efficiency and business responsiveness.

### Communication Channel Fragmentation

The need for integrated, real-time communication channels or APIs to connect core business applications with collaboration tools leads to fragmented communication flows. This fragmentation creates coordination challenges and information exchange barriers among critical stakeholders, undermining the collaborative effort and creating operational inefficiencies.

### Barriers to Access and Participation

Technical and general barriers, such as complex authentication processes and the absence of integrated access controls or data-sharing capabilities, prevent stakeholder participation in crucial discussions. Without seamless integration to facilitate easy access to information and discussions, the full potential for stakeholder engagement and contribution is not possible, slowing decision-making and innovation.

The initiative to integrate Oracle Fusion Applications with a broader ecosystem of social collaboration platforms is designed to address these challenges. It aims to create a unified, efficient, and accessible framework for communication and collaboration. FASNI will provide a clean, efficient, and straightforward collaborative reference implementation for this integration.

## Objectives

The FASNI project aims to meet comprehensive technical and operational goals:

1. **Cost Efficiency**—Deliver an integration solution that utilizes existing financial investments to avoid additional expenses or licensing costs for customers.
2. **Interface Extension**—Enhance the user interface capabilities, exposing conversations for relevant business objects within the native Fusion UI.
3. **Access Control Integration**—This involves extending Fusion Applications' access control lists into the integration layer, ensuring users have appropriate and secure access rights.
4. **Universal API**—To provide a standard, social collaboration-agnostic API that allows the adoption of various social media platforms.

5. **Performance Excellence**—To ensure the integration operates at a level that meets or surpasses the performance of the current Fusion Applications UI.
6. **Configuration Over Customization**—To minimize reliance on custom code by favoring a configuration-first approach, simplifying the initial setup and ongoing adaptability.
7. **Business Object Updates**—To capture changes to Oracle Business Objects and push those changes to the conversation occurring on the social media platform.
8. **Inclusive Collaboration**—To enable critical stakeholders who are not direct users of Fusion Applications to participate fully in conversations, making the collaborative process more inclusive and broadening the scope of engagement.
9. **Open-Source Framework**—To facilitate independent customer implementation by offering the integration solution as an open-source distribution.

By accomplishing these objectives, the FASNI project is positioned to expand and automate enterprise collaboration, establishing an adaptable, secure, and inclusive environment that aligns with current and future business needs.

## Methodology

The FASNI project's methodology was rooted in a comprehensive analysis of interaction patterns and preferences regarding collaboration on Oracle Business Objects—logical business entities that encapsulate key data, relationships, and business logic pertinent to enterprise functions such as sales, human resources, and finance. By focusing on how different stakeholders, including customers, employees, and partners, engage with these entities, the project ensures that the integration solution is not only technologically sound but also universally applicable and aligned with the diverse needs and usage scenarios of all users. The following steps outline the methodology:

1. **Customer Engagement Analysis**—Conduct a detailed review of the collaboration habits of top-tier customers who actively engage in discussions related to Business Objects.
2. **Platform Preference Survey**—Survey which social platforms are company standards for these customers to identify a target social platform with which to integrate.
3. **Technological Inventory**—Compile an inventory of existing products and services these customers use, assessing how these could be leveraged or integrated with potential social collaboration platforms.
4. **Requirement Gathering**—Engage with SaaS Product Managers (PMs) through interviews and questionnaires to capture specific functional and technical requirements, focusing on usability, performance, and security expectations.
5. **Feasibility Study**—Analyze the compatibility and integration potential of various social platforms with Oracle Fusion Applications, considering API availability, extensibility, and compliance with Oracle's security standards.
6. **Technology Selection Criteria**—Establish criteria for selecting the integration technology, including ease of implementation, adaptability, support for open standards, and alignment with Oracle's existing technological ecosystem.
7. **Pilot Testing**—Conduct pilot tests with a subset of Business Objects on the selected social platform to validate the integration approach and refine the solution based on feedback.
8. **Evaluation and Final Selection**—Evaluate the outcomes of the pilot tests against the established criteria and PM feedback to make an informed decision on the most appropriate combination of social platform and integration technology for the FASNI project.

By adhering to this systematic approach, the FASNI project aimed to develop an integration solution that was customer-centric, scalable, and technologically aligned with the evolving landscape of enterprise collaboration. This



ensured that the selected platforms and technologies met the current demands of Oracle Fusion Applications users and positioned the system to adapt to future challenges and advancements in social collaboration tools.

## Solution Overview

Based on the results of a thorough analysis of customer preferences and existing licenses, the following key decisions were incorporated into the solution design:

1. **Choice and Adoption of Microsoft Teams**—Microsoft Teams was selected as the social platform for integration due to its predominant choice among customers for both internal and external collaboration. Its widespread enterprise adoption and the capability to include external stakeholders in discussions make it the ideal collaborative platform for integration with Oracle Fusion Applications. This choice aligns with the objective of inclusive collaboration, enabling participants to drive collaborative efforts effectively.
2. **Integration with Oracle Integration Cloud (OIC)**—OIC was chosen as the integration layer due to its existing licensing among customers and for its robust capabilities which serve the objectives of cost efficiency and performance excellence. The integration optimizes existing investments and includes:
  - The inclusion of Visual Builder Cloud Service (VBCS) allows for an extended UI that is both functional and intuitive, mirroring the familiar Fusion Applications experience.
  - Support for Lookups aids in a configuration-first approach, simplifying the customization process without extensive coding.
  - Pre-built adapters for Microsoft Graph APIs and Fusion Applications REST APIs, facilitating efficient data exchange and integration.
  - Event handling capabilities to capture and respond to business object changes, keeping the collaborative environment synchronized with business operations.
3. **Social Platform-Independent API**—To respond to the objective of a universal API, the solution includes a platform-agnostic API that supports the integration of various social media platforms, providing adaptability to meet current and future collaboration needs.
4. **Fusion Applications Callback for Authorization**—This aspect of the solution directly addresses the objectives of access control integration and security. The callback mechanism ensures that user access to business objects within the social platform is strictly governed by Fusion Applications' access control lists, upholding data integrity and compliance.
5. **Open-Source Framework**—The solution's open-source framework reflects its objective to facilitate independent customer implementation. It empowers customers to customize and extend the integration, fostering a sense of ownership and adapting to unique business requirements.

By carefully aligning each solution element with the project's objectives, the FASNI project illustrates a clear roadmap from goals to implementation. It showcases a thoughtfully crafted solution that addresses the current challenges of enterprise collaboration and is designed to be scalable and forward-looking. Detailed guidelines and the architectural blueprint can be found at <https://www.ateam-oracle.com/post/fusion-applications-and-microsoft-teams-integration-architecture>, providing insights into the various components of the sample and links for more detail.

## Relationship With the Whitepaper Trends

To correlate the FASNI case study with the integration trends mentioned in this whitepaper, we highlight how the project aligns with the latest industry trajectories and technological advancements. Let's explore the relevant trends from the whitepaper and relate them to the aspects of the FASNI case study.

1. **Application Integration Pillar**—The FASNI project's integration of Oracle Fusion Applications with Microsoft Teams is a prime example of Application Integration, where two distinct software services are linked to enhance collaboration and workflow continuity.

2. **iPaaS Utilization**—Utilizing OIC aligns with the trend toward cloud-based integration platforms. Per the whitepaper, iPaaS is key to modern integration, offering components like APIs and middleware for efficient communication between disparate applications.
3. **Low Code Solutions**—Utilizing OIC and the FASNI initiative's preference for a configuration-first approach over extensive custom coding corresponds with the move towards low-code platforms. This resonates with the industry's pivot to more accessible solutions that enable rapid development and deployment.
4. **API-First Development**—The whitepaper discusses the significance of API-first development, and FASNI's creation of a social collaboration-agnostic API aligns with this trend. It emphasizes a standards-based, interoperable approach critical for flexible and scalable integrations.
5. **Security and Compliance**—According to the whitepaper, FASNI's use of a callback mechanism to Oracle Fusion Applications for user authorization ensures that the integration adheres to strict security and compliance standards, a trend that is becoming increasingly important.
6. **Multi-cloud and Hybrid Integrations**—Although FASNI's current scope involves Oracle products and Microsoft Teams, the industry is moving towards multi-cloud strategies. The FASNI API's platform-agnostic nature positions it well for future expansion into multi-cloud environments.
7. **Prebuilt Integrations and Connectors**—FASNI leverages OIC's pre-built adapters for Microsoft Graph APIs and Fusion REST APIs, which exemplifies the whitepaper's mention of the advantage of prebuilt connectors and adapters for quick and efficient integration.

By drawing these connections, we can position the FASNI project as a forward-thinking example that embodies several of the key integration trends of 2024, emphasizing its relevance and strategic foresight within the industry's future trajectory.

## In Summary

The FASNI project illustrates a targeted response to the integration needs within the enterprise software domain. The initiative directly addressed the identified collaboration challenges by connecting Oracle Fusion Applications with a collaborative platform, offering a practical solution rooted in current usage trends and customer feedback.

As highlighted by the alignment with the integration trends discussed in the accompanying whitepaper, FASNI embodies the latest industry trajectories and technological advancements. The project's utilization of iPaaS, low-code solutions, and an API-first approach reflects a commitment to current and future industry standards, ensuring its position at the forefront of integration technology. This strategic foresight not only enhances the project's immediate effectiveness but also solidifies its role as a model for future integration efforts across various technological landscapes.

Furthermore, the project's emphasis on configuration over customization, coupled with its open-source distribution and strict adherence to security and compliance standards, supports the broader move towards accessible, secure, and sustainable technology practices. The inclusion of stakeholders outside Fusion Applications' core user base into collaborative processes signifies progress towards more inclusive and comprehensive communication strategies within organizations.

In summary, the outcomes of the FASNI project have significantly bolstered the collaborative capabilities of Oracle Fusion Applications users, showcasing a practical implementation of integration technology that is aligned with both current needs and future possibilities.




## H. Conclusion

Integration in 2024 is a thriving software development domain. There are many approaches to solving a problem, with tooling to suit many different tastes and enterprise development approaches. There is a vibrant future for integration, with many future paths and exciting new technologies poised to make it an even richer and more useful tool.

The intent of this whitepaper is to arm the reader with the knowledge to choose the most suitable approach to solve their integration challenges. The deep dive sections of Application Integration and Process Integration provide the reader with guidelines to choose the iPaaS or Integration platform for their integration requirements. Further in the Development and Deployment trends, we saw the various choices of integration patterns and deployment architectures to not only build solid integrations for today but also make them future-ready by providing a glimpse into where the integration trends are evolving in the future. Finally, the FASNI case study was designed to provide a real business use case that ties the concepts discussed in this whitepaper and applies them as a methodology for architecting good integration solutions.

## Connect with us

Call +1.800.ORACLE1 or visit **oracle.com**. Outside North America, find your local office at: **oracle.com/contact**.

 [ateam-oracle.com](mailto:ateam-oracle.com)

 [facebook.com/oracle](https://facebook.com/oracle)

 [twitter.com/oracle](https://twitter.com/oracle)

Copyright © 2024, Oracle and/or its affiliates. This document is provided for information purposes only, and the contents hereof are subject to change without notice. This document is not warranted to be error-free, nor subject to any other warranties or conditions, whether expressed orally or implied in law, including implied warranties and conditions of merchantability or fitness for a particular purpose. We specifically disclaim any liability with respect to this document, and no contractual obligations are formed either directly or indirectly by this document. This document may not be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without our prior written permission.

Oracle, Java, MySQL, and NetSuite are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.