International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING
ISSN:2147-6799 www.ijisae.org Original Research Paper

Hybrid Integration Architecture for Enterprise Digital Transformation: Combining Cloud-Native iPaaS and API-Led Connectivity for Real-Time Order-to-Cash and Logistics Automation

Rajasekhar Reddy Putta

Abstract: Enterprise digital transformation at scale also requires integration architectures that can connect ERP and CRM systems to mission-critical business processes at the required reliability, latency, and governance levels. This is commonly achieved with a hybrid integration architecture that combines an iPaaS suite optimized for SAP-native semantics and an API-led EAI platform optimized for inter-ecosystem integration that avoids the limitations imposed by legacy middleware and eases the adoption of event-driven, API-first, and cloud-native integration patterns. The two flagship integration scenarios, an OMS-ERP Order to Cash (O2C) flow along with an ERP-3PL logistics automation flow, based on Event-Driven Architecture (EDA), show how synchronous API orchestration can be combined with asynchronous event distribution for low-latency, scalable order fulfillment and partner enablement. Governance is accomplished with a federated API lifecycle model, OAuth2/JWT centralized policy enforcement, and a unified observability based on iFlow execution telemetry, events broker telemetry, and API transaction traceability. Prebuilt accelerator content, OOTB (Out-of-the-Box) connectors, and canonical data models minimize customization and time to market. Protocol-aware reprocessing design and separation of IDoc (Intermediate Document) persistence from OData/SOAP (Simple Object Access Protocol) non-persistence enable predictable incident recovery and minimize reliance on wide-ranging system knowledge. The hybrid architecture enables enterprise-class design for sustained integration maturity as business volumes, partner ecosystems, and the number of platforms increase over time.

Keywords: *Hybrid Integration Architecture, API-Led Connectivity, Event-Driven Architecture, Enterprise Application Integration, Order-to-Cash Automation.*

1. Introduction

Out of necessity, to accommodate a new generation of real-time cross-platform digital experiences required by modern organizations, integration architecture is often referred to as a "strategic" rather than purely "technical" endeavor. Legacy point-to-point middleware and monolithic integration buses are simply unable to cope with the latency, multi-cloud deployments, and event volumes that modern business processes demand [1]. Indeed, as (customer-facing) processes (executed in CRM & OMS) become increasingly decoupled from (operational) processes (executed in ERP systems), the integration layer becomes a foundational element of enterprise IT [2].

A hybrid integration architecture addresses this by using two complementary platforms: an integration Platform as a service (iPaaS) suite for SAP-native semantics, eventing, and business process content; and an API-led integration platform to connect systems through a multi-layered API architecture [3]. The iPaaS layer enables deep integration with SAP business processes using pre-packaged integration flows (iFlows) running on the Apache Camel runtime, a cloud-native event broker for real-time event distribution, and a central API gateway for policy enforcement [4]. API-led layer: supplemented by reusable System APIs (stable façades to systems of record), Process APIs (orchestration and business rules), and experience APIs (channel-oriented consumer views), as well as certified connectors and prebuilt accelerator content for SAP and CRM systems [5].

The article describes how to design, govern, and operate this hybrid pattern to deliver business value with demonstrable outcomes in the OMS to ERP order creation in the O2C (Order to Cash) process

Pondicherry University, India.

and ERP to 3PL event distribution through Electronic Data Interchange (EDI) and Representational State Transfer (REST) and the program management, governance, and observability for reliable and scalable delivery [6].

The model is based on standard integration architecture frameworks, including ISA-M (Integration Solution Advisory Methodology), API-led connectivity, SAP architecture, and published B2B/EDI practitioner approaches. The model provides a way to deliver integration architectures with the following characteristics: reusing out-of-the-box (OOTB) content, canonical data models, end-to-end observability, and federated lifecycle governance [1][3][4].

Integration Layer	Primary Function	Protocol Support	Deployment Model
iPaaS (SAP-native)	SAP business process integration	AMQP, Webhook, OData, SOAP	Cloud-managed iFlow runtime
API-led EAI	Cross-ecosystem connectivity	REST, OData, IDoc, RFC	Multi-cloud API runtime
Event Broker	Asynchronous message distribution	AMQP, Webhook	Cloud-native managed service
API Gateway (iPaaS)	Policy enforcement for ERP endpoints	OAuth2, JWT	Centralized gateway
API Manager (EAI)	Design standards and consumption analytics	OAuth2, REST	Federated API manager

Table 1: Hybrid Integration Platform Roles and Domain Assignments [1][3][4]

2. Reference Architecture and Hybrid Platform Rationale

An effective hybrid integration architecture places each platform in the integration domain where it offers the greatest native value. For example, ERP serves as the system of record (SoR) for orders, pricing, inventory, and financial documents. The CRM/OMS provides omni-channel order capture and customer engagement; the event broker provides asynchronous durable messaging; and the iPaaS suite handles integration flows using the native semantics, such as events and APIs, of SAP applications. The API-led platform links services and applications through reusable, versioned APIs [3][4].

The hybrid integration, which is the main motivation for this iPaaS, has a suite tailored for this purpose: understanding SAP business process semantics; SAP-specific eventing such as AMQP and Webhooks through a cloud event broker; packaged integration content for Lead-to-Cash and Procure-to-Pay; integration runtime for synchronous and asynchronous processing of iFlows; and a central API gateway with OAuth2 enforcement, quota, and identity propagation [4][5]. The API-led and application-centric integration is complemented by a multi-layer connectivity architecture, declarative transformation support with a domain-specific Mapping Model Language, certified SAP OData/SOAP/IDoc/RFC connectors for core integrations, and accelerator content for connecting Salesforce to SAP [5][7].

As an integration fabric, these platforms are based on ISA-M guidance on using a use case-driven approach to platform selection rather than a homogeneous middleware usage [1]. This hybrid approach can help avoid vendor lock-in and enable federated governance models and change management by isolating SAP-native technical complexity (in the iPaaS layer) with stable, versioned API contracts available to all other consumers [3]. Target state reference architectures, such as those on the SAP BTP GitHub repository or SAP PRESS architecture deep-dives, are good

places to validate designs quickly at program kickoff [4][6].

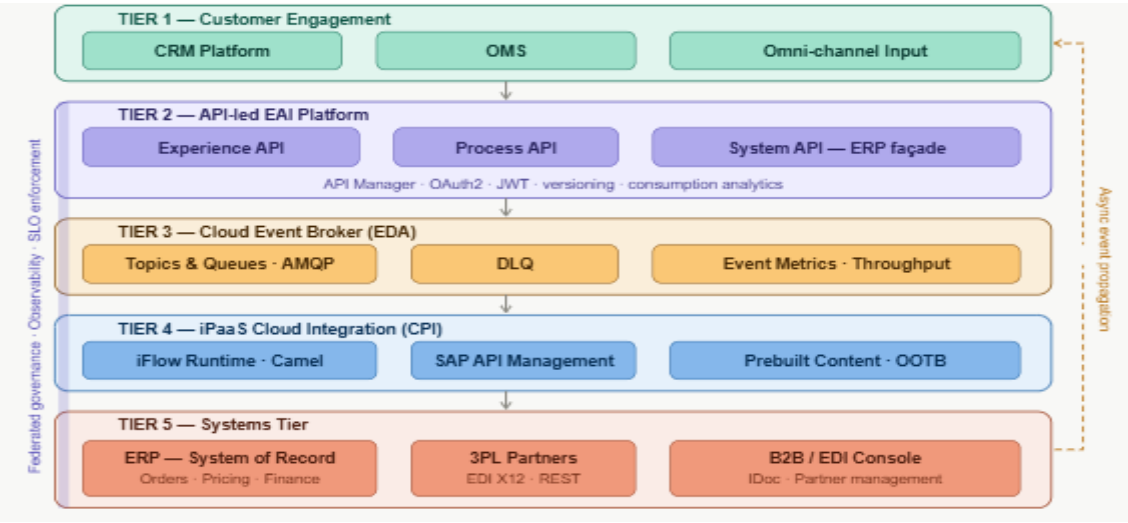

Fig. 1. Hybrid Integration Reference Architecture for Enterprise Digital Transformation [1][3][4]

3. Use Case A: OMS-to-ERP Order-to-Cash Integration

The first scenario is transferring an order from an OMS to an ERP system. A customer order being created in an OMS generates an event targeting a channel-specific Experience API. A Process API is triggered, which checks the structure of the payload, adds missing attributes and calls a System API, which is a stable facade over the ERP's OData and SOAP APIs [5]. The System API creates the Sales Order in the ERP. Once the created business event has been generated, the event broker then asynchronously publishes it to the OMS and downstream subscribers to transition order statuses without blocking the customer-facing transaction [7].

This hybrid synchronous order creation and asynchronous status propagation is a prevailing approach for low-latency O2C processing. Synchronous calls are used where an immediate confirmation is required, such as availability to promise (ATP) checks, pricing and order commit. This is supplemented by event-driven distribution for other order processing functions, such as CRM record updates or fulfillment triggers [4]. Prebuilt accelerator content for SAP includes an O2C template, canonical payload mappings, and SAP business object schemas, which drastically reduce the customization and ease integration testing [5][7].

In addition to meaning data contracts (the canonical message schemas that are agreed to by OMS and the ERP teams), SLA targets for how quickly orders should be created, and integration rehearsals that test the end-to-end completeness of the flows for freight orders, outbound deliveries, and billing documents as specified in the OMS implementation guides [7], the API-led layering also allows changes to the internal API surface of the ERP not to bleed into the external surface that is consumed by OMS. This makes it less risky to upgrade the ERP and allows for independent versioning of integration components [3][5].

API Layer	Responsibility	Consumer	ERP Interface Used
Experience API	Channel-specific order event reception	OMS (Order Management System)	None (façade only)
Process API	Payload validation, enrichment, orchestration	Experience API	None (orchestration only)
System API	Stable façade over ERP transactional interfaces	Process API	OData, SOAP
Event Broker	Asynchronous Sales Order. Created distribution	OMS, downstream subscribers	AMQP, Webhook

Table 2: API Layer Architecture for OMS-to-ERP Order-to-Cash Flow [5][7]

4. Use Case B: Automate end-to-end logistics, from ERP systems to 3PL via event-driven architecture

The second scenario involves the automation of outbound logistics by publishing ERP delivery and goods-movement events to 3PL partners. When the ERP publishes a delivery event or goods-movement document to the event broker over AMQP or Webhook, the iPaaS Cloud Integration runtime subscribes to the event queue and hydrates the integration payload where additional information is needed (e.g., a synchronous API call to the system of record) and transforms it to the partner's desired format (EDI X12 transaction sets or REST, depending on partner capability) [4][8]. Messages that cannot be delivered are sent to dead-letter queues (DLQs). Event broker metrics show the depth and throughput of the queue [4].

Another advantage of this approach is the decoupling of the ERP core from the partner integrations in the case of the logistics industry, where partners usually create different EDI transaction sets, message versions, and acknowledgments [8]. In addition, the ability of the event broker to scale to the peak load (for example, when shipments peak at the end of a quarter) is a structural advantage over synchronous point-to-point partner integrations, which would require the ERP to maintain many concurrent outbound connections [4][9].

With the partner ecosystem standardized on the API-led platform's B2B stack, a B2B/EDI console provides end-to-end transaction visibility and IDP capabilities. SAP PRESS's implementation articles detail the setup of the SAP example configuration, master data dependencies, IDoc mapping, and partner reconciliation processes required for fit-to-standard delivery scenarios in warehousing and transportation. [9]. The IDoc protocol, in particular, allows complete persistence and reprocessability of messages in the ERP, allowing failure recovery and audit trails to be achieved in contexts with high regulation, such as logistics [8][9].

Stage	Trigger	Transformation Target	Delivery Mechanism	Failure Handling
ERP event publication	Delivery / goods-movement document	N/A	AMQP / Webhook to event broker	DLQ (Dead-Letter Queue)
iPaaS subscription	Event queue arrival	Payload hydration via synchronous API	iFlow runtime	Retry with backoff
Partner format transformation	Hydrated payload	EDI X12 or REST per partner capability	B2B/EDI console	Transaction visibility log
IDoc reprocessing	Failure resolution in ERP	N/A	Native ERP reprocessing	Full IDoc persistence

Table 3: ERP-to-3PL Event-Driven Logistics Flow: Message Handling Characteristics [4] [8] [9]

5. Governance, Observability, and Safe Reprocessing

Governance and observability apply to the end-to-end message journey (API design through event propagation into a partner and confirmation receipt) and are federated per platform domain. For the iPaaS, security policies (OAuth2/JWT, rate quotas, and IP allow-listing) are set at the API gateway against all connectivity to endpoints on the ERP. For the API-led approach, API design and governance policies, versioning strategies, and consumption metrics are enforced at all API layers [3][5]. For example, integration assets (iFlows, API specifications, and connectors) are stored in the platform's asset registry to avoid shadow integrations and make them discoverable across teams and geographies [5][10].

A semantic separation between notification events (lightweight, triggering downstream queries) and data-carrying events (carrying the entire payload, obviating downstream queries) balances the coupling/performance tradeoff. The event broker's topic and queue model allows subscribers to receive only events relevant to their domain, reducing coupling to the entire ERP event stream [4][11].

Observability end-to-end: The iPaaS message monitoring console shows the status of the iFlow and the adapter. The event broker shows the queue statistics, throughput, and dead letter queue rates. The API-led platform contains a monitoring capability that provides API call logs, metrics, and distributed traces for API health and transaction traceability [10][12]. This information can be used for troubleshooting as well as creating an audit trail

of an assembly line and regulatory processes like revenue recognition and customs declaration.

Reprocessing is protocol-dependent and must be implemented. IDocs are fully persisted in the ERP, allowing them to be natively reprocessed after the condition is resolved. OData and SOAP processing are not guaranteed to persist in the ERP, so failure messages should be caught at the API gateway or middleware, and retry and manual recovery processes should be documented for this pattern [8][12]. If these protocol differences are acknowledged, no surprises should be encountered in production, and operational impacts from erroneous messages can be minimized.

Program governance is a hybrid delivery train comprising platform squads (one per integration platform), domain squads (O2C, Logistics), and a cross-functional Architecture and Governance (A&G) board [6]. Stage gates (blueprint, build, performance validation, and business validation) provide quality control at each stage in the delivery process. SLOs (Service Level Objectives) for p95 and p99 latency, cutover playbooks, post-go-live hypercare, and observability dashboards related to business SLAs are created before UAT (User Acceptance Testing) [6][13]. These are part of an operational readiness model.

Platform Component	Monitoring Capability	Telemetry Type	Operational Use
iPaaS Cloud Integration	iFlow execution status, adapter behavior	Message-level logs	Failure detection, SLA tracking
Event Broker	Queue depth, throughput, DLQ rates	Metrics	Capacity management, backpressure detection
API-led Platform	API health, transaction traces	Logs, metrics, distributed traces	Latency profiling, audit lineage
API Gateway (iPaaS)	Policy violations, quota consumption	Access logs	Security compliance, throttling enforcement

Table 4: Observability Scope by Platform Component [4][10][12]

Conclusion

Enterprise integration between ERP and CRM systems spans both iPaaS and API-led EAI architectures. The best architecture connects the domain of native value of integration platforms: iPaaS for SAP-native eventing, packaged process content, and centralized API policy enforcement; API-led EAI for cross-ecosystem connectivity, reusable API layering, and accelerator-driven quote-to-cash (O2C) process automation. OMS-to-ERP illustrates how synchronous CRO and asynchronous event-driven status propagation can deliver low-latency Order-to-Cash without requiring a coupling between customer-facing and operational systems. The ERP-to-3PL example illustrates how an event broker acting as a decoupling layer between an ERP core and a heterogeneous, third-party logistics partner ecosystem can deliver the scalability, DLQ-backed reliability, and IDoc reprocessability required when dealing with regulated logistics environments. Federated API governance, unified end-to-end observability, and explicit protocol-aware reprocessing design enable an operational discipline for predictable, audit-capable enterprise operations. Combined with OOTB connectors, prepackaged iFlow content, and canonical API models published with accelerators, the integration fabric minimizes custom build and long-term maintenance costs while helping retain the architectural flexibility to scale with increasing business complexity. The hybrid iPaaS-plus-API-led model is therefore an enterprise-ready blueprint for digital transformation initiatives that require both immediate delivery velocity and long-term integration maturity.

References

[1] SAP Community, "SAP Integration Architecture Guide," SAP Community Network. [Online]. Available: https://help.sap.com/docs/sap-btp-guidance-framework/integration-architecture-guide/summary

[2] Hohpe & Woolf, "Enterprise Integration Patterns," Addison-Wesley, 2004. https://ptgmedia.pearsoncmg.com/images/9780321200686/samplepages/0321200683.pdf

[3] SAP SE, *"Integration Architecture Guide,"* Available: https://help.sap.com/docs/sap-btp-guidance-framework/integration-architecture-guide/summary

[4] SAP SE, "SAP Integration Suite Architecture Guide," SAP Help Portal, 2024. https://community.sap.com/t5/technology-blog-posts-by-members/sap-integration-architecture-guide/ba-p/13955539
[5] MuleSoft, "API-Led Connectivity," Whitepaper, 2023. https://www.mulesoft.com/lp/whitepaper/api/api-led-connectivity
[6] TOGAF Standard, "Enterprise Architecture Framework," The Open Group, 2022.https://www.opengroup.org/togaf
[7] MuleSoft, "MuleSoft Accelerator for SAP," MuleSoft Accelerator Documentation. [Online]. Available: https://www.mulesoft.com/exchange/org.mule.examples/mulesoft-accelerator-for-sap/minor/1.4/pages/Use%20case%203%20-%20Quote-to-cash/?no_navbar=1&no_cookie=1&embedded=1&is_marketing=1
[8] SAP PRESS, "What Is SAP Event Mesh?" SAP PRESS Blog. [Online]. Available: https://help.sap.com/docs/event-mesh/event-mesh/what-is-sap-event-mesh
[9] SAP PRESS, "Integration with SAP S/4HANA," 2023.https://www.opengroup.org/togaf
[10] Abraham Asher, "Developing a B2B E-Commerce Implementation Framework: A Study of EDI Implementation for Procurement," Taylor & Francis online, [Online]. Available: https://www.tandfonline.com/doi/abs/10.1080/10580530701586144
[11] Richardson, C., "Microservices Patterns," Manning, 2018.https://microservices.io/patterns/cn/data/event-driven-architecture.html
[12] Dipan Ghosh, "Secure On-Premise API using SAP API Management in SAP Integration Suite," SAP Community, 2022. [Online]. Available: https://community.sap.com/t5/technology-blog-posts-by-members/secure-on-premise-api-using-sap-api-management-in-sap-integration-suite/ba-p/13524781
[13] Salesforce, "Salesforce Order Management." [online]. Available: https://help.salesforce.com/s/articleView?id=commerce.om_order_management.htm&type=5
[14] GS1, "EDI Standards for Supply Chain Integration," 2022.https://www.gs1.org/standards/edi
[15] MuleSoft, "API-Led Connectivity," MuleSoft Whitepaper. [Online]. Available: https://www.mulesoft.com/lp/whitepaper/api/api-led-connectivity?d=701ed00000ag1TyAAI&nc=701ed00000agclDAAQ&utm_content=701ed00000ag1TyAAI&gclsrc=aw.ds&gad_source=1&gad_campaignid=22826734684&gbraid=0AAAABAVuWUkrWZZqWCUtqSiqpyWSlR8bj&gclid=Cj0KCQjwqPLOBhCiARIsAKRMPZpNrXra7JKLTIwldILjmNWQKZ_Be0esNgWc3idfwkpkg3xqi2mT6tUaAku9EALw_wcB