Gartner

Licensed for Distribution

This research note is restricted to the personal use of Can Huynh (can.huynh@loto-quebec.com).

Choosing Application Integration Platform Technology

Published 5 July 2019 - ID G00383317 - 54 min read

By Analysts Matt Brasier, Gary Olliffe

Initiatives:Integration Architecture and Platforms for Technical Professionals

Integration requirements and technology are more diverse than ever. Application technical professionals responsible for integration architecture should use this comparison to select the right combination of technologies, including iPaaS, ESB and API management, for their integration workloads.

Overview

Key Findings

- Mature organizations are creating hybrid integration platforms (HIPs) from distinct and complementary tools, and providing clear guidance regarding which tool to use for which use cases.
- Integration platform vendors have embraced cloud delivery models. Integration platform as a service (iPaaS) has become the default delivery model for modern integration. The innovation in the integration market is happening in these products.
- Characteristics that were previously associated with iPaaS, such as a simple web-based user experience focused on ad hoc integrators, are now available in modern integration platform software.
- Pervasive integration requirements multiply the demand for integration work and increase urgency. Expanding the integrator resource pool beyond centralized integration teams to include ad hoc integrators is essential to meet that demand.

Recommendations

Application technical professionals choosing an integration technology platform should:

- Use iPaaS for SaaS or API-centric integration and to retain centralized oversight and governance while including “ad hoc integrator” tasks.
- Use existing platform enterprise service bus (ESB) deployments for predictable and systematic integration requirements that must be centrally developed, managed, monitored and governed.
- Use a distributed integration platform (DIP) when creating a platform for ad hoc integrators that are connecting modern API-based systems or are integrating within departments but are not moving application workloads to the cloud.
- Use the API management platform only to mediate between API interface contracts and to enforce API policies, not to implement the API by creating business logic or as a generic integration technology.

Comparison

Modern applications and modern system architectures are inherently heterogeneous, distributed and diverse. At the same time, business goals such as optimized customer experience, process efficiency, data quality and digital business agility require application integration architects to connect and compose the elements of those systems in timely, efficient, reliable, secure and cost-effective ways.

This assessment is intended to help integration architects evaluate and compare the most common options for addressing application integration requirements. This may be as part of procurement activity to guide investment in new integration platform capabilities or as part of your project’s design phase when choosing between existing capabilities deployed as part of a hybrid integration platform.

The demand for integration is increasing, and a team of dedicated “specialist integrators” using complex integration software can no longer meet the needs of the business. To avoid shadow-IT integrations appearing, you must provide a platform that enables “ad hoc” integrators; that is, people who have a technical background and can create integration flows when given simple-to-use tools.

Innovations and new customer experiences that originated in the iPaaS market are now available in integration platform software, such as IBM App Connect, MuleSoft Anypoint and TIBCO BusinessWorks Container Edition. Most organizations have existing integration middleware that cannot be easily retired, and that offer more connectors for legacy systems. The result is a diverse landscape of integration technologies to choose from. This comparison explores the strengths, weaknesses and overlapping capabilities for five of the most common application integration technologies of interest to Gartner clients. These five technologies are:

- **Traditional platform ESB** is the traditional deployment model for integration suites and platforms. This category assumes you are centrally deploying a farm of ESB servers and managing the software centrally. You develop integration flows using application and technology connectors,

logic flow steps and transformations, and deploy them into or onto the ESB platform. Examples include Microsoft BizTalk Server, Oracle Service Bus and TIBCO BusinessWorks.

- **Integration platform as a service** provides integration capabilities using a vendor-managed and provisioned model. They often include web-based integration development tools that provide both ease of access and ease of use. The defining feature of an iPaaS is that the management features and control plane of the platform exists in the cloud, and the provider hosts and manages it. Integration flows can be deployed and executed either in the iPaaS provider's cloud infrastructure or, in the case of leading products, in vendor-managed integration runtimes installed on infrastructure controlled by the customer (either on-premises or IaaS). Examples include Dell Boomi, Informatica Intelligent Cloud Services, Jitterbit and SnapLogic.

- **Distributed integration platform** covers integration platform software that you have to install and manage yourself, where the integration flows do not share a single runtime. These technologies mirror many of the advances seen in an iPaaS, such as a simpler user interface aimed at ad hoc integrators, but without being hosted and managed for you by the vendor. See "Enabling Agile Integration With a Distributed Integration Platform" for more details. Examples, when configured and deployed appropriately, include IBM Integration Bus, MuleSoft Anypoint Platform's Mule runtime engine, Red Hat Fuse and TIBCO BusinessWorks Container Edition.

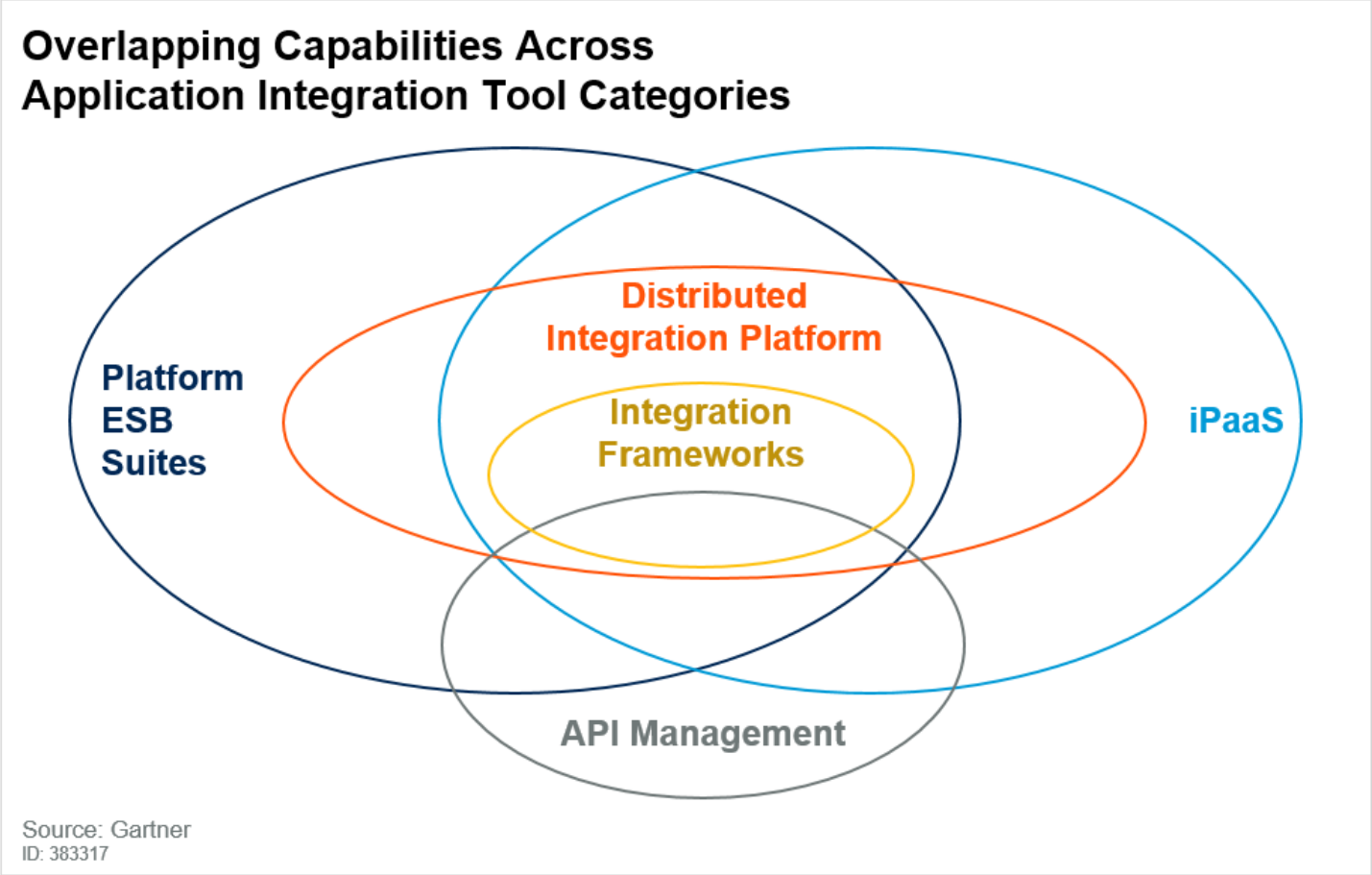
- **Integration frameworks** provide a lightweight approach to developing and embedding integration flows into your application and service implementations. They provide a code- or configuration-based domain-specific language (DSL) that simplifies the implementation of common integration patterns; they also provide adapters to common application technologies and protocols. In some cases (for example, Apache Camel and Mule Embedded Runtime), these integration frameworks are at the heart of the runtime engine used in the other technology categories.

- **API management** products and technologies allow you to define and publish APIs using web-based protocols and formats. They focus on developer (consumer) enablement, policy enforcement and lightweight mediation of underlying service implementations. Mediation features within API management products create an overlap with dedicated integration platforms. Examples include Apigee Edge, Broadcom-CA Technologies API Management, Red Hat 3scale and TIBCO Software Cloud Mashery.

You can use any of these technologies to mediate and orchestrate interactions between applications, data and services, and in general, their technical capabilities overlap, as shown in Figure 1.

Some products break or extend these definitions or offer capabilities that fall into two or more categories, and many vendors offer suites consisting of products in multiple categories.

Figure 1. Overlapping Capabilities Across Application Integration Tool Categories



Integration Technology Selection Criteria

When comparing these categories, Gartner used the following criteria, which define the most common implementation and operational attributes that influence the choice of integration technology:

- Integration implementation criteria:
  - **Complex composition and orchestration of existing assets** assesses the ability to implement and execute complex integration logic, including sophisticated data transformation, protocol mediation and orchestration of integration flows.
  - **API enablement and mediation** assesses support for the creation and controlled publication of APIs that give access to application services, which may be composed using the tool.
  - **Application (COTS) and legacy technology connectivity** requires strong support for connectivity with widely deployed commercial third-party applications (for example, ERP, CRM or HRM) and

legacy application technologies.

- **Cloud application connectivity** requires strong support for connectivity with widely used software as a service (SaaS) applications and modern application protocols.
- **Support for ad hoc integrators** assesses how easy a technology is to learn and use productively without the need for job specialization in integration.
- **Stream-based integration** assesses how well the technology supports performing integration mediation on streams of data, rather than on discrete payloads.

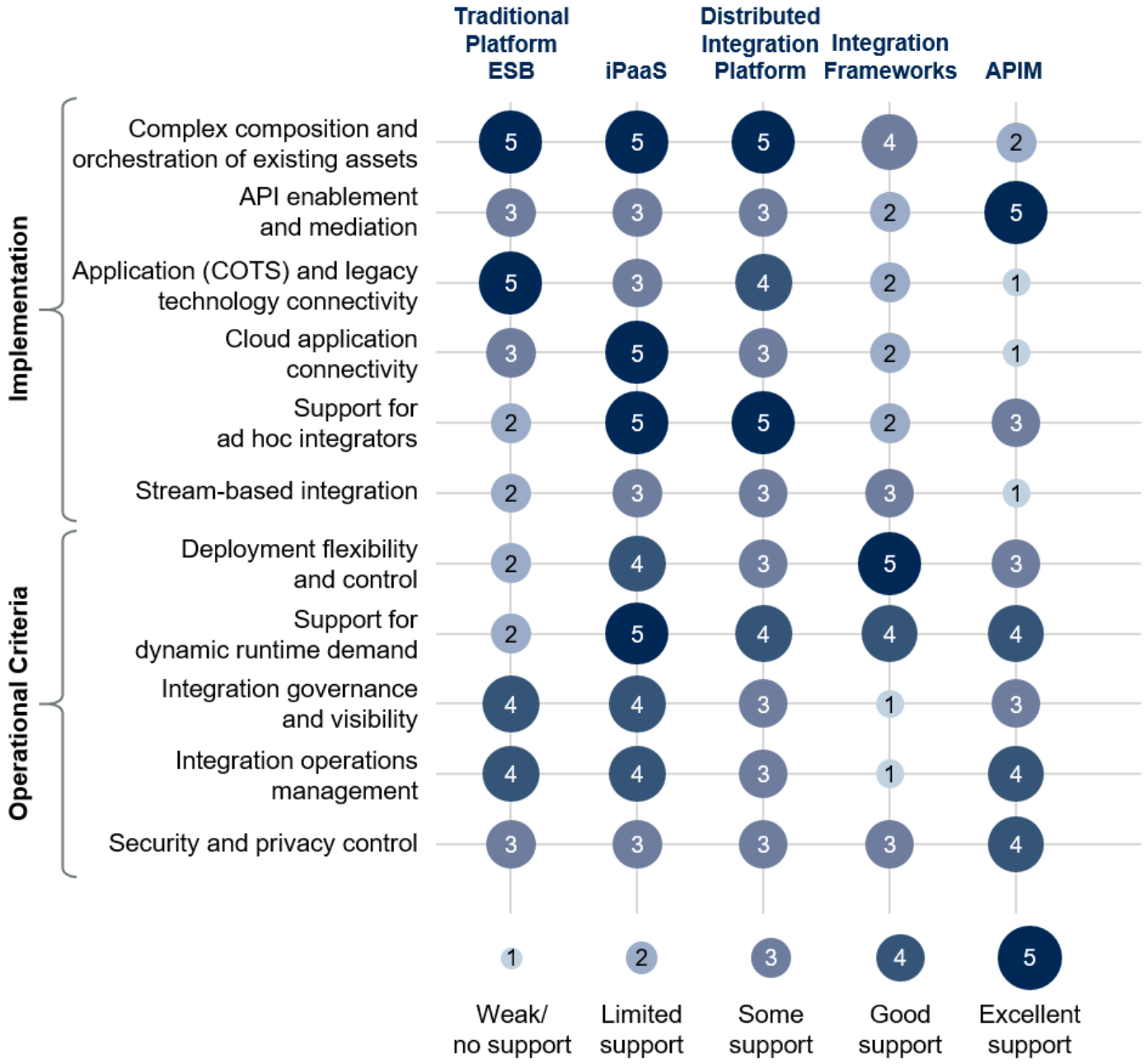
- Integration operational criteria:
  - **Deployment flexibility and control** assesses the support for a flexible and adaptable deployment model that is easy to change over time and can, if necessary, be controlled by the team delivering integration work.
  - **Support for dynamic runtime demand** requires deployment and operational models that support workloads with demand curves that are cyclical or unpredictable.
  - **Integration governance and visibility** assesses how well an organization can maintain appropriate control and visibility over the system and application dependencies that you implement via integration logic and application services.
  - **Integration operations management** requires strong support for the day-to-day management and operation of a portfolio or catalog of integrated services and the platform on which they run.
  - **Security and privacy control** assesses how well an organization can enforce and manage its security and compliance policies associated with application services and integration.

The Analysis section of this assessment describes these criteria in more detail and provides further definitions of the technologies compared in this assessment.

Figure 2 shows Gartner's comparison of each technology against these criteria, which should be used in conjunction with the Guidance section of this assessment to understand how these technologies support a variety of common scenarios.

Figure 2. Comparison of Application Integration Technologies

Comparison of Application Integration Technologies



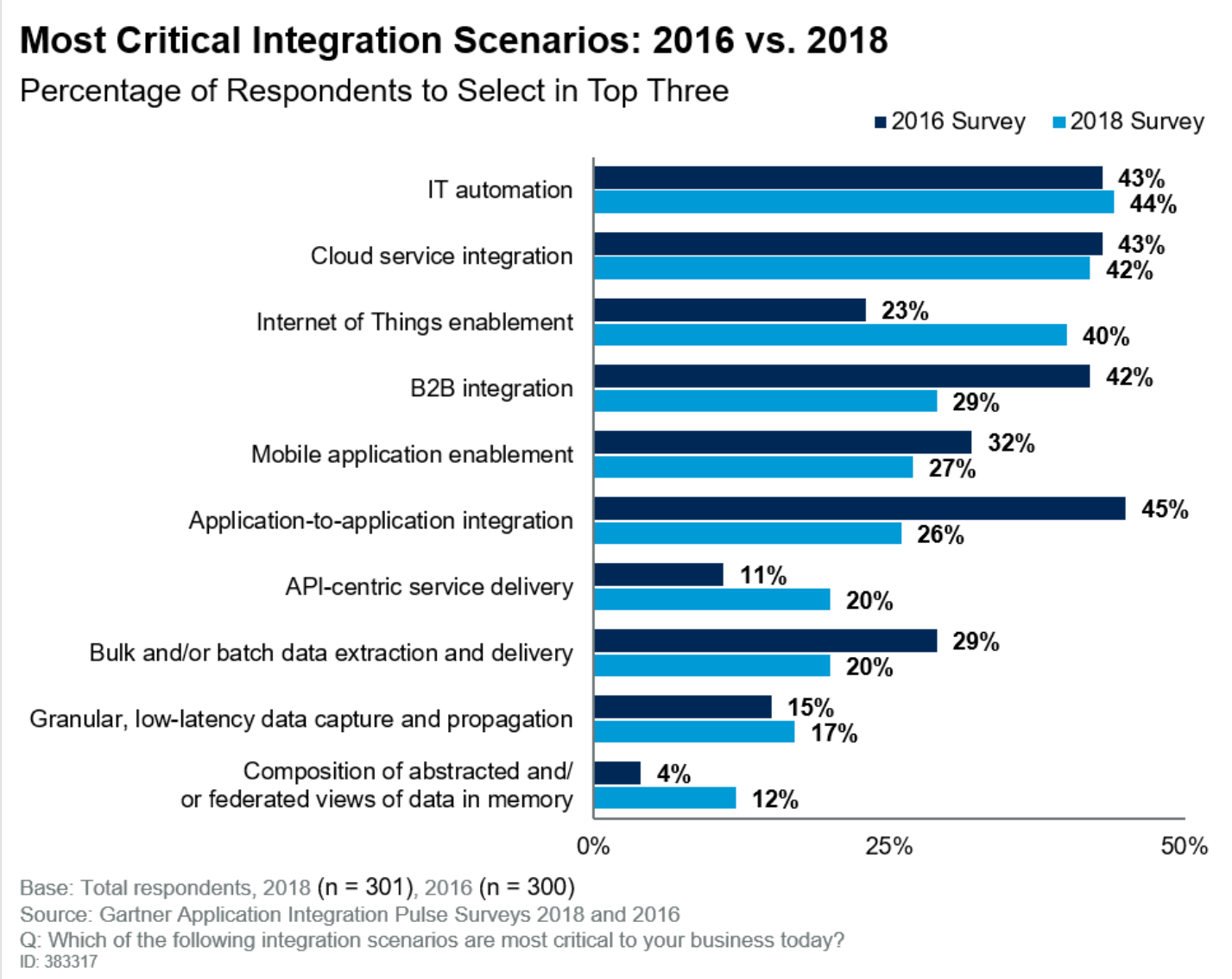
Source: Gartner
ID: 383317

Analysis

Application integration is diverse — and critical to your business. In Gartner's Application Integration Pulse Survey 2018, respondents identified their top three critical integration scenarios: IT automation, cloud service integration and IoT enablement (see Figure 3). This shows a marked increase in the importance of IoT integration and a reduction in the importance of traditional application-to-application integration. These changing integration priorities mean organizations must constantly reassess and add to their integration capabilities. For example, if your integration technologies consist only of ESB, extraction, transformation and loading (ETL) and managed file transfer (MFT)

capabilities, you will not be able to meet your business's needs when it wants to analyze streams of IoT data.

Figure 3. Business-Critical Integration Scenarios in 2016 and 2018



As integration work becomes more pervasive, it becomes more diverse, and the technology platforms and tools to deliver these requirements become more sophisticated and capable. In many ways, they become more similar; basic technical features are shared across product categories. However, significant differences remain, and those differences will dictate the long-term health and success of your integration environment.

These technologies target different constituencies and use cases. This, in turn, drives the differentiation. As a result, the selection process must assess technical capabilities as well as productivity and usability from the perspective of the integrators and operators who will use these technologies.

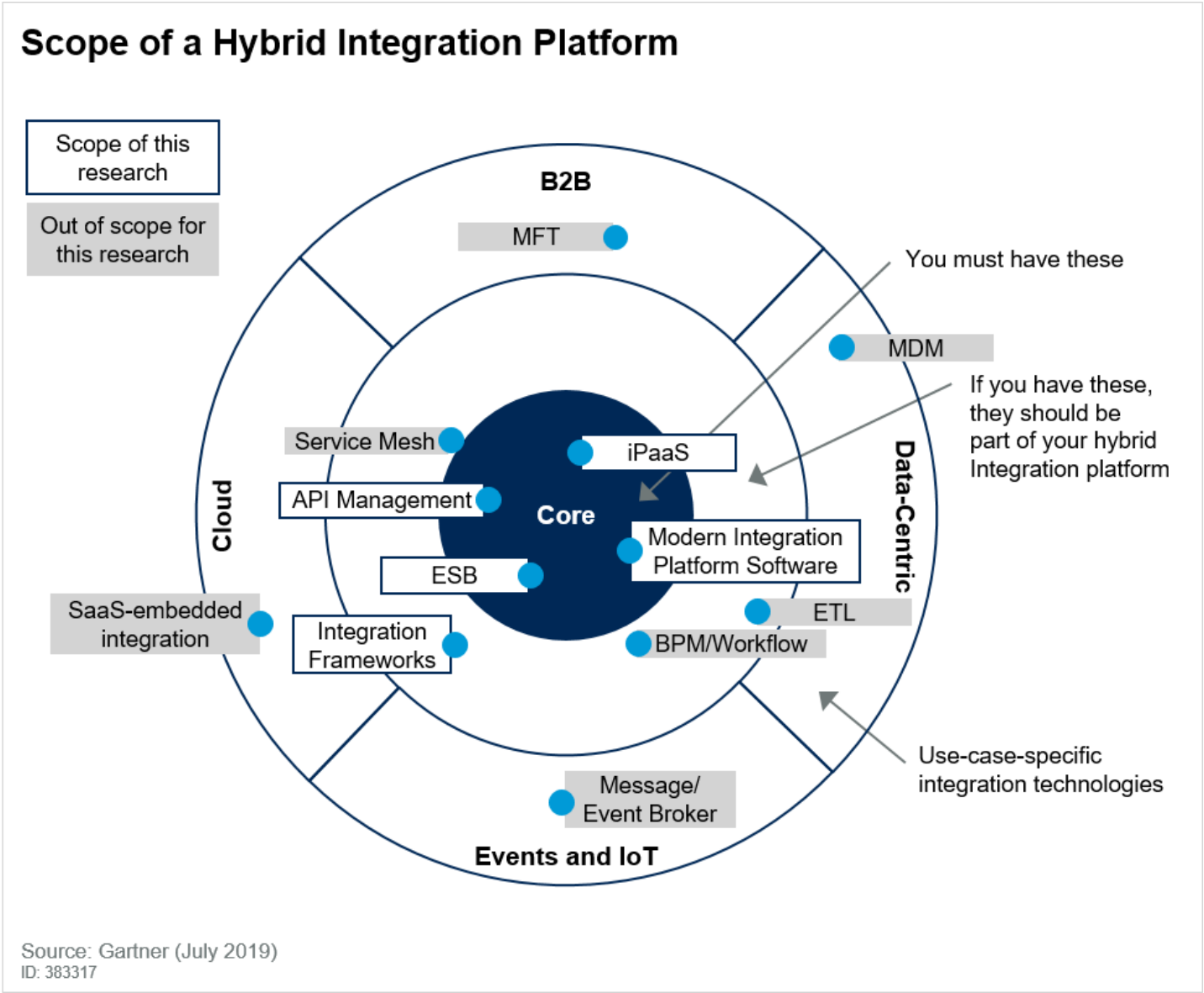
A Hybrid Integration Platform Is a Box of Tools, Not a Multitool

Gartner's definition of a hybrid integration platform (HIP) is not a single-sourced technology product. It is the combination of technology capabilities needed to support all of the integration requirements within an enterprise in the most effective manner. The full scope of a HIP extends beyond application integration technologies to include technologies for:

- **B2B integration,** for example, managed file transfer of electronic data interchange (EDI) documents with business partners
- **Data-centric integration**, for example, ETL, data virtualization and master data management tools
- **Cloud integration,** such as iPaaS and the embedded integration capabilities of SaaS software
- **Event integration and IoT,** for example, event bus and stream processing technologies.

Figure 4 outlines the scope of a hybrid integration platform.

Figure 4. Scope of a Hybrid Integration Platform



For this research, we have limited our scope to just those integration technologies that provide an application-centric approach to mediation. See Note 1 for information on the technologies we have not included in this comparison and the reasons. See "How to Implement a Hybrid Integration Platform to Tackle Pervasive Integration" for more details on the scope of HIPs and how to establish one for your organization.

Of course, establishing a HIP requires you to identify the integration capabilities your organization needs and the organizational model to support its delivery and use. Start by cataloging and documenting your existing integration technology investments. Identify those that clearly support your integration requirements and include those in the HIP, and look to replace or remove those that do not. Identify gaps in your integration capabilities, and where there is a business requirement, choose new integration technologies to fill those gaps. You may even be able to source all your required HIP capabilities from a single vendor, but do not sacrifice business requirements in doing so.

The golden rule of the HIP components is that "one size does not fit all."

Your HIP is a dynamic set of capabilities that constantly evolves over time as your integration requirements and the capabilities of your current products change. Once you have established a HIP, architects and integrators will need guidance to help them choose effectively between the technologies available. Use the strengths and weaknesses of each component of your HIP to determine when you should use it. Create your guidance based on these factors:

- **Target persona:** The skills and expectations of the people creating the integration flows will have a big impact on the platform choice. If they do not use integration tools every day, then a simple user experience with limited customization options will be easier to use (for example, from an iPaaS or distributed integration platform). If they have significant software development skills they may be more comfortable creating an integration in code using an integration framework.
- **Data vs. application-centric approach:** The traditional line between data and application integration is blurring, with many tools and techniques applying to both scenarios. However, many platforms still organize their interface and workflow around either transforming a data model or processing a request. This can have an impact on the scenarios when you should use the tool.
- **Location of the endpoints you are integrating:** Network latency, bandwidth, and cloud and internet data transfer costs can all have an impact on the most appropriate technology to use. If you are

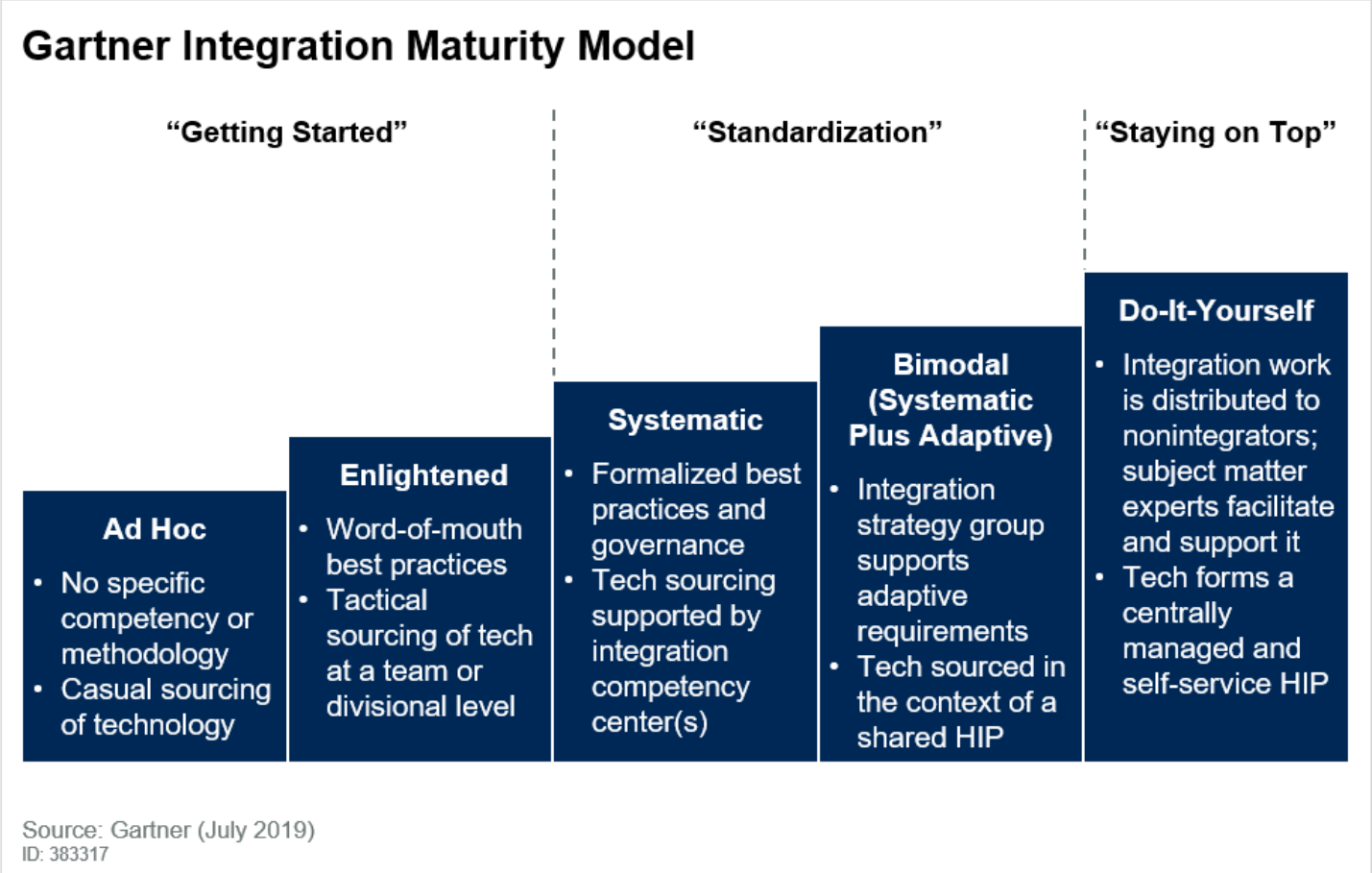
creating an integration flow between two systems hosted on-premises, then choosing a cloud-hosted platform will incur costs and perform poorly.

- **Life cycle of the integration flow:** Some integrations are coupled to the life cycle of the systems they integrate. An integration flow that connects two legacy ERP systems, both of which are due to be replaced within five years, has a lifetime of five years. If one of those systems is due to be replaced in 18 months, the integration flow has a lifetime of 18 months. When an integration flow is coupled to legacy systems, it makes sense to create it in legacy integration middleware to avoid "polluting" your modern integration platforms.

Integration Maturity Drives Technology Selection

Integration as a discipline has a checkered history in many IT organizations. Systems have been integrated in some fashion for decades, whether it be through batch file exchange, ETL, direct database connection or first-generation service-oriented architecture (SOA). Yet building maturity into how integration work is delivered and advanced often remains an afterthought because integration lives in the "white space" between application domains — a necessary evil. This leaves many organizations at a low maturity level in the Gartner Integration Maturity Model, as shown in Figure 5. For more details about the model, see "Use the Integration Maturity Model to Assess and Improve Your Integration Competency."

Figure 5. Gartner Integration Maturity Model



Note: The maturity model considers the challenge of handling “pervasive integration.” It accounts for all disciplines (spanning application, data, B2B [ecosystem] and process integration domains) and all endpoints (such as APIs, on-premises data and applications, cloud services, mobile apps, and Internet of Things devices).

Technology and maturity are not directly linked, and buying a technical solution does not guarantee a step-up in maturity.

> **Badly implemented technology can destroy your delivery models and undermine productivity. Improving maturity requires changes in the related processes, culture, skills and outcomes.**

The Comparison Criteria

The criteria chosen for this assessment reflect the most common questions Gartner receives from its clients regarding integration technology evaluation and selection. These are both technical and operational in nature. They are not solely focused on whether the technology can integrate a given COTS application with a given SaaS solution or whether it connects to an application using a specific protocol and process data in a specific format. While there are differences in these characteristics, they are vendor-product-specific, and there is often at least one way to solve an integration problem with each technology. As noted, the bigger challenge is assessing the long-term impact of selecting and deploying a given technology and the development, integration and operational practices it encourages and supports.

Implementation Criteria

The following criteria focus on the capabilities that help implement integration functionality:

- **Complex composition and orchestration of existing assets:** This criterion measures how well a technology implements and executes complex integration logic, including sophisticated data transformation, protocol mediation and orchestration of integration flows. The technology will have extensive support for integration with existing application assets, protocols and data sources. Orchestration logic can support:
  - Conditional flows
  - Looping and enumeration
  - Parallel and sequential execution (for example, fork/join and scatter/gather)

  - Exception path handling
  - Scheduling
  - Transaction management

- Processing, manipulation and transformation of inbound and outbound data can include:
  - Format translation
  - Validation
  - Filtering
  - Masking
  - Mapping
  - Aggregation
  - Support for expressions or code to define more sophisticated processing logic

The technology should also have support for long-running integration processes and flows, distributed transactions (XA) and scheduling of integration execution. These capabilities will enable the technology to support an extensive range of enterprise integration patterns.

- **API enablement and mediation:** This criterion measures how well a technology supports the creation and publication of APIs that give access to application services, which may be composed or integrated using the tool. The technology should support simple integration and composition of API-centric services, for example, by providing easy-to-use tools for filtering and mediation of payloads, particularly JSON and XML formats. The publication capabilities should support the separation of concerns between policies controlling access to underlying resources (for example, security, identity and traffic management) and service implementations, enabling you to apply consistent policy to diverse service implementations. Technologies that support managed self-service of API consumers such as app developers will also score more highly for this criterion.
- **Application (COTS) and legacy technology connectivity:** This criterion measures how well a technology supports connectivity with widely deployed commercial third-party applications (for example, ERP, CRM or HRM). For example, the technology should provide supported application adapters and integration templates for common platforms. This criterion also addresses support for legacy and proprietary technology connectors such as CICS/remote procedure call (RPC) and terminal protocols like IBM 3270. It excludes generic protocol and technology adapters such as

HTTP, SOAP, MQTT, file system and Open Database Connectivity (ODBC)/Java Database Connectivity (JDBC), which we cover in subsequent criteria.

- **Cloud application connectivity:** This criterion measures how well a technology supports integration with leading SaaS application and modern application protocols. For example, technologies should provide supported application adapters and integration templates for commonly used SaaS applications and may offer a marketplace and partner ecosystem for additional adapters. The maintenance and update cycles of these cloud application adapters should closely follow the life cycles of the applications themselves. This criterion also assesses support for widely used modern protocol and technology adapters such as HTTP(S), OpenAPI Specification (formerly Swagger Specification), GraphQL, SOAP, gRPC, JDBC/ODBC, AMQP, MQTT, FTP.

- **Democratization of integration:** This criterion measures how well a technology supports an environment that is easy to learn and use productively without the need for job specialization. This can help to release resource pressure on constrained resources in dedicated integration teams (for example, as part of an integration competency center). By allowing ad hoc integrators self-service access to the tools to solve integration problems within their domains on their own timelines, this model also supports a more responsive and adaptive approach to integration delivery.

- **Stream-based integration**: This criterion measures how well a technology supports integrating endpoints that send or receive streams of events or data (such as readings from IoT sensors), rather than requests or individual messages. Integrating a stream requires that data or events are ingested at high speed, to avoid slowing down producing systems, while using back pressure from consuming systems to control delivery rates. Mediating a stream requires the ability to perform operations, such as filtering, transforming and enriching components within a stream, or combining multiple streams together. More advanced features include the ability to create streams from endpoints that don't expose them or convert streams into discrete messages or requests. This approach to integration is important when working with IoT or streaming analytics technologies.

Operational Criteria

The following criteria focus on the capabilities that help to operate integration functionality:

- **Deployment flexibility and control:** This criterion measures how well a technology supports a flexible and adaptable deployment model that is easy to change over time and can, if necessary, be controlled by the team delivering integration work. This flexibility should allow the deployment of the technology to adapt with changing functional and operational demands in a controlled way. Technologies that typically result in prescriptive deployment models will not score as well in this criterion.

- **Support for dynamic runtime demand:** This criterion measures how well a technology allows you to scale capacity up and down to meet cyclical or unpredictable demand. In simple terms, they should allow you to provision capacity for the troughs and scale for the peaks in demand. This requires both technical and commercial models that support fluctuating workloads or on-demand provisioning of capacity to support new workloads.

- **Integration governance and visibility:** This criterion measures how well a technology enables an organization to maintain appropriate control and visibility over the system and application dependencies that are implemented via integration logic and application services. This control should ideally include management of the entire life cycle of services and integrations, including design time and runtime. Design time governance capabilities should allow integration work to be distributed or delegated to appropriate resources without a loss of control. Runtime governance capabilities should allow visibility into the deployed and executing integrations and the dependencies they establish. The technology should also support audit requirements as they pertain to changes in the integrations and deployment of the technology itself.

- **Integration operations management:** Technologies that score well for this criterion will have strong support for the day-to-day runtime management and operation of a portfolio or catalog of integrated services. The technology should support consistent operational practices for both centralized and distributed teams administering and monitoring the health of the technology platform and the integration workloads deployed to it. This includes managing centrally deployed, clustered instances and, where appropriate, distributed deployments with multiple runtime environments. This criterion also takes into account the typical impact of upgrades to the technology and support for the migration of workloads between versions.

- **Security and privacy control:** Technologies that score well in this criterion enable an organization to enforce and manage its security and compliance policy associated with application services and integration. This includes complying with policies relating to both application data and metadata processing and persistence — for example, ensuring that application data is processed and persisted (even transiently) only in approved deployment locations or environments. This criterion also extends to metadata required by integration flows and generated by them, including integration configuration data (for example, service endpoint and credentials, algorithms forming part of an integration) and logging data generated during execution.

The Technologies Assessed

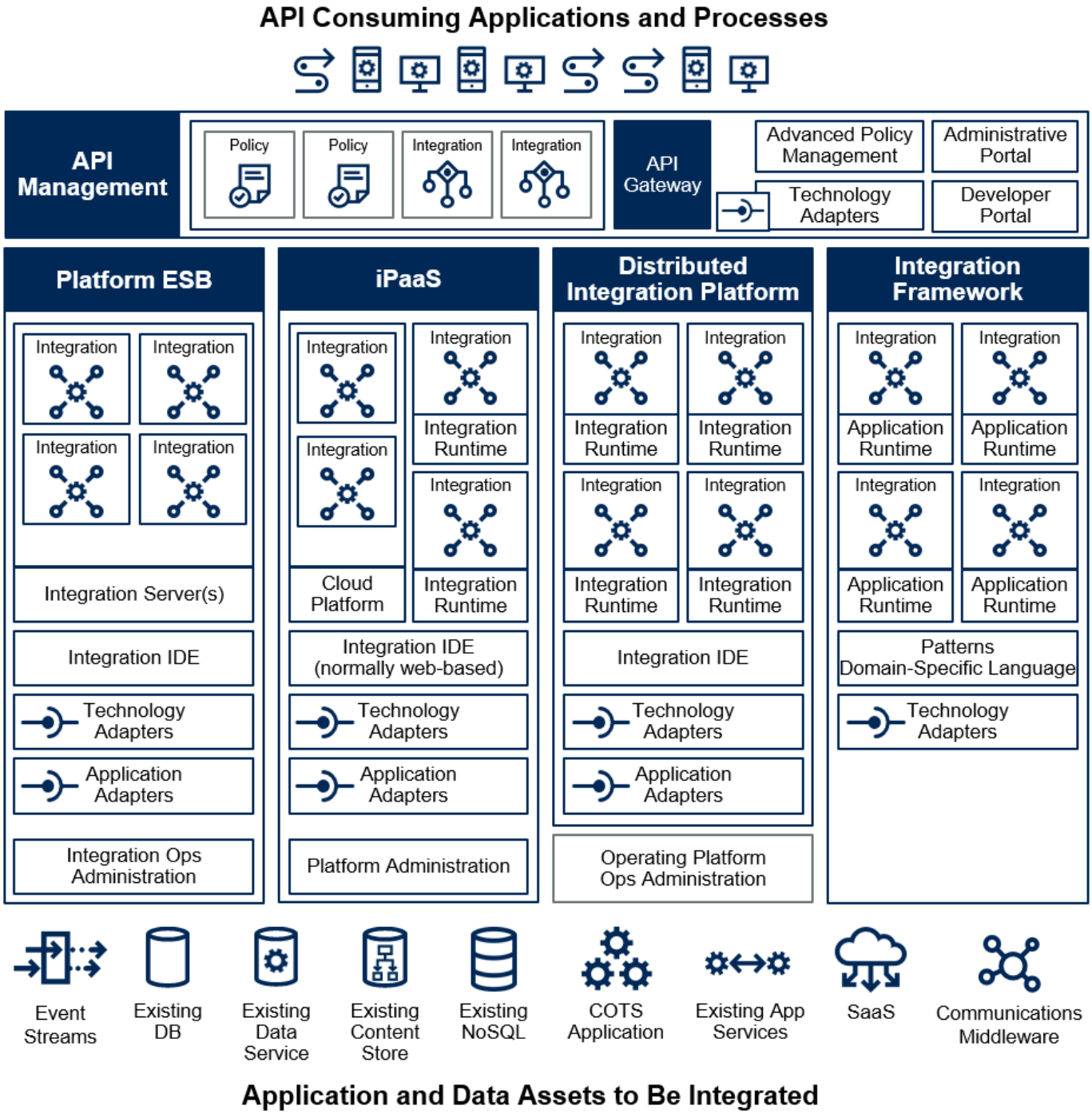
The technologies assessed in this research represent a majority of the application-integration-related inquiries from Gartner's clients and a significant amount of IT spending. In most organizations, there is at least one (and often many) incumbent technology, and questions arise about longevity, suitability for new requirements and how the technology compares with alternate products, often in other categories.

In the context of a HIP, these technology categories can be complementary. In particular, organizations that are maturing their SOA delivery should be more capability-centric (see the Integration Maturity Drives Technology Selection section). You should create a HIP that offers API management, agile integration and complex integration capabilities. There may be additional capabilities that you require in your HIP that we do not cover in this research. Note 1 describes some of these technologies.

Figure 6 shows a high-level comparison of the capabilities and alignment of the five technologies covered in this assessment. Subsequent sections provide logical architecture diagrams for each of those technologies.

Figure 6. Comparative View of the Assessed Integration Technologies

Comparative View of the Assessed Integration Technologies



Source: Gartner (July 2019)
ID: 383317

The nature of the application integration marketplace and the varied origins of the products available mean there is overlap in technical capabilities. Additionally, some vendor offerings fall into more than one category based on how you configure, deploy and use them. For example, IBM, MuleSoft and Red Hat have ESB offerings that support both deployment models. Table 1 shows a sample of representative products mapped to these technology categories.

Table 1: Representative Sample of Vendors and Products in Each of the Technology Categories

Traditional Platform Enterprise Service Bus	Integration Platform as a Service	Distributed Integration Platform	Integration Frameworks	API Management
■ IBM Integration Bus ■ Microsoft BizTalk Server ■ Oracle Service Bus ■ Red Hat Fuse (Fabric) ■ SAP NetWeaver Process Integration ■ Software AG webMethods Integration Platform ■ TIBCO BusinessWorks ■ WSO2 Enterprise Service Bus (carbon cluster)	■ Dell Boomi ■ Informatica Intelligent Cloud Services ■ Jitterbit ■ Microsoft Logic Apps ■ MuleSoft Anypoint Online ■ Oracle Integration Cloud Service ■ Red Hat Fuse Online ■ SnapLogic	■ IBM App Connect Enterprise ■ MuleSoft Anypoint (Mule Runtime Engine) ■ Red Hat Fuse (OpenShift) ■ TIBCO BusinessWorks Container Edition	■ Apache Camel (Java) ■ Mule Embedded Runtime (Java) ■ Pivotal's Spring Integration (Java) ■ NServiceBus (.NET) ■ Akka Alpakka (Java, Scala) ■ TIBCO Project Flogo (Go) ■ WSO2 (Ballerina)	■ Google's Apigee Edge ■ Axway AMPLIFY API Management ■ Broadcom-CA API Management ■ IBM API Connect ■ MuleSoft Anypoint API Manager ■ Red Hat 3scale ■ TIBCO Cloud Mashery

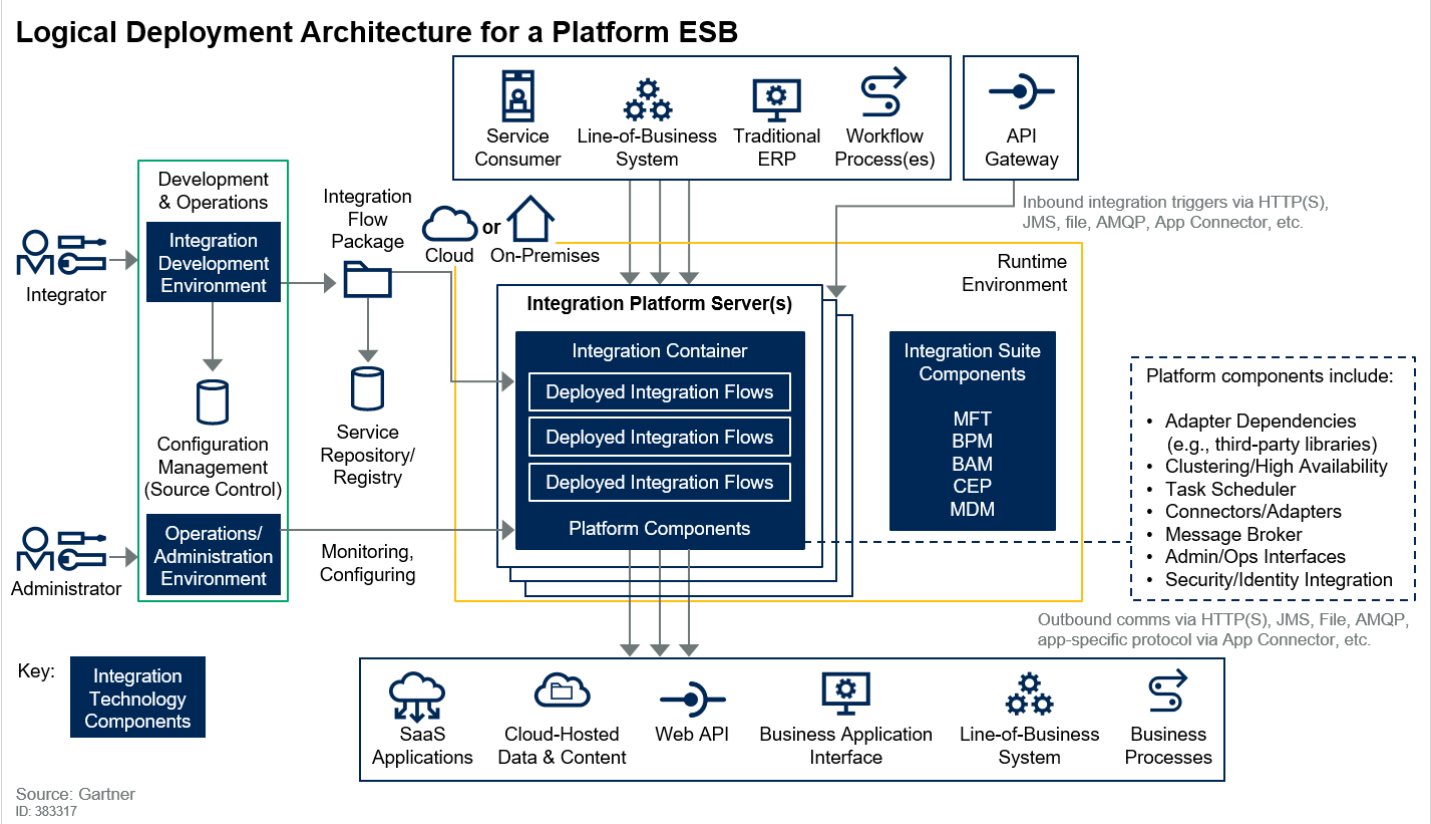
Source: Gartner (July 2019)

Traditional Platform ESB

The traditional ESB, installed as software, is the integration technology that most organizations are familiar with. You probably already have one (if not more) ESBs in use, and so when you choose an integration technology to use, the ESB is the baseline you compare against. The traditional model for deploying enterprise service buses and related integration suites is to build a highly available cluster of servers. This creates a platform into which you deploy integration workloads for execution and management. This deployment model helps to centralize and standardize the delivery of integration work and operation of the integration environment, which can, in turn, drive efficiencies and reduce diversity and complexity in the integration environment.

Figure 7 shows a logical architecture for this type of technology.

Figure 7. Logical Deployment Architecture for a Platform ESB



In this deployment model, distinct integration flows share capacity. The ESB provides monitoring and management tools that give insight into the health of the overall integration environment, as well as the health and state of specific integration processes that map to business activity.

The technologies that support this model of deployment include a wide spectrum of integration capabilities, including:

- Connectors for common COTS and SaaS applications
- Modern and legacy technology connectors
- A range of industry and marketplace standard file, message and data formats
- Proprietary protocol connectors for enterprise software products offered by the vendor

The cost and complexity of establishing the platform demands that it must fulfill a wide variety of integration requirements. Therefore, it is typically managed and used by “specialist integrator” team members with deep knowledge and experience of the platform.

Examples of products that support the platform ESB deployment model include:

- IBM Integration Bus
- Microsoft BizTalk Server
- Oracle Service Bus
- Red Hat Fuse (Fabric)
- Software AG webMethods Integration Platform
- TIBCO ActiveMatrix BusinessWorks
- WSO2 Enterprise Service Bus (using carbon cluster)

Strengths

Strengths of platform ESB solutions include:

- **The widest set of application integration capabilities:** These are bundled into a single platform that can fulfill a wide variety of integration scenarios. These capabilities include:
  - Application connectors for common COTS solutions supported by the platform vendor and/or vendor partners
  - Sophisticated orchestration, transformation and mediation capabilities
  - Technology adapters — wide-ranging protocol adapter/connector support including HTTP, FTP, messaging, file system and Internet Message Access Protocol (IMAP)/SMTP
  - Support for asynchronous integration patterns, often supported by a built-in message broker
  - Support for long-running orchestration processes and process state persistence
  - Support for creating and sharing common integration patterns or templates for reuse
  - Visual integration design environments to help simplify integration development
- **High availability at the integration platform level:** This means that all integration flows deployed into the platform benefit from the investment in appropriate infrastructure, configuration and operations. In some products, this high availability is a native capability to the ESB software; for example, MuleSoft Mule Runtime High Availability (HA) Clusters. In other cases, the ESB leverages underlying middleware clustering and HA capabilities (for example, Oracle Service Bus).
- **Mature technology that incorporates operational monitoring and management tools:** These tools are needed to ensure robust operation of business-critical integration workloads common in

systematic “Mode 1” delivery models, where you value predictability and control over agility. These technologies are proven in highly demanding scenarios, integrating hundreds of endpoints and processing tens of millions of integration processes per day.

- **Extended integration suites that provide additional features:** These features support, build on or extend the application integration capabilities. These integration suites include capabilities such as managed file transfer, business process management (BPM), business activity monitoring (BAM), complex-event processing (CEP) and master data management (MDM).

Weaknesses

Weaknesses of platform ESB solutions include:

- **Deploying a platform ESB typically results in a technically complex environment:** This approach requires platform-specific expertise to deploy, configure and manage over time. This model is monolithic in nature such that many operational decisions, such as upgrades and patching, need to be taken for the entire platform and all the workloads it is hosting. The sophistication of these tools also means that runtime capacity typically scales out in server-size increments — both for infrastructure and licensing.

- **The complexity of platform ESB environments means that specifically skilled or trained resources are needed to use and operate them effectively:** This often leads to resource constraints for integration work and change management because the number of these resources is limited. The situation is exacerbated by the fact that these skills command a premium in the marketplace.

- **Platform investments are typically in the form of “license plus maintenance” or “per core per year support subscriptions”:** This means you must license your environment for the peak amount of deployed processing capacity. The cost per unit can also be significant. For predictable, systematic integration workloads, this may be manageable, but it can be challenging for adaptive “Mode 2” integration work, where demands and capacity can increase or decrease rapidly. Also, there is the hidden risk and cost of embedding or combining application logic and code into integration flows. Not only is this bad practice that degrades maintainability and flexibility, it incurs the cost of running that logic on a much more costly compute platform (i.e., consider your ESB per core/container pricing versus the cost of your runtime stack, which may well be free).

- **Extended integration suite features can bloat the platform:** Many ESB products are delivered as part of an integration suite that might include MFT, BPM, BAM, CEP, and MDM. Arbitrarily adopting these suite components can lead to platform bloat, overbuying of capability and complexity in deployment and operations.

- **Lack of support for cloud application integration:** Most ESB products provide few application-specific connectors for SaaS applications. This means integrators will have to revert to using

technology connectors to integrate manually with SaaS web APIs. This approach is suitable for specialist integrators (the target user of an ESB) but not for ad hoc integrators.

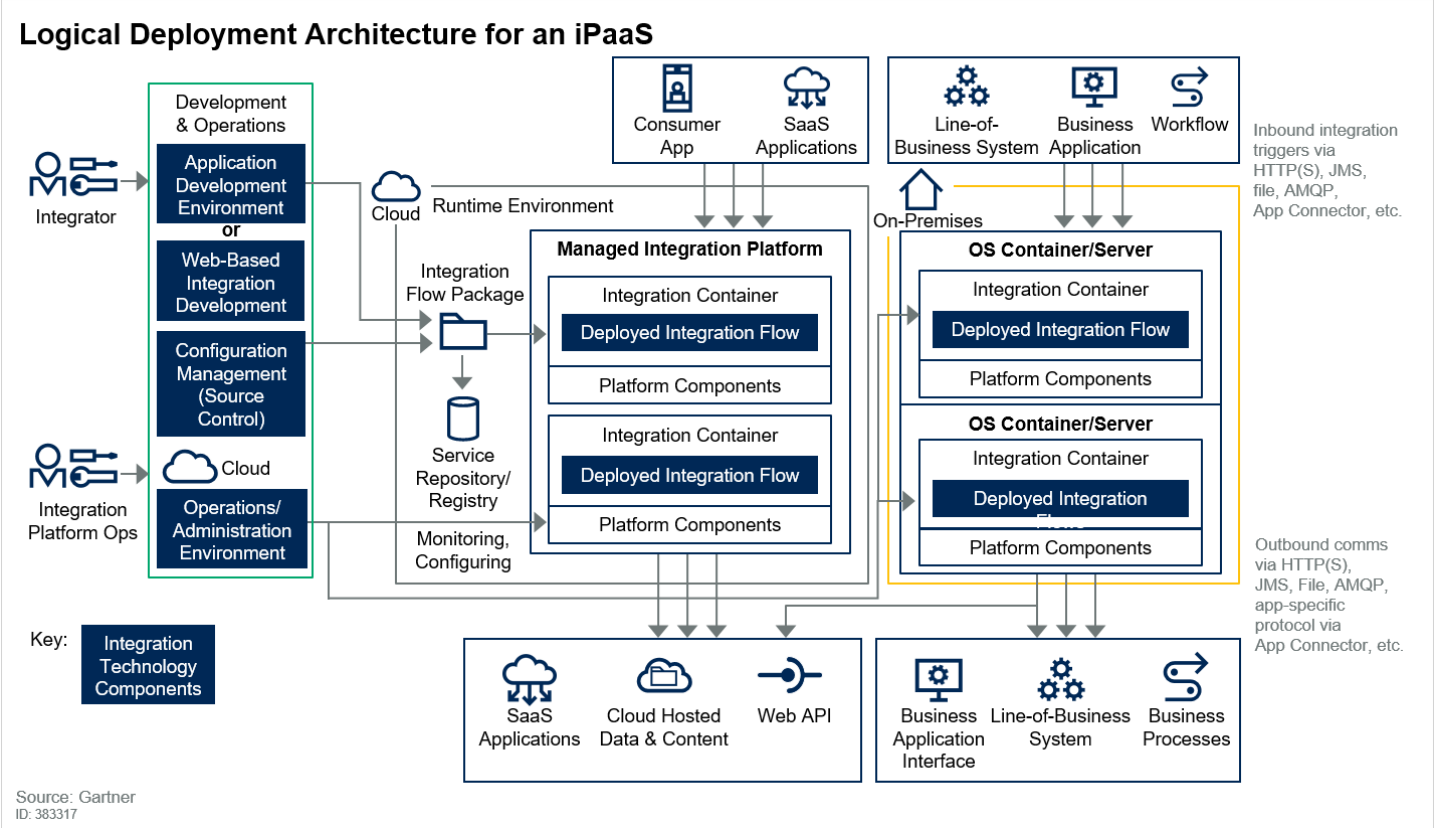
- **Weak support for streaming integration:** The recent popularity of stream processing postdates most ESB platform software. While ESBs often have good support for message-based integration, they can only deal with individual messages. Some vendors, such as IBM and Oracle offer separate stream processing platforms that you can integrate with your ESB.

iPaaS

Integration platform as a service technologies deliver integration capabilities using a vendor-managed and provisioned model. This must include the ability to deploy and operate integration workloads in a provider-managed environment that abstracts the customer from any underlying platform or infrastructure management. However, in the case of leading products, there is the option to deploy integration runtimes to customer-managed infrastructure (including IaaS). These runtimes are connected to the iPaaS cloud environment to provide centralized management, integration deployment and monitoring.

Due to their cloud-based nature, iPaaS products offer browser-based administration tools and often include browser-based integration development tools. iPaaS products target use cases like cloud application integration where time to delivery is critical and will often focus on both ease of access and ease of use for their integration tooling. Most iPaaS products include development life cycle features such as source control, test environments and deployment tools in the platform, rather than requiring you to provide your own. Figure 8 shows the logical deployment architecture for a typical iPaaS product.

Figure 8. Logical Deployment Architecture for an iPaaS



The reduced provisioning cycle and ease of use offered by iPaaS combine to support scenarios where time to market is critical. The deployment model eliminates or reduces infrastructure provisioning and configuration time, while the ease of use can increase both productivity and the pool of resources with adequate technical skills to solve the business problem. For many organizations, their first exposure and demand for iPaaS is in a cloud-to-cloud or cloud-to-ground integration scenario triggered by the implementation of one or more SaaS solutions.

Leading iPaaS solutions now have a spectrum of capabilities that can make them appealing for a more extensive set of integration requirements, including B2B and on-premises “ground-to-ground” scenarios. Connectivity to on-premises applications is supported by runtime software agents that customers can deploy to their own infrastructure. These agents connect to the iPaaS control plane, which deploys the integration flow and controls the execution of the integration logic.

For more details of the iPaaS vendor market, see “Magic Quadrant for Enterprise Integration Platform as a Service.”

Examples of products in the category include:

- Dell Boomi
- Informatica Intelligent Cloud Services
- Jitterbit

- Microsoft Logic Apps
- MuleSoft Anypoint Online
- Oracle Integration
- Red Hat Fuse Online
- SnapLogic

Strengths

Strengths of iPaaS solutions include:

- **Deep support for cloud service integration (CSI) from both integration and operational perspectives:** Integration with SaaS applications and common web APIs is supported through catalogs of application adapters for common business applications and services. Some products (including Informatica Intelligent Cloud Services, Jitterbit and SnapLogic) extend to data integration capabilities not found in the other technologies compared.
- **The cloud service delivery model supports fully managed hosting of integration workloads and connection- or consumption-based pricing models:** Using an iPaaS frees you from performing platform maintenance tasks, such as applying patches and performing upgrades. It also supports centralized management and monitoring of integration workloads deployed to both cloud-hosted and customer-managed (on-premises or IaaS) hosting environments.
- **The vendor-hosted execution model supports cloud-to-cloud integration without requiring "repatriation" of data or processes to the enterprise infrastructure:** When repatriation is desirable, leading iPaaS solutions offer cloud-managed, on-premises execution models. Having the ability to execute integration flows on-premises while managing them from the cloud provides more flexible deployment and control over the flow of transactional data through third-party platforms.
- **Web-based visual integration design tools help simplify integration tasks:** Simplified orchestration, transformation and mediation capabilities make the products easier to learn and use. Also, the multitenant cloud model means that products like Dell Boomi and Oracle Integration can offer "recommendation engines." Recommendation engines provide guidance and suggested mappings for common integration use cases based on analysis of the deployed integrations of other customers. Such recommendations are made possible because iPaaS vendors can maintain and expand a knowledge base of all deployed integration pipelines. This supports adaptive, iterative integration work in bimodal or digital business scenarios.
- **Capacity, scale and deployment decisions can be managed on a "per integration" basis:** This reduces change inertia and isolates service implementations. Vendors like Dell Boomi also offer subtenancy capabilities within their platforms, allowing central control while integration and

operation work can be delegated to tenant teams using the platform on a day-to-day basis. This can help integration competency center teams become facilitators of integration work and focus on high-value requirements (see the Gartner Integration Maturity Model in Figure 4).

Weaknesses

Weaknesses of iPaaS solutions include:

- **The integration capabilities in iPaaS solutions are frequently less sophisticated than their ESB software "cousins":** This is due to a focus on ease of use and the need to have a cloud-managed deployment model. This means that integration workloads with complex orchestration requirements, complex transformation, mediation and support for legacy protocols and payloads may not be appropriate for an iPaaS.
- **Integration metadata and operational data stored and processed in the provider's cloud platform:** This can include integration definitions, configuration, credentials, and runtime logging and exception data. This can be the case even when you execute the integration workloads on-premises. When integrating and processing sensitive data, you must take this into account and assess these aspects of the iPaaS platform you plan to use against your governance, risk and compliance (GRC) requirements.
- **iPaaS products tend to have limited support for COTS application connectors and legacy protocols and formats:** Integration with these products may still be feasible using native software development kits (SDKs), libraries and APIs. However, the complexity of both developing and supporting the integration increases.
- **Cost and consumption monitoring must become part of your operational processes:** When purchasing iPaaS using a consumption-based or connection-based pricing model, unanticipated growth in usage and throughput will directly impact your cost of operation.
- **The on-premises integration runtime environments for leading iPaaS vendors must be predeployed and configured:** Although this is often easier and less complicated than traditional platform ESB deployments, it can lead to overprovisioning of capacity. It can also lead to contention for compute resources between workloads deployed to the same on-premises environment.

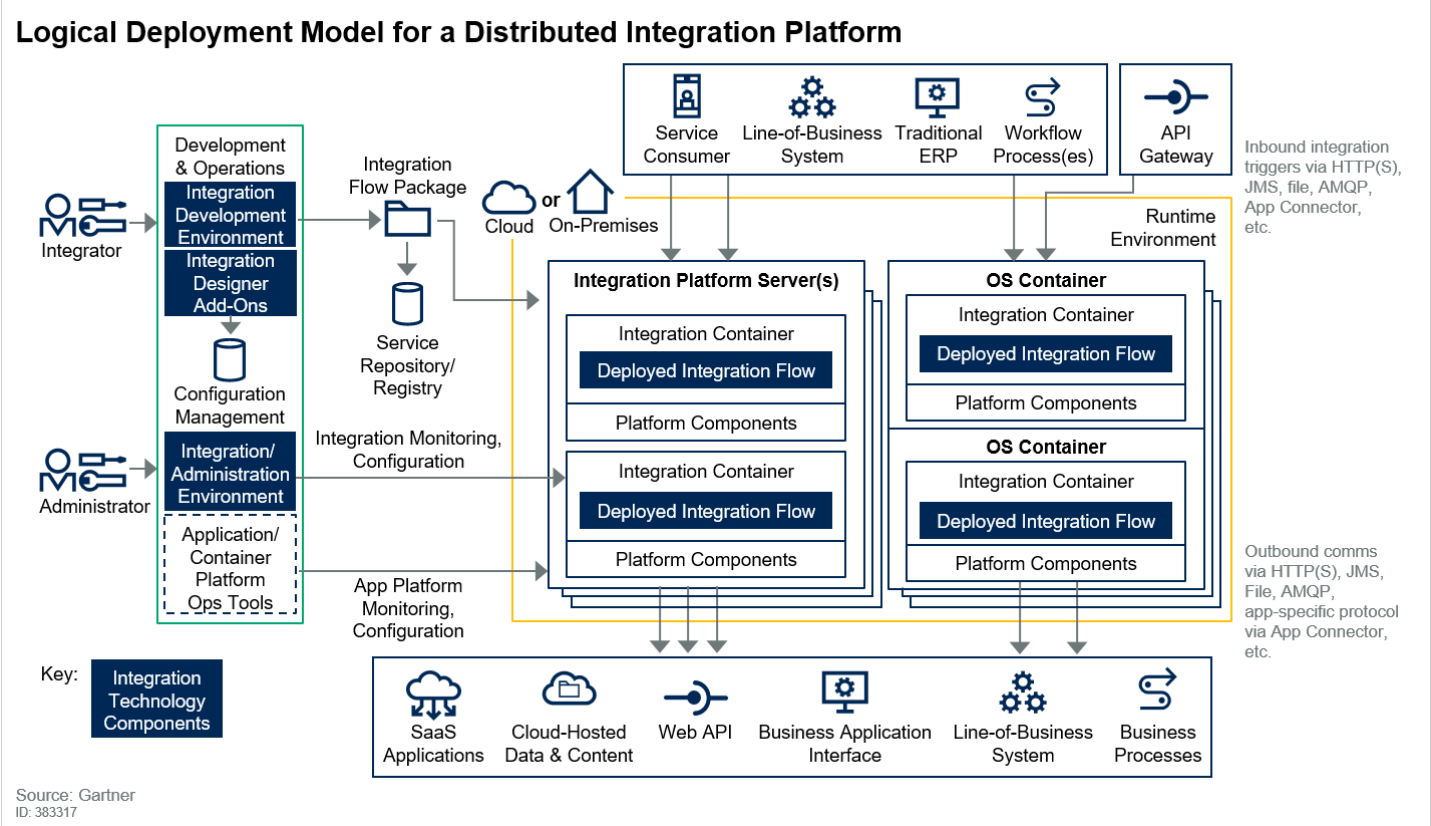
Distributed Integration Platform

Integration platform software that you deploy and manage yourself has moved beyond the traditional ESB model. Many of the features that were originally confined to an iPaaS, such as a web-based integration designer aimed at citizen integrators, are now available as software that you run on-premises. In many cases (such as MuleSoft Anypoint and Red Hat Fuse) the software you install and manage yourself is the same software the vendor uses to offer its iPaaS. In a DIP, the integration

runtime components and a related set of integration flows are packaged and deployed together; each integration is packaged in a container with its own runtime. Products such as IBM App Connect, MuleSoft Anypoint Platform and Red Hat Fuse offer flexibility in their deployment model to support both traditional platform ESB and distributed models. Others, like TIBCO BusinessWorks Container Edition, support only this "distributed" deployment model.

In a DIP integration, workloads are loosely coupled both at design and deployment time. They can also be operated, scaled and updated in isolation from other integration workloads. One of the drivers for this model of deployment is to align the delivery of integration workloads with application development and deployment life cycles, tooling and operating environments. Much like the microservices architecture model, distributing the deployment of your ESB might simplify an individual workload and its runtime stack, but this moves complexity to the underlying "outer architecture." This model should not be assumed to be easier to manage as a whole. However, if you have addressed the complexity of running loosely coupled application workloads, this model can help take full advantage of public- or private cloud-native infrastructure and platforms, leveraging the elasticity and automation capabilities they provide. Figure 9 shows the logical deployment architecture for a typical modern integration software deployment.

Figure 9. Logical Deployment Architecture for a Distributed Integration Platform



For further details about the DIP model, see "Enabling Agile integration With a Distributed Integration Platform."

Examples of DIPs include:

- IBM App Connect
- MuleSoft Anypoint
- Red Hat Fuse (OpenShift)
- TIBCO BusinessWorks Container Edition

Strengths

Strengths of DIP include:

- **Web-based visual integration design tools help simplify integration tasks:** Web-based graphical development environments are not only available in an iPaaS. DIP can offer a similar user experience aimed at ad hoc integrators. Simplified orchestration, transformation and mediation capabilities make the products easier to learn and use.
- **This approach aligns with modern application development and delivery processes and platforms:** This includes dependency management, continuous integration (CI)/continuous delivery (CD), version control and PaaS or container orchestration. This approach allows you to leverage experiences and investments across application and integration delivery cycles. As integration requirements become ever more pervasive, this will provide more consistency and agility when addressing adaptive Mode 2 business demands.
- **Distribution and isolation of individual integration workload deployments provide more flexibility:** This is true for resource consumption, change management and infrastructure life cycles. Capacity, scale and deployment decisions can be managed on a “per integration package” basis, reducing change inertia and isolating service implementations.
- **This model supports the adoption of new application platform practices:** This includes immutable infrastructure and deployment automation. DIP can help take full advantage of public or private cloud-native infrastructure and platforms, leveraging the elasticity and automation capabilities they provide.

Weaknesses

Weaknesses of DIP include:

- **Commercial licensing or support models for these technologies is still typically an upfront commitment to a specific number of cores or container instances:** Product technology exists to support very flexible deployment and scalability models. However, the commercial licensing model may reduce the real-world flexibility of deployment, even when the technology supports it. In addition, the hidden risk and cost of embedding or combining application logic and code into integration flows remain. You must balance the convenience of embedding that logic with the cost

of running that logic on a more costly platform (that is, consider your ESB per core/container pricing versus the cost of your runtime stack, which may well be free).

- **This deployment model is still evolving:** This means expertise in the marketplace for deploying specific technologies in this configuration is limited. It also means the technical and performance limitations of this model versus the platform ESB are not fully understood. Vendor support may be required to fully understand the feature and function constraints of vendor products and to define the mitigating technology and process. In addition, the platform and tools for managing OS containers such as Docker are still immature, add a new layer of complexity that you must manage and continue to evolve rapidly.
- **Some capabilities supported in platform ESB deployment are not available or must be deployed and managed externally from the packaging and deployment of integration workloads:** For example, long-running, stateful integration processes will require an external persistence mechanism, such as a database or in-memory data grid that the vendor supports. Also, asynchronous integration patterns require an external message broker to provide an asynchronous messaging transport.

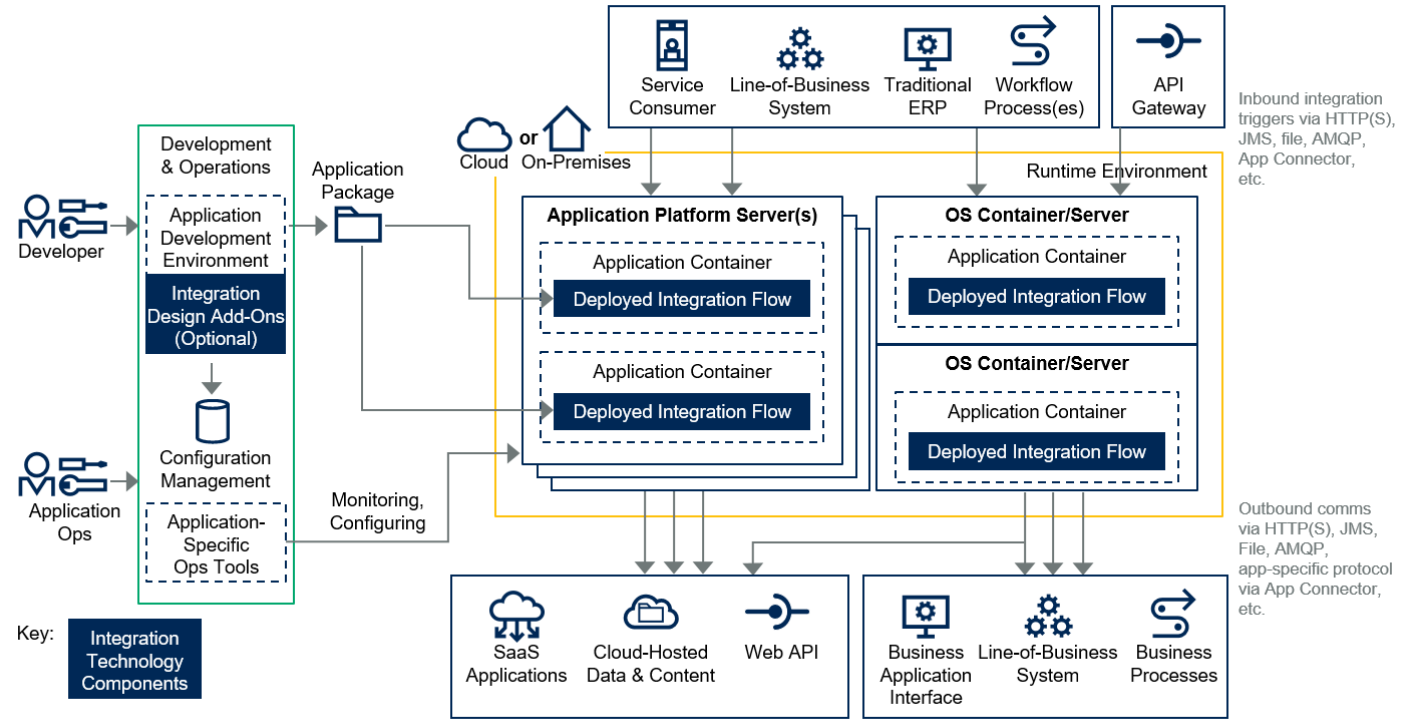
Integration Frameworks

Integration frameworks are development frameworks or libraries that provide tools for developing and embedding integration flows in your application workloads or building them as stand-alone service implementations. The framework itself will provide a code library and/or support for a domain-specific language that simplifies the implementation of common integration patterns and provides adapters to common technologies and protocols.

Many vendors use integration frameworks as the heart of the integration runtime in the other technology categories. Most of the integration frameworks are open source and freely available to developers to integrate into their own codebases. However, you must assess whether the developer support from communities and/or commercial providers is adequate for your needs when evaluating integration frameworks. Integration frameworks often make their way into your integration tool portfolio via the back door as gray infrastructure. Developers who do not feel the integration middleware platforms meet their needs will embed an integration framework into their codebase. Figure 10 shows the deployment model for an integration framework. However, because these tools really are just frameworks, the deployment model will be largely dictated by the architecture of the application you embed it into.

Figure 10. Logical Deployment Architecture for an Integration Framework

Logical Deployment Model for an Integration Framework



Examples of integration frameworks include:

- Apache Camel
- Spring Integration
- Mule Community Embedded Runtime
- Particular Software NServiceBus
- Akka Alpakka
- Project Flogo
- WSO2 Ballerina

Strengths

Strengths of integration frameworks include:

- **Integration implementation is treated as "code"**: These are lightweight, developer-oriented tools that help to streamline the implementation of common integration patterns. As frameworks, they have very few environmental prerequisites, other than the language runtime itself, and consume fewer resources than the alternatives. Many of these frameworks are easy to learn for experienced developers and often have large communities providing a basic support channel. Treating

integration as code means you can take advantage of existing investments in configuration control and continuous build and delivery infrastructure for your integration flows.

- **Proven as components of commercial integration technologies:** Many of these frameworks are widely used and proven as part of commercial integration products (for example, Mule within MuleSoft Anypoint and Apache Camel within Red Hat Fuse and Talend Application Integration). With careful design, these product relationships can offer a migration path from building integrations using the framework to using a more powerful integration platform.

- **The delivery model is mostly free open source:** This includes community and/or third-party commercial support (for example, Apache Camel, Spring Integration). Some products are available with per-developer or per-compute-node licensing and support (for example, NServiceBus).

- **You are in full control:** Integration frameworks give you the full power of the programming languages they are implemented in. This allows you to implement very complex business logic or connect using custom protocols that would otherwise not be possible to implement. This flexibility allows you to create integration flows that match emerging trends (such as streaming integration or agile integration) without having to wait for your vendor to provide the capability. You have flexibility regarding deployment models, for example, whether you run your integration flows inside an application server, a container or even a function platform as a service.

Weaknesses

Weaknesses of integration frameworks include:

- **Integration frameworks provide no tools for management or governance to help manage the delivery or operation of your integration workloads:** This runs the risk of integration workloads being fragmented and distributed in a way that makes application and system dependencies harder to understand and manage. Embedding integration logic into existing applications can lead to point-to-point integration and complex dependencies over time.

  - You must apply any implementation governance and control through your architecture, design and development processes, and you must implement monitoring or management capability at the application or runtime level.

- **Frameworks are development-language-specific (e.g., Java-specific):** The frameworks have limited support for visual design tools, making the learning curve to productivity steeper and very specific to a given framework.

- **Version management (of the framework and all connectors) is the responsibility of the developer:** Integration frameworks can have a high number of external dependencies. For example, Apache

Camel 2.17.0 has 45 added, seven dropped, and 114 changed dependencies based on a comparison of the package dependency files.

- **The number of technology adapters available varies widely from framework to framework:** Some, like Apache Camel, have a vast library while others, like MassTransit or NServiceBus, focus less on integrating with external endpoints and have more limited connectors. Even the deepest adapter libraries have limited support for SaaS application connectors (Salesforce is the most common) and even more limited or nonexistent support for COTS application connectors and legacy protocols and formats.

API Management

API management technologies have three core components.

At the heart of the solution is an API gateway that is responsible for enforcing your API access policies. There are two types of API gateway deployment. You deploy enterprise or edge gateways as a central shared service, and expose many APIs via the same gateway. Use this model to protect your APIs that you expose over public network infrastructure. The second model is microgateways, where you deploy the gateway on or near the servers that are running your services, with each gateway hosting one or a small number of services. With the increasing popularity of service mesh technology, and its capabilities to control service-to-service communication, the microgateway deployment model is becoming less common, with enterprises using service meshes instead. (See "Assessing Service Mesh for Use in Microservices Architectures.") For both of these models, the API gateway component can be installed in multiple ways, including cloud-hosted services, physical or virtual appliances, or installed software. A single API management deployment can include multiple gateways under common control.

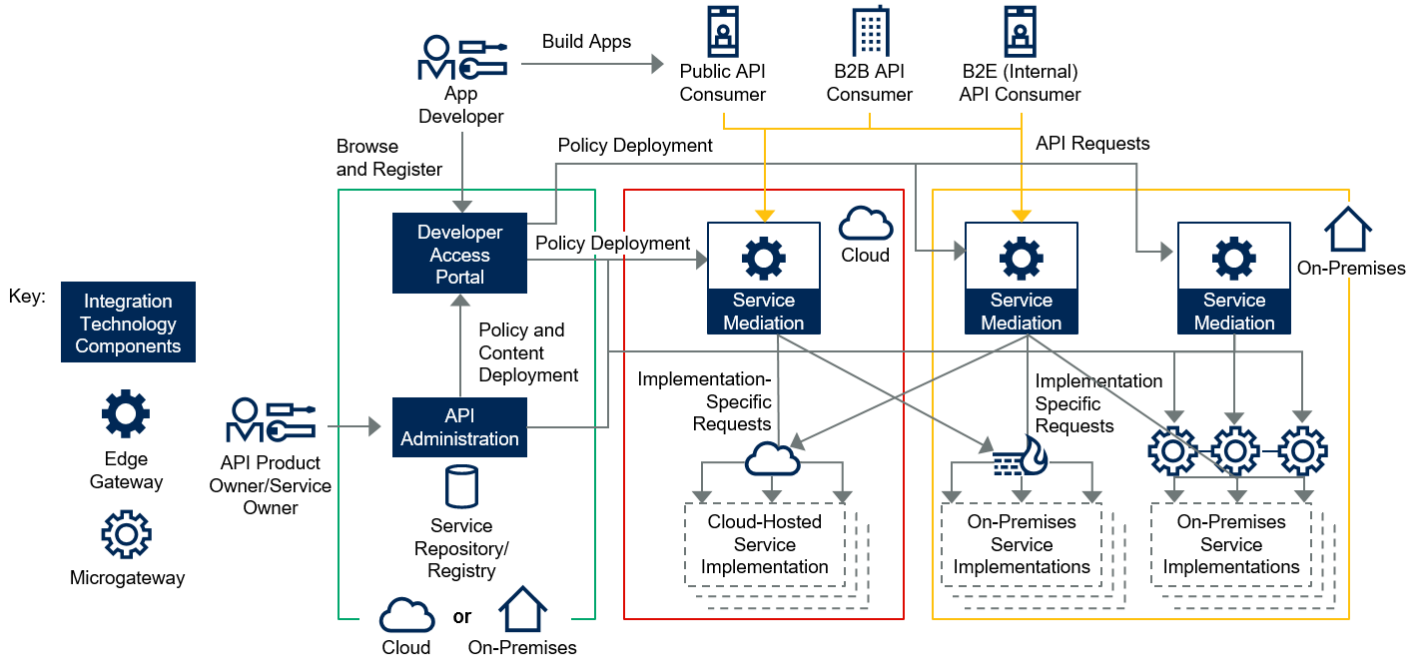
The other two core components are an API administration capability and an API developer portal. API providers use the administration capability to define, manage and monitor the publication of API endpoints and the access policies associated with them. The API developer portal is used to provide self-service to API consumers, allowing them to discover APIs and documentation, access support and usage metrics, and request and manage API keys for their applications.

> **API gateways include mediation features, but these should be used only to map from one interface specification to another.**

The logical deployment model for an API management solution is shown in Figure 11.

For further information on API management technologies and their alignment with both modern application architecture and application integration, see "A Guidance Framework for Evaluating API Management Solutions," "Choosing an Architecture for Managing APIs and Services" and "Decision Point for API and Service Implementation Architecture."

Figure 11. Logical Deployment Architecture for an API Management Solution



For more detailed analysis of the capabilities of API management technology, see "A Guidance Framework for Evaluating API Management Solutions" and "How to Successfully Implement API Management."

API management solutions are most relevant when your integration goals include the publishing of APIs that expose data or application capabilities to groups of developers or integrators who operate independently from the team providing or implementing those services. These groups include disparate teams or lines of business within an organization and external organizations, including customers, partners or third-party developers. API management solutions also help manage and promote the appropriate consumption of external APIs, allowing for access to new services and data sources without losing control of service dependencies.

For additional information on the API marketplace, see "Magic Quadrant for Full Life Cycle API Management."

Examples of API management products include:

- Google's Apigee Edge

- Broadcom-CA API Management
- IBM API Connect
- Kong Microservice API Gateway
- MuleSoft Anypoint API Manager
- Red Hat 3scale
- Software AG API Gateway
- TIBCO Cloud Mashery
- Tyk API Management Platform

Strengths

Strengths of API management solutions include:

- **Manages the publication and consumption of services via APIs using web-based protocols:** The technology also supports and promotes the service utilization, with integrated support for developer discovery and self-service through API developer portals. Using an API management solution to publish service capabilities can also provide valuable traceability of service dependencies and consumption. Most solutions also provide developer and administration portals to encourage self-service and to support the tracking and analysis of API utilization.
- **Manages and enforces policies to standardize and control use of distinct service implementations:** Policies can include security, authentication, authorization, traffic management and monetization. API management is complementary to other application integration technologies and application service implementation architectures (e.g., microservices and coarse-grained services using traditional application servers and frameworks). This is because the gateway deployment model is agnostic to the service implementation architecture and platform.
- **Decouples API specification from service implementation:** This is done by supporting basic payload and protocol mediation, including payload transformation, mapping, and filtering (e.g., between SOAP and REST and between XML and JSON).
- **Supports flexible and distributed deployment model:** This model is supported by API management products, including Apigee, MuleSoft and Red Hat 3scale, and it allows hybrid cloud and on-premises deployment architectures. Some products support consumption-based pricing rather than licensing deployed software instances.

Weaknesses

Weaknesses of API management solutions include:

- **Cannot be used to move data between systems and should not be used to create composite services:** Most API management products lack support for complex orchestration, sophisticated data transformation and application adapters. The features are required to aggregate application services and data into coarse-grained composite APIs. Most products have no application or SaaS-specific connector support available and rely on protocol-level connectors.
  - Using integration capabilities in the API management platform dilutes the decoupling and separation of concerns between policy enforcement and service logic. When orchestration capability is available in the API gateway component, it should be used with caution.
- **Predominantly focused on HTTP-based web services (REST, SOAP and XML-RPC):** Current products provide limited support for message-oriented or event-based protocols and patterns. Some products support WebSockets and/or integration with JMS providers.
- **Most are not suitable for integration scenarios that require polling or monitoring for changes:** These scenarios are best addressed by other integration technologies. (Note that some API gateways support file-based interaction — for example, Broadcom-CA Technologies, Axway and IBM.)

Guidance

Choosing application integration technology is tough. Not only are the requirements diverse in scope, scale and urgency, but they are also multiplying. Add to that the variety of technology solutions available and their overlapping characteristics — along with “we can do it all” marketing — and it is easy to see why there is often confusion or a lack of confidence in the decision.

This is compounded by the scope and complexity of some integration projects. Most enterprises will need multiple integration technologies that they manage as a hybrid integration platform. For example, combining API management and an iPaaS to govern the publication of composite services, or combining iPaaS and DIP to fully meet complex cloud-to-ground integration requirements.

Additionally, some vendors offer multiple technologies covering API management, ESB and iPaaS (for example, IBM, MuleSoft, Software AG and TIBCO Software). This is sometimes based on the same underlying platform and can offer advantages through better integration and more transferrable skills between the products. However, the maturity of a vendor’s integration products can be uneven across the portfolio. You should select the right technology for your particular requirement, rather than assume that buying everything from one vendor is a better option than combining offerings from multiple vendors.

Traditional Platform ESB Scenarios

Use a platform ESB technology deployment when:

- You have existing platform ESB software
- You have extensive orchestration, connectivity and operational requirements
- Application-to-application integration is still your primary integration model
- When you have expert integrators who will be using the ESB as their primary integration tool
- You are not focused on new Mode 2 integration requirements
- Your integration workloads need to be built, deployed, run and managed on-premises

Use your existing platform ESB software when you have extensive orchestration, connectivity, and operational requirements. When your integration requirements are prolific and complex, and when you can predict demand, use a platform ESB with supported application and legacy technology adaptors and built-in management, monitoring and high-availability capabilities. This often aligns to Mode 1 (systematic) integration delivery in a bimodal integration strategy.

You should also use platform ESB when application-to-application integration is still your primary integration model (e.g., as part of integration-centric service-oriented architecture), rather than service enablement (application-capability-centric service-oriented architecture), and when a majority of those applications are on-premises and you need a high quality of service.

Do not invest in new platform ESB software unless it is the only option that gives you the connectors and business logic you require. The platform ESB requires that you deliver integration technology, development and operations as a business-critical function of IT. This requires a center of excellence (COE) or integration competency center (ICC) with the skills and capacity to create and manage a large portfolio of application integrations and composite services. This creates a bottleneck for integration delivery that is incompatible with modern agile IT and business processes.

iPaaS Scenarios

Use iPaaS technology when:

- You have mostly cloud-hosted applications to integrate
- Time to productivity is a critical success factor
- Your capacity requirements (and ROI) are unknown from the beginning
- You have confirmed there are no compliance issues with the iPaaS provider that is hosting integration metadata in the cloud

Use iPaaS when you have cloud-hosted applications to integrate (SaaS, applications deployed to aPaaS or self-managed IaaS) with either on-premises or other cloud-hosted applications and when there is heavy data or application gravity toward the cloud.

When time to productivity is a critical success factor, the ease of deployment (near-instant access) and ease of use often found in iPaaS products can accelerate project timelines. The simpler user experience allows team members without deep integration experience to learn and use the tool quickly. This also expands the pool of integration resources available for future requirements.

If your capacity requirements (and ROI) are unknown from the beginning, the iPaaS operational expenditure (opex) cost model will allow you to explore the benefits of your service integration and grow cost with capacity and usage.

When you have extreme sensitivity about integration metadata, be aware that iPaaS solutions use cloud services to configure integrations and manage metadata related to their execution of those integrations. This information can include application connection details, security credentials, transaction logs and exception reports that may be an issue for organizations with strict security, privacy and compliance policies, even if managed securely by the iPaaS service provider.

Distributed Integration Platform Scenarios

Use DIP deployment when:

- Time to productivity is a critical success factor
- Your integration with existing assets will primarily use standard, protocol-based connectors
- Your integration workloads need to be built, deployed, run and managed on-premises

DIP offers an integration experience designed for ad hoc integrators connecting applications using modern protocols, but deployed and managed by you in your own data center. This can align to either Mode 1 (systematic) or Mode 2 (adaptive) in a bimodal integration strategy. A modern software platform will also support complex orchestration and transformation requirements that primarily connect to existing assets/services using standard protocol-based connectors (e.g., HTTP, SOAP, AMQP, SMTP, RSS, Lightweight Directory Access Protocol [LDAP] and file systems).

The DIP style is suitable when your integration workloads need to be built, deployed, run and managed as independent services. Building and running integrations as independent workloads (even miniservices or microservices) improve agility when you have application platform infrastructure and operating processes that can support your deployment, high availability and monitoring requirements. However, you must have mastered deployment and operation of distributed workloads. Whether using OS containers or more traditional methods (like virtual machines), the

benefits of modern integration software can only be realized if you have robust platforms, tools, processes and expertise for efficiently deploying, monitoring and managing disparate workloads.

Integration Framework Scenarios

Use integration framework technology when:

- Your integrators are professional developers
- You are creating integration flows that involve custom applications developed in-house
- Your capacity requirements do not justify a traditional ESB or modern integration software deployment
- You have mature application and integration governance processes and culture that will prevent misuse and uncontrolled proliferation

Use integration frameworks when your integrators are professional developers first and foremost and when your integration requirements will primarily connect to existing assets/services using standard protocol-based connectors (e.g., HTTP, SOAP, JDBC, JMS, AMQP, FTP, SMTP, RSS, LDAP and file systems).

Integration frameworks are also suitable when your integration requirements align to custom application deliverables (i.e., you have embedded your integration logic in other application platform operations) and when improving the productivity and quality of developer-created integration code by using proven application integration patterns is a priority.

Do not use integration frameworks extensively as an inexpensive alternative to a platform or modern integration software. Integration frameworks lack the governance, management and operational tools that are needed for effective delivery of integration workloads at scale. At small scale, when integration is “managed as code” through application delivery processes, frameworks are a useful tool. As your use scales up, do not fall into the trap of building your own integration management tools. Instead, plan to migrate to an appropriate modern integration software solution.

API Management Scenarios

Use API management technology when:

- You need to manage and secure your APIs
- Your mediation needs are limited to transforming between protocols and data formats
- You are not creating orchestrations that implement business logic

Use API management every time you share/publish the services you create across delivery boundaries, even if you need one of the other technologies to create the services effectively. It provides a common control plane for governing, managing and securing the delivery of any (web) service via published API. You can also use API management when you have existing services that need minimal/basic protocol mediation and payload transformation using web services protocols and formats (SOAP, REST, XML, JSON).

Remember, API management cannot be used for complex orchestration of services and should not be used to create new composite services. If you require distributed transaction coordination or complex orchestration of existing services exposed with a managed API, then you should use both API management and one of the alternative technologies. If your published API only needs to connect to one inner service API, then API management is a good fit. If you need to connect to more than one, you might be using the wrong tool.

Evidence

The Gartner Application Integration Pulse 2018 Survey was conducted to test understanding and adoption of the different approaches to application integration. The primary research was conducted online from 29 October through 3 December 2018 among 301 respondents in the United States, United Kingdom, Canada, Australia and Singapore.

Qualified respondents are IT end users who have at least 50 employees and at least $50 million in annual revenue. Respondents were required to be personally involved in delivering application integration. Quotas were set to ensure an even spread of respondents across geographic regions.

The study was developed collaboratively by Gartner analysts and the Primary Research Team.

Disclaimer: Results do not represent "global" findings or the market as a whole, but reflect sentiment of the respondents and companies surveyed.

Note 1
Technologies Not Assessed

The scope of a hybrid integration platform is broad — any technology that acts as integration middleware, connecting two or more systems, could be considered part of your HIP. To manage the scope of this research, we have focused on the main technology platforms used by Gartner clients for application integration. We did not assess the following technologies as part of this research, and we have outlined our reasons below.

- **Message and Event Broker Middleware** is used to connect event-driven and message-based systems to each other. The broker is sometimes responsible for making routing decisions and may include the capability to filter messages or events. While these technologies are important to application integration, their capabilities do not include sufficient mediation capabilities to make them the kind of general integration middleware that forms the core of a HIP. Their limited

mediation and specific use case make any assessment of them against the criteria used in this research meaningless. For more details on these platforms. see "Assessing Event-Driven Middleware Technology for Modern Application Architecture."

- **Master Data Management, Data Pipelines and Data Integration Technologies** form an important part of your HIP, but you use them when you are taking a data-centric (rather than application-centric) approach to integration. The focus of this research is application integration; therefore, these technologies were not assessed. For more details on data-centric integration approaches, see "Deploying Effective iPaaS Solutions for Data Integration."

- **The Integration Capabilities of SaaS Applications** offer a quick way to connect SaaS and on-premises systems. However, these features are not general-purpose integration middleware as you can use them only when one of the things you are integrating is the SaaS application. We have not assessed these capabilities here, but we compared them against iPaaS and integration platform software in "Comparing Integration Options for Cloud-Hosted, SaaS and On-Premises Applications."

- **Managed File Transfer** is not a common choice for creating new integration flows. While many organizations have existing managed file transfer capability that makes up part of their hybrid integration platform, they do not see it as a strategic integration technology.

- **Service Meshes** have emerged as the technology of choice for connecting services together in a microservices architecture. However, a service mesh provides routing and security features, rather than mediation and transformation. A service mesh has some overlap with an API gateway, but very little with an ESB, DIP or iPaaS. For more details on service meshes, see "Assessing Service Mesh for Use in Microservices Architectures."

- **Business Process Management (BPM) and Robotic Process Automation (RPA)** technologies also connect systems together. One of the key use cases of integration technology is connecting systems together to automate business processes. However, the focus of BPM and RPA is on processes, rather than requests or events, and it specializes on this one integration use case. For more information on how RPA fits with integration technologies, see "Comparing Digital Process Automation Technologies Including RPA, BPM and Low-Code."

Document Revision History

Choosing Application Integration Technology - 28 October 2016

Recommended by the Authors

A Guidance Framework for Evaluating Application Integration Platforms

A Guidance Framework for Evaluating API Management Solutions

How to Successfully Implement API Management

Comparing Integration Options for Cloud-Hosted, SaaS and On-Premises Applications

Essential Skills for Modern Integration Architecture

Recommended For You

Choosing Between Data-, Application- and Event-Centric Integration Styles

A Guidance Framework for Evaluating Application Integration Platforms

Gartner Peer Insights 'Voice of the Customer': Enterprise Integration Platform as a Service

Choosing the Best Approach for Salesforce Integration

Gartner Peer Insights 'Voice of the Customer': Data Integration Tools

© 2020 Gartner, Inc. and/or its affiliates. All rights reserved. Gartner is a registered trademark of Gartner, Inc. and its affiliates. This publication may not be reproduced or distributed in any form without Gartner's prior written permission. It consists of the opinions of Gartner's research organization, which should not be construed as statements of fact. While the information contained in this publication has been obtained from sources believed to be reliable, Gartner disclaims all warranties as to the accuracy, completeness or adequacy of such information. Although Gartner research may address legal and financial issues, Gartner does not provide legal or investment advice and its research should not be construed or used as such. Your access and use of this publication are governed by Gartner's Usage Policy. Gartner prides itself on its reputation for independence and objectivity. Its research is produced independently by its research organization without input or influence from any third party. For further information, see "Guiding Principles on Independence and Objectivity."

© 2020 Gartner, Inc. and/or its Affiliates. All rights reserved.