Gartner

Innovation Insight: The Digital Integration Hub Turbocharges Your API Strategy

Refreshed 1 November 2019, Published 26 June 2018 - ID G00360082 - 21 min read

ARCHIVED This research is provided for historical perspective; portions may not reflect current conditions.

By Analyst(s): Massimo Pezzini, Eric Thoo

Initiatives: Software Engineering Technologies

API-based fast access to data dispersed across multiple sources is costly and needs notable integration work. Application leaders should implement a digital integration hub to enable high-scale access, minimize workload on systems of record and deliver additional value via use cases like analytics.

Newer version of this document

10 July 2020 Innovation Insight: Turbocharge Your API Platform With a Digital Integration Hub

Overview

Key Findings

- Implementing APIs that provide fast and high-scale access to back-end system of record applications and data sources may imply:
  - Exposing these systems to a potentially massive, often low-value, workloads.
  - Implementing complex integration between the API services layer and the systems of record.
- Supporting these scenarios may be quite costly (for example, in terms of hardware upgrades of the back-end systems) and difficult to implement (for example, if the data needed by the front-end API services are dispersed across a large number of systems).
- To address these challenges, application leaders have at times implemented various approaches that replicate back-end data into caching layers, but they have done this in a tactical fashion that impedes a broader utilization of this valuable business data.

Recommendations

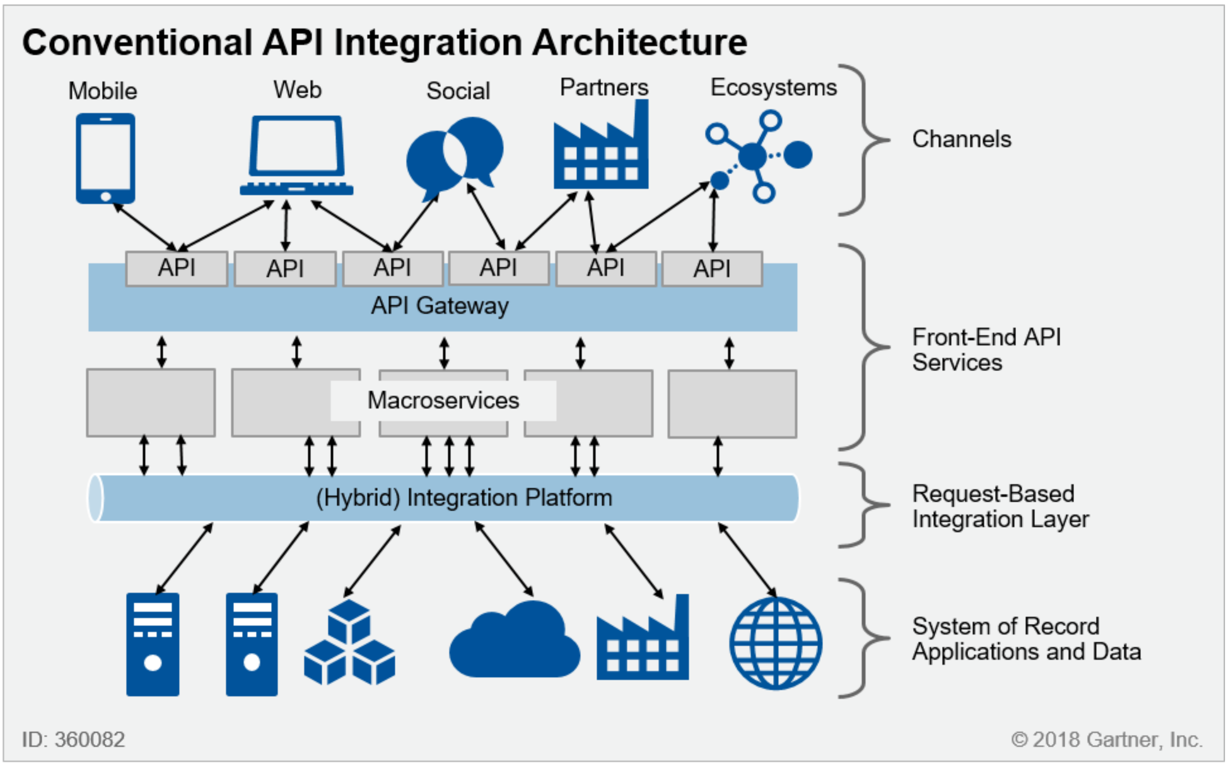
Application leaders responsible for integration strategies and infrastructure, who are planning the use of APIs to support large-scale, multichannel digital scenarios should:

- Reduce the cost and complexity of the API service layers, decouple them from the system of record, and enable 24/7 operations by consolidating back-end data sources into a scale-out, high-performance digital integration hub (DIH).
- Enable synchronization between the DIH data store and the back ends by fully supporting event-driven, request-driven and batch interactions in their hybrid integration platform.
- Empower repurposing of DIH data for other usages by tracking and sharing metadata and lineage of all data flow supporting the DIH implementation.

Analysis

Application programming interfaces (APIs) are the ubiquitous and mainstream enablers for digital modernization. They empower omnichannel customer experience platforms, mobile apps, social networks, chatbots, ecosystem partners and other systems that are used to access enterprise data. Most of that data, however, is typically hosted in "system of record applications," such as ERP, CRM, SCM, core banking, core insurance, travel reservation and other business-critical, "back end" systems (see Figure 1).

Figure 1. Conventional API Integration Architecture



API = application programming interface

Source: Gartner (June 2018)

Implementing and managing these APIs (including the associated support service as well as data and application integration flows) might pose challenging issues in terms of extensive impact on the system of record layer due to:

- The stress on back-end systems to support, often significantly high, additional workload generated by the applications consuming the front-end API services.
- The integration work needed to implement the API services — which might prove substantially, or even intractably, complex — should the API services themselves require access to data scattered across multiple, dispersed back ends.

Increasingly application leaders responsible for integration strategies and infrastructure are forced to tackle these issues when planning, designing and implementing APIs to support large scale, multichannel digital scenarios.

The emerging DIH architecture can help application leaders reduce the cost and complexity of enabling API access to data held in system of record applications for large-scale/high-performance scenarios.

Although this is the primary observed use case, in some instances a DIH architecture has also been used to create a data store for analytical purposes. Scenarios in which data needs to be continuously refreshed in real time from "events" coming from multiple sources are currently proliferating. For these scenarios, which typically maintain a 360-degree, real-time view of critical business entities (for example, customers, products, suppliers, patients or students), the DIH may prove a good architectural fit.

The DIH can also be used to empower mixed scenarios where the support for a "transactional" API layer is combined with some form of analytical application.

Definition

A digital integration hub is an advanced application architecture that aggregates multiple back-end system of record data sources into a low-latency and scale-out, high-performance data store. A DIH typically supports access to data via an API services layer. The high-performance data store is synchronized with the back-end sources via some combination of event-based, request-based, and batch integration patterns.

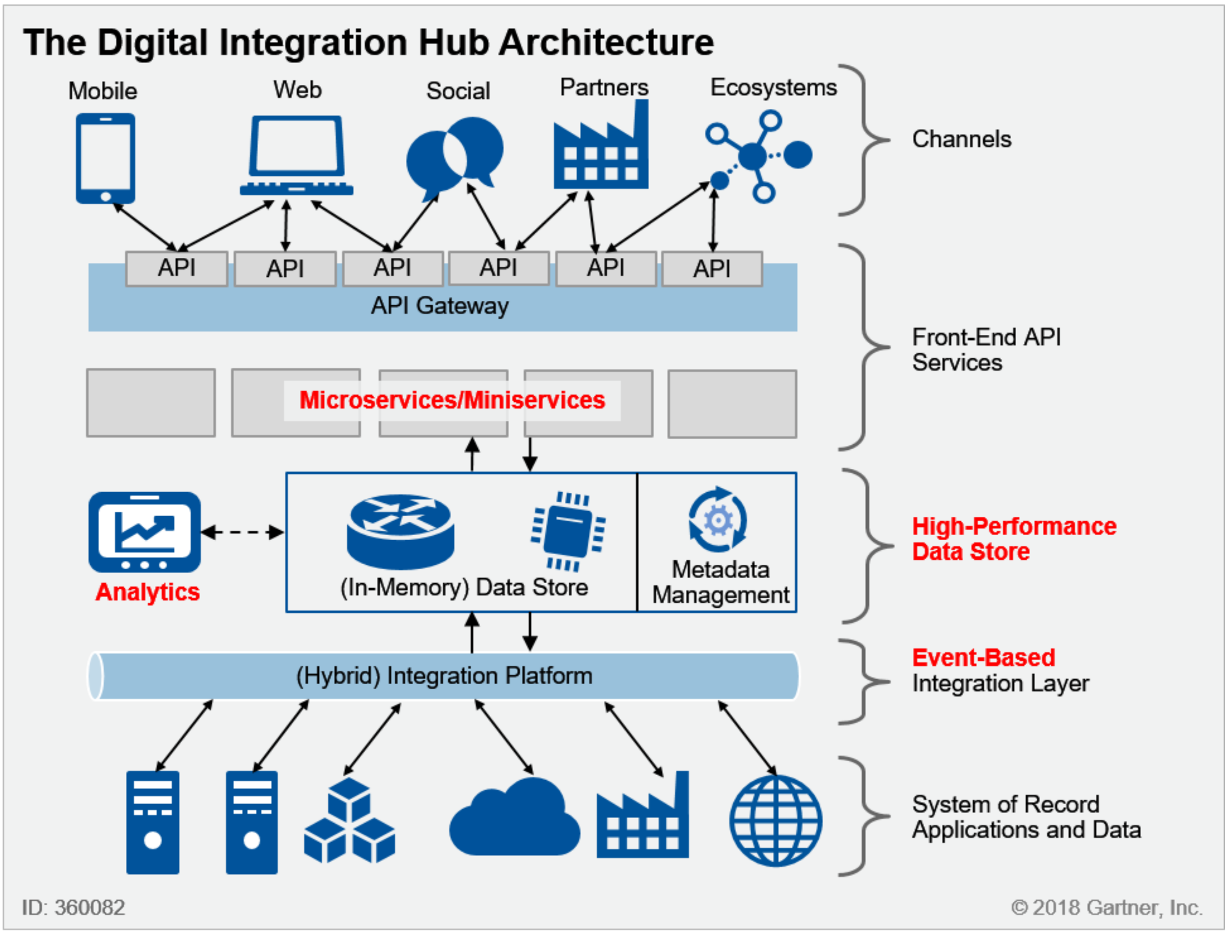
The goal of a DIH is to enable the implementation of large-scale, high-throughput and low-latency front-end API services by:

- Preventing the back-end systems from being overwhelmed with excessive API services-generated workloads.
- Avoiding intractably complex integrations to support the API services implementation.
- Decoupling the back-end data sources from the front-end API services.

The DIH can be seen as an advanced form of push-caching between the front-end API services layer and the system of record layer. In contrast to classic caching, however, the DIH stores "all" of the relevant data (for example, information about customers, products and employees) and not just "the most frequently accessed." Moreover, additional business value can be obtained by extending the architecture with capabilities that support additional use cases, such as analytics.

Figure 2 shows the logical components (in **Red**) that differentiate the DIH from the conventional architecture seen in Figure 1.

Figure 2. The Digital Integration Hub Architecture



Source: Gartner (June 2018)

Description

Access to dispersed system of record data via an API service layer is conventionally implemented by developing integration "macroservices" that aggregate back-end data via orchestration or data virtualization techniques. This approach has been well-known since the late 1990s — the early days of service-oriented architecture — and is still routinely implemented by mainstream organizations via a typically request-based integration layer used to develop the macroservices. This integration layer is implemented either via custom code or integration platforms (enterprise service buses, iPaaS or, increasingly, a hybrid integration platform [HIP] combining these technologies) that connect the API layer with the back-end endpoints (see Figure 1).

However, in a demanding, high-performance and high-scale digital business environment, where APIs must serve millions or even tens or hundreds of millions of calls per day, this approach may prove inadequate for the following reasons:

- **High cost:** The huge number of API calls potentially imply a massive, often low-value workload (primarily inquiries that do not directly generate revenue) which ultimately hits the system of record applications and the integration layer. This may entail significant costs to upgrade your infrastructure to support the additional workload. It may even be impossible to support this kind of workload altogether without radically redesigning the system or record applications, often a simply unacceptable proposition (see Note 1)

- **Complexity:** The integration macroservices could be extremely complex to develop, deploy, run and manage, especially if the back-end data is highly fragmented across multiple applications and data sources.

  For example, one U.S. consumer packaged goods company uses a DIH architecture to enable mobile apps for salespeople to have real-time access to aggregated customer and product data that is scattered across several dozen ERP systems.

- **Availability:** If some back-end systems are unavailable — for example, during a maintenance window — the integration layer cannot serve the front-end API services, and can significantly impede business operations. This is increasingly unacceptable in a 24/7, always-on, global environment.

- **Tight coupling:** A change in the back-end systems (for example, a modification of a field definition in a database table) implies some re-engineering of the integration macroservices or their complete redesign, for example, if a back-end system is replaced with a different solution.

The DIH is an emerging architecture that addresses these challenges by implementing a high-performance data store layer between the API service layer and the system of record.

By storing an aggregated replica of the system of record data needed by the channel applications, the DIH protects the latter from excessive workloads while optimizing the data access latency and responsiveness for the former.

As such, the DIH can be considered as a use case that your HIP strategy must support (for example, by providing event propagation/brokering capabilities), but it is not necessarily part of the HIP implementation itself (see "How to Implement a Truly Hybrid Integration Platform"). Nonetheless, some user organizations and vendors refer to the DIH as a way to "put data in the integration layer."

The key architectural building blocks of the DIH (as seen in Figure 2) are:

- **The high-performance data store:** This layer includes two components — the data store itself and a metadata management feature:
  - **The data store:** This holds a copy of an appropriate subset of the back-end data. This subset provides a single consolidated "view" of entities such as customers, products, prices, itineraries, students and patients, the data for which is originally stored in one or multiple system of record applications and data sources.
  - The high-performance data store can be developed in many ways. For example, via a regular RDBMS. In the most-demanding scenarios, however, it is often implemented by using in-memory computing technologies (typically in-memory data grids) and/or NoSQL DBMSs. This is because of the scale-out characteristics of these technologies (see "Market Guide for In-Memory Computing Technologies" and "Magic Quadrant for Operational Database Management Systems").
  - **Metadata management:** Critical to the ability of supporting diverse types of data flow, the core of a DIH architecture must emphasize reusable capabilities rather than narrow, one-off support of individual applications or data structures. As usage scenarios evolve — for example, by repurposing data and process rules to other use cases — the DIH must enable flexible changes in back-end systems and API service data consumers. It must also support the evolution of the integration processes and events supporting the implementation. This requires the DIH to be equipped with metadata management capabilities that allow it to discover, capture and synchronize metadata models with back-end applications. It must also be able to derive physical/logical models, discern and reconcile relationship, track lineage and impact, share and synchronize metadata with other tools, and enable metadata-driven development and introspection (for example, to support domain-driven design approaches).

- **Front-end API services:** These implement the API services' functionality. Those (mini or micro) services do not access the system of record endpoints directly, as the data they need is available in the DIH high-performance data store. For practical reasons, however, the front-end services can, at times, directly access certain back-end data or federated views of data made available via data virtualization. For example, in basic implementations the API services access the high-performance data store only for queries, whereas updates are performed directly on the back-end systems, according to a Command Query Responsibility Segregation (CQRS) pattern. In more advanced scenarios, however, the API services can update the high-performance data store, too.

- **The integration layer:** This is in charge of keeping the system of record sources and the high-performance data store in sync. In those implementations where this data store is "read only" (that is, from the front-end API services standpoint) the synchronization is unidirectional (changes in the back-end systems are propagated to the high-performance data store). In scenarios where the high-performance data store can also be updated by the API services, the synchronization must be bidirectional (that is, changes performed in the intermediate data store are propagated back to the system of record applications).

  Note that in the conventional architecture (Figure 1) the integration layer's main goal is to enable the front-end services to synchronously "call" the back-end system to get data. In DIH architecture, the integration layer is primarily in charge of sending asynchronous event notifications ("customer XYZ's address has changed") from the systems of record to the high-performance data store layer, and vice versa.

  This layer is typically implemented using a variety of techniques, such as:

  - ETL (usually for the initial high-performance data store loading, but also for periodic refreshes).
  - Data virtualization.
  - Event brokering/message oriented middleware (MOM).
  - Change data capture (CDC).
  - Integration platforms (ESBs or iPaaS, for example).
  - Stream processing platforms (such as Apache Spark and Flink).

In forward-looking implementations, these technologies are increasingly combined into a cohesive HIP that is designed to support a variety of other use cases.

- **Analytics:** Although not a defining characteristic of DIH architecture, in some situations the data in the consolidated data store is also utilized for analytics, via a variety of tools (for example, classic BI tools, data visualization tools and predictive analytics). At times, however, support for analytics is the primary driver of DIH adoption. Typically, this is in situations where the analysis must be performed on data that is constantly updated from real-time systems (for example, in IoT, e-commerce, travel reservation, banking or insurance applications). The real-time flow of API calls and events crossing a DIH infrastructure also provides an opportunity to further extend the architecture with stream analytics capabilities (for example, to deliver value-added services to clients).

Benefits and Uses

The prevailing use case for DIH architectures seen so far by Gartner are related to the cost, complexity, availability, and coupling benefits of this approach over conventional API architectures.

Use cases for a DIH architecture include, but are not limited to enabling:

- **Responsive user experience:** This is about providing users (for example, customers, students or salespeople) with a responsive user experience while enabling them access to a consolidated, yet real-time, view of data scattered across multiple system of record back-ends.
- **"Defending" systems of record:** Providing a defense from the potentially excessive, often low-value, workloads generated by the channels.
- **24/7 support:** Always-on access to the front-end API services, even in situations where the back-end systems must be put offline for maintenance, upgrade or other reasons. In these cases, the DIH can continue to provide the API services with the data they need, although certain functions (for example, those requesting updates to data in the back-end systems) may be temporarily inhibited or limited in functionality.

- **Decoupling the front-end layer from the system or record applications:** This is a desirable DIH outcome for two primary reasons:
  - **Supporting legacy system of record applications modernization:** By making it possible to replace some system of record applications in the future, while minimizing the impact on the API service layers and, consequently, the channel applications.
  - **Normalizing the APIs for a certain application domain:** So that with a single set of APIs the channel applications can access data held in multiple systems of record (for example, ERP packaged applications), possibly from different vendors.
- **Providing real-time business insight:** Gathering data on user behavior or offering them additional services (including search, predictive analytics or custom analytics on historical data — see Macquarie Bank case study in Note 2).

  In particular, DIH architectures are at times used to maintain an up-to-date picture of customer behaviors, competitor prices, shipments or other fast-changing data that can be analyzed in real time to detect "business moments." These are situations that require actions in business real-time (that is, fast enough to maximize an opportunity or minimize the impact of a threat). For example, a customer steadily purchasing higher and higher value products, thus indicating a change from "gold" to "platinum" status. However, also in these cases, API-based access to the data in the consolidated data store is considered a valuable functionality of the DIH implementation.

The benefit of DIH broadens when applied as part of a wider scale of data hub strategy (see "Use a Data Hub Strategy to Meet Your Data and Analytics Governance and Sharing Requirements"). This represents the aspirations of forward-looking enterprises that seek to be comprehensively equipped with common capabilities of practices and tooling using coherent, hub-based approaches that span integration, data and analytics, alongside governance. Deploying a DIH can be seen as a way to add a set of complementary or embedded capabilities to existing data hub and HIP strategies. It sets out to modernize both application and data management infrastructures in a shared, consistent approach toward data and analytics governance, data sharing and integration.

Adoption Rate

The implementation of DIH projects is still primarily limited to leading-edge organizations with large-scale requirements. They have the skills, and technical and financial resources needed to implement such an architecture.

Industries where Gartner has observed deployments of various forms of DIH architecture are typically those serving large populations of users, where providing low-latency, real-time, consolidated and accurate access to data is critical for business success. These industries include retail, travel and transportation, banking, insurance, telecom, consumer packaged goods manufacturing, gaming and higher education.

The sophistication of these implementations varies greatly, from relatively simple advanced caching layers, all the way to full implementations of the DIH architecture, including bidirectional synchronization as well as real-time and stream analytics functionality.

> **In the majority of the examples analyzed by Gartner, the ultimate driver toward DIH adoption is some form of digital transformation initiative.**

Factors that will facilitate and drive future adoption, also by mainstream organizations, include:

- Vendors coming to market with applications or cloud services based on DIH architectures, typically to support real-time, 360-degree customer views (for example, customer engagement hub solutions) or similar use cases.
- Vendors proposing various out-of-the-box combinations of integration platforms, data management, event brokering and stream analytics, either in the form of software products or cloud services (such as the data hub form of iPaaS).
- The urgency of rethinking organizations' "customer experience" along the lines of multichannel and real-time paradigms.
- Businesses desire to have deeper and more real-time insights about their business by leveraging event-based approaches and analytics.

A growing number of organizations will invest in DIH architectures to support their digital business initiatives, often by adopting these architectures as "embedded" in packaged and/or SaaS applications and other software packages and/or cloud services.

Risks

Implementing a DIH architecture is still challenging as it requires application leaders to address complex and often unfamiliar issues, such as:

- Deploying "scale out" and low-latency data management technology (such as NoSQL DBMSs and/or in-memory data grids) to implement the high-performance data store.
- Supporting bidirectional, event-driven synchronization between the high-performance data store and system or record applications at the back end.
- Developing API services that can tolerate the event-based "eventual consistency" between the data in the DIH and the back-end systems.
- Designing a comprehensive, future-proof and shared-information format for the business entities in the DIH (for example, a customer data format that must support a variety of channel applications and analytics use cases).
- Providing monitoring, management, administration and security across a complex, highly distributed architecture made of multiple, loosely coupled components and moving parts.

For many application leaders, tackling each of these issues might prove challenging in itself. The compounding of these challenges exposes them to even higher risk of failures or suboptimal outcomes if addressed without proper architectural and technical analysis and skills building. Nor should application leaders expect much support from service providers, given the industry's still limited experience of DIH architecture.

The use of potentially different, complementary application infrastructure, integration, and data management technologies to implement their DIH also exposes application leaders to the challenge of sharing and reconciling metadata across these different tools to facilitate a common operating environment.

Failure to consider this need can expose a DIH initiative to multiple risks, including:

- Lack of openness in the supporting tools that inhibits business analysts' ability to discover data across multiple integration platforms, or event streams.
- Need to understand the structure of (or to interact directly with) the underlying applications and data environments due to the limited metadata-driven discovery and introspection of abstracted interfaces to data sources and application flows. This risk can be compounded if APIs are not sufficiently abstracted from the underlying applications and data environments, so that complex underlying data models are surfaced in API definitions. To avoid this problem — often referred to as "data model dumping" — an "API first" approach can be used to design APIs based on understanding of consumer requirements.
- Inability to monitor, track, analyze and share metadata, predicated on having an inventory of data assets and catalogs.

To avoid these risks, DIH approaches must employ shared metadata through which cross-projects can obtain insights. Ensuring semantic consistency of diverse data and application flows prepares the DIH for repurposing of data for future usage, such as real-time analytics, and to enable data preparation.

> **Application leaders should set expectations that implementing DIH-architectures requires the assembly of multiple technologies, possibly from different providers, to implement the core functionality and support data governance, monitoring, management, administration and security across a complex set of systems.**

Recommendations

Adopt DIH architectures when tackling projects with some combination of the following requirements:

- Providing a responsive, low-latency user experience, especially in multichannel scenarios targeting large constituencies (hundreds of thousands or greater), where the channel applications require real-time access to system of record applications.
- Protecting system of record applications and data from potentially excessive, low-value workloads generated by the front-end API services.
- Enabling front-end API services to access data scattered across multiple back-end systems, which would be hard to implement using conventional integration/orchestration techniques.
- Drastically decoupling front-end API services from the systems of record to favor a dual-speed, bimodal evolution of the two layers.
- Maintain an up-to-date consolidated picture of fast changing data that can be:
  - Used to provide additional services to users.
  - Analyzed in real time to detect "business moments."

Implement a DIH architecture considering that:

- Out-of-the-box DIH-based products are very few and limited to specific use cases.
- You will have to assemble multiple technologies and/or cloud services, possibly from multiple providers, to build up a DIH that meets your requirements.
- You must provide integrated monitoring, management, administration and security across a complex puzzle of systems.
- You should factor in metadata and lineage of integration constructs as evaluating criteria for your DIH component parts, in which open metadata accessibility is supported to enable reuse of designs, data, and aligned integration flows.
- You cannot count on widespread availability of readily available and low cost skills as industry experience is still limited and concentrated in few, pioneering vertical sectors.

Frame DIH as complementary to HIP and data hub strategies that can be deployed together to enable data and analytics governance, data sharing and integration.

Representative Providers

The following providers do not offer a "DIH in a box" solution, but have products or cloud services that, in combination with other technologies, can support the implementation of a DIH architecture. One of these providers makes available cloud services that are implemented on the basis of a DIH concept.

Providers Combining In-Memory Data Grids, Event Brokering and Stream Analytics:

- GigaSpaces (InsightEdge)
- GridGain (GridGain In-Memory Computing Platform)
- Hazelcast (Hazelcast + Jet)
- Pivotal Software (GemFire + Greenplum Connector)
- Striim (Striim Platform)
- Fincons (Fincons Fast Data Lake)
- Software AG (Terracotta DB + Apama + Universal Messaging)

Providers Combining Integration and Data Management Capabilities:

- Bouvet (Sesam)
- Dell Boomi (Boomi AtomSphere, Boomi MDM and Boomi Flow)
- Informatica (Informatica Data Integration Hub)
- Liaison (Alloy)
- Maestrano (Open Cloud Integration Platform),

Providers With In-Memory-Enabled Data Virtualization Technologies

- Denodo (Denodo Platform)
- TIBCO (TIBCO Data Virtualization)
- Stone Bond Technologies

Providers With DIH-Based Software or Cloud Services

- Segment (Segment)

Note 1
Examples of DIH Implementations to Support Large Channel-Generated Workloads

- A consumer electronic company in Europe implemented a proto-DIH architecture to offload its SAP ERP from the massive inquiry workload generated by its B2B e-commerce portal. At some point, this was pushing more workload into the firm's SAP ERP than the rest of the enterprise.
- A railway company in Asia implemented a DIH architecture because its conventional architecture could not support the dramatic increase in workload in its reservation and ticketing application system coming from the digital channels. This resulted in system unavailability at critical peak times.
- A government portal in the U.S. had to be re-engineered according to a DIH-like architecture because the original design could not support the unexpectedly high workload coming from citizens. This was causing crashes and unavailability of the system in the first weeks of operation.

Note 2
Example of DIH Delivering Innovative Services to Clients

Macquarie's Banking and Financial Services Group — the retail banking and financial service business within Macquarie — identified the need to build a smarter digital banking platform and mobile app that could help it continually evolve and build a more personalized and intuitive experience for its Australian retail banking customers.

This was achieved by implementing a high-performance data store that aggregates customer-related data from a variety of internal systems of record. The data is then made available to mobile and web apps for clients as well as to white-label partners who want to incorporate Macquarie's services into their own offerings. The bank also supports an open banking API strategy, which is the first of its kind in Australia.

The digital platform, which is deployed in a public cloud environment, is based on APIs supported by an API gateway, and microservices-based architecture is deployed in containers. Orchestration is managed by Kubernetes. The platform uses a NoSQL DBMS (Apache Cassandra) to implement the high-performance data store. Through the use of an event stream platform, real-time analytics (Apache Spark) and a search capability (Apache Solr) against the high-performance data store the bank has been able to deliver innovative services to its clients. These include a natural-language search in the customer transaction history, customer self-service analytics as well as personalized recommendations and services.

Document Revision History

Innovation Insight: Turbocharge Your API Platform With a Digital Integration Hub - 10 July 2020

Recommended by the Authors

Some documents may not be available as part of your current Gartner subscription.

Innovation Insight for Hybrid Integration Platforms

Magic Quadrant for Operational Database Management Systems

Market Guide for Application Integration Platforms

Magic Quadrant for Data Integration Tools

Magic Quadrant for Metadata Management Solutions

Innovation Insight for Event Brokers

Adopt Stream Data Integration to Meet Your Real-Time Data Integration and Analytics Requirements

Data Hubs: Understanding the Types, Characteristics and Use Cases

Infuse Your Data Hub Strategy With Data and Application Integration