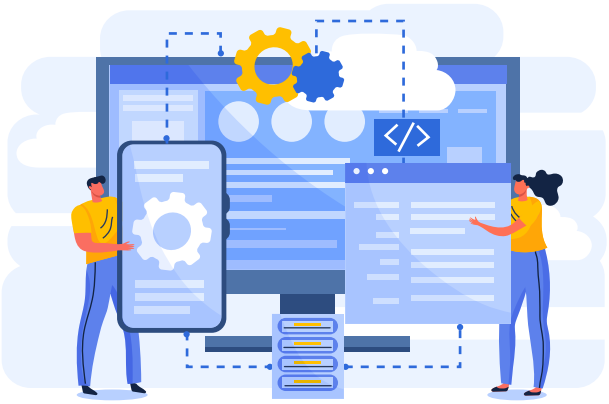
opshub

accelerating innovation

Integrating Applications: Orchestration or Federation?

Web - www.opshub.com | Email - sales@opshub.com | Phone - +1.650.701.1800

Software is eating the world. Everywhere we look, software applications are used to perform and automate tasks that were previously manual ones. Innovative products and business models are quickly displacing outdated ones, and innovation trumps incremental efficiency improvements every time. However, the innovation process is time sensitive: being early to market has substantial first-mover advantages. But as we all have a finite capacity for ideation, the more teams and people are involved in the process of innovation the greater the likelihood of success.



This proliferation of applications with the corresponding information fragmentation and the need for speed is fundamentally driving the need to create a unified application mesh. An application mesh in which individual teams are empowered to make data-based decisions. In unifying this application mesh, businesses are often challenged by the need to integrate the various tools being used in their business processes.

These challenges can broadly be categorized as either orchestration challenges or federation challenges.

Loan application use case: Understanding where orchestration is ideal

A simple orchestration problem is the need to read data from multiple systems to make an informed local decision or choice, optionally followed by a single action which could be writing an atomic piece of data into another system.

> **For example, a business process could need to check a customer's address (read in from a CRM system), then get their credit score and social security number from a cloud-based credit rating system, and finally their employment history from a third system of record, in order to make a decision on their eligibility for a loan.**

Then, once the decision has been made, we might need to create a new account for them in our loan application, which is a single atomic write. Recovery in case of failure in these types of systems is pretty simple. Since each system contains a distinct data set with no overlap and there is a single 'write' action, recovery is a simple rollback step. You check if the single write failed, and if it did, you redo it.

In the previous example, if recovery capability is needed, it could be programmed or scripted, as all is needed is to check if the account was successfully created, and if not, simply retry the action to create the account. This can be achieved with orchestration.

Orchestration is insufficient when multiple applications need to be updated

But if orchestration involves writing/updating multiple applications, or if the systems contain overlapping or interrelated data objects, then recovery can quickly become extremely complex. We need to check for concurrent updates to interrelated objects made by multiple systems, and ensure that any interdependencies are correctly handled, leaving the systems in a consistent and correct state. Rollback for recovery is typically not feasible as other people/processes may have read the initial update/write and acted on its basis.

In reality, a rollback is nearly impossible when you have multi-system writes across independent systems (multiple users, other people/processes reading/writing the same data);

It becomes even more complex when the information being updated is not simple records, but contains embedded references to other objects, links or references to other systems for traceability or non-textual information like images. In those cases, simple copying of the data will not work.

in such cases, you need “roll-forward”- bringing all systems to a consistent state without going back to the previous state. This is very hard to do with orchestration and needs a different approach- federated integration.

Federation use case

Federation integration based approach to integration is needed when there are multiple replicas of some logical object, each residing in a different system of record, which need to be kept consistent across all systems, even as they are being worked on and possibly updated in one or more systems. For example, customer data such as their address might be kept in both the CRM system and the marketing automation system and independently updated in either one, with the need to make a corresponding change in the other.

Similarly, a customer issue might be initially reported into and stored in a ticketing system of record like ServiceNow, and its status might be updated in a developer's issue management system, which then needs to be reflected in the ticketing system. Keeping those two instances which are the same logical object in sync is a federation management problem.

By relying on a **federation model instead of a centralized database**, manufacturers and their suppliers can securely connect complex lifecycle data across different tools. This approach ensures that shared information remains synchronized while allowing each group to retain full control over its own systems and intellectual property.

Information in modern systems is typically active, interconnected, and large – there are complex relationships, embedded objects, active references, large attachments etc. In the above example, the customer issue in ServiceNow might include a screenshot of the issue, the developer's issue management record might embed an automated test that reproduced the problem and a configuration file that fixes the issue. As you can see, -this requires complex handling to keep the two instances in sync. This handling achieves eventual consistency through 'roll-forward' – multiple bidirectional synchronization processes between systems that get to a state where all data is logically consistent in the new, updated state.

The key insight is that there are two distinct problems here. While both are the result of having multiple software systems that need to be integrated, they are fundamentally different, and as a result, addressing them requires different tools.

Orchestration solutions solve the problem of needing to get data from multiple sources and then acting on that data in a local transactional way. They provide a programmed or scripted workflow that automates repetitive, multi-step processes. They support many tools, including ALM and DevOps systems as well as CRM or SFA, out of the box. These orchestration solutions are often complex and requires extensive development efforts to deal with multi-step processes. But for all their complexity, these tools do not guarantee reliability or integrity of replicated data across multiple systems of record because that Is not the problem they were designed to solve.

If your goal is to ensure consistency of replicated data residing in different systems while empowering all team members to share and modify the same data while using the best tool for their specific job – you have a federation problem. On the other hand, if you want to ensure that relationships between data items are maintained, along with their history – you have a data federation problem. For that, you need a federation solution.

The promise of Eventual Consistency

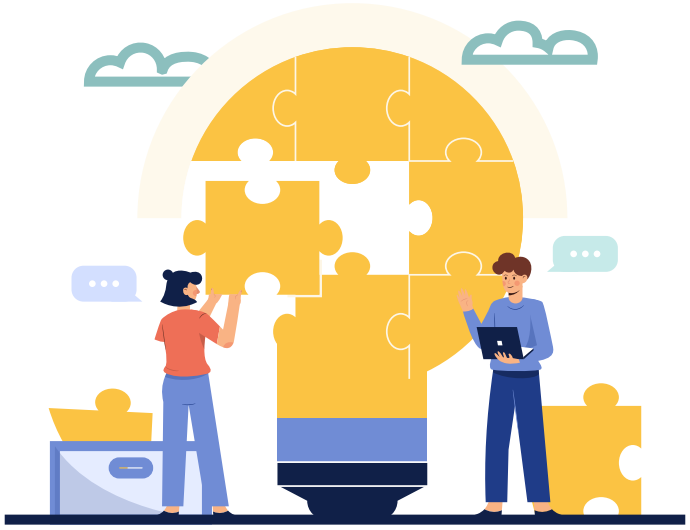
Some of the key attributes one should look for in federation solutions include proper handling of history, and synchronization of all data elements, including references, embedded images, and attachments. The large data sets found in most enterprises make scalability an issue, and addressing scalability requires efficient synchronization mechanisms. While at any given point in time, the two systems might not be 100% consistent (given the nature of independent systems working in parallel on the same data set), it is crucial that the underlying federation system is be able to guarantee ***eventual consistency*** – that after a period of time, all the data will be updated, in all systems, in a logically consistent manner.

In order to guarantee such eventual consistency, the federation system must support bidirectional synchronization and have a mechanism for monitoring the changes to the data set that doesn't require a full scan – so that only differences need to be updated. It should have a defined method to detect out-of-sync entities and consistent rules to be applied to such entities. The reconciliation rules should be robust so that there is no duplication or data loss, even when the data is being moved from a manual integration platform to an automatic integration platform.

When failures occur, the solution should allow for automatic retries of failures, and if those fail, enable an administrator to easily take corrective actions for all the entities in the failure queue. The administrator should easily group similar entities and take bulk actions to resolve them. It is only solutions that provide ALL these capabilities that can guarantee eventual consistency.

Accelerating innovation needs complete information and data-led decisions

For federation-based integration, OpsHub is the only solution. OpsHub Integration Manager (OIM) creates an enterprise federation fabric and ensures the availability of complete, rich interconnected information in each person's tool of choice. This includes core information (information you own), shared information (information owned jointly between teams/tools), extended information (information owned by other teams but needed for decisions) and enhanced information created by combining domain-specific knowledge and insights with historical information across multiple systems. OpsHub's unique bidirectional sync mechanism was designed to solve the federation problem and is the only tool ensuring data consistency across disparate systems.



By breaking down information silos, OpsHub empowers innovation teams to make data-driven, informed decisions, and to take action to deliver higher quality products sooner.

If you would like to flesh out the details of your specific federation use case, our **solutions expert** will be happy to help.

Web - www.opshub.com | Email - sales@opshub.com | Phone - +1.650.701.1800