ESB (Enterprise Service Bus)

By: **IBM Cloud Education**

7 April 2021

Integration

What is an ESB?

ESB (Enterprise Service Bus)

In this guide, learn more about the ESB (an essential component of SOA), the benefits it offers, and how it relates to microservices architecture.

An ESB, or enterprise service bus, is an architectural pattern whereby a centralized software component performs integrations between applications. It performs transformations of data models, handles connectivity, performs message routing, converts communication protocols and potentially manages the composition of multiple requests. The ESB can make these integrations and transformations available as a service interface for reuse by new applications.

The ESB pattern is typically implemented using a specially designed integration runtime and toolset (i.e., esb product) that ensures the best possible productivity.

ESB and SOA

An ESB is an essential component of SOA, or service-oriented architecture, a software architecture that emerged in the late 1990s. SOA defines a way to make software components reusable via service interfaces. These services typically use standard interfaces (i.e., web services) in such a way that they can be rapidly incorporated into new applications without having to duplicate the functionality performed by the service in new applications.

Each service in an SOA embodies the code and *data* required to execute a complete, discrete business function (e.g. checking a customer's credit, calculating a monthly loan payment, or processing a mortgage application). The service interfaces provide loose coupling, meaning they can be called with little or no knowledge of how the service is implemented underneath, reducing the dependencies between applications. Applications behind the service interface can be written in Java, Microsoft .Net, Cobol or any other programming language, supplied as packaged enterprise applications by a vendor (e.g., SAP), SaaS applications (e.g., Salesforce CRM), or obtained as open source applications.

Service interfaces are frequently defined using Web Service Definition Language (WSDL) which is a standard tag structure based on xml (extensible markup language). The services are exposed using standard network protocols—such as

processes.

Services can be built from scratch but are often created by exposing functions from legacy systems of record. Businesses can choose to provide a standards based service interface in front of the legacy systems, use the ESB to directly connect to the legacy system through an adapter or connector, or the application may provide its own api. In any case the enterprise service bus shields the new application from the legacy interface. An ESB performs the necessary transformation and routing to connect to the legacy system service.

It is possible to implement a SOA without an ESB architecture, but this would be equivalent to just having a bunch of services. Each application owner would need to directly connect to any service it needs and perform the necessary data transformations to meet each of the service interfaces. This is a lot of work (even if the interfaces are reusable) and creates a significant maintenance challenges in the future as each connection is point to point.

Benefits

In theory, a centralized ESB offers the potential to standardize—and dramatically simplify—communication, messaging, and integration between services across the enterprise. Hardware and software costs can be shared, provisioning the servers as needed for the combined usage, providing a scalable centralized solution. A single team of specialists can be tasked (and, if necessary, trained) to develop and maintain the integrations.

Software applications simply connect (‘talk’) to the ESB and leave it to the ESB to transform the protocols, route the messages, and transform into the data formats as required providing the interoperability to get the transactions executed. The enterprise service bus architectural approach supports scenarios for application integration, data integration, and service orchestration style automation of business processes. This enables developers to spend dramatically less time integrating and much more time focusing on delivering and improving their applications. And the ability to reuse these integrations from one

organizations the ESB came to be seen as the bottleneck. Making changes or enhancements to one integration could destabilize others who used that same integration. Updates to the ESB middleware often impacted existing integrations, so there was significant testing required to perform any update. Because the ESB was centrally managed, application teams soon found themselves waiting in line for their integrations. As the volume of integrations grew, implementing high availability and disaster recovery for the ESB servers became more costly. And as a cross-enterprise project, the ESB proved difficult to fund, making these technical challenges that much more difficult to resolve.

Ultimately the challenges of maintaining, updating, and scaling a centralized ESB proved so overwhelming and expensive that the ESB often delayed the very productivity gains that it, and the SOA, were intended to yield, frustrating line of business teams who anticipated a greater pace of innovation.

For a deeper dive into the rise and fall of the ESB, read "The fate of the ESB."

ESB and microservices

Microservices architecture enables the internals of a single application to be broken up into small pieces that can be independently changed, scaled, and administered. Microservices emerged and gained steam with the rise of virtualization, cloud computing, Agile development practices, and DevOps. In these contexts, microservices offer the following:

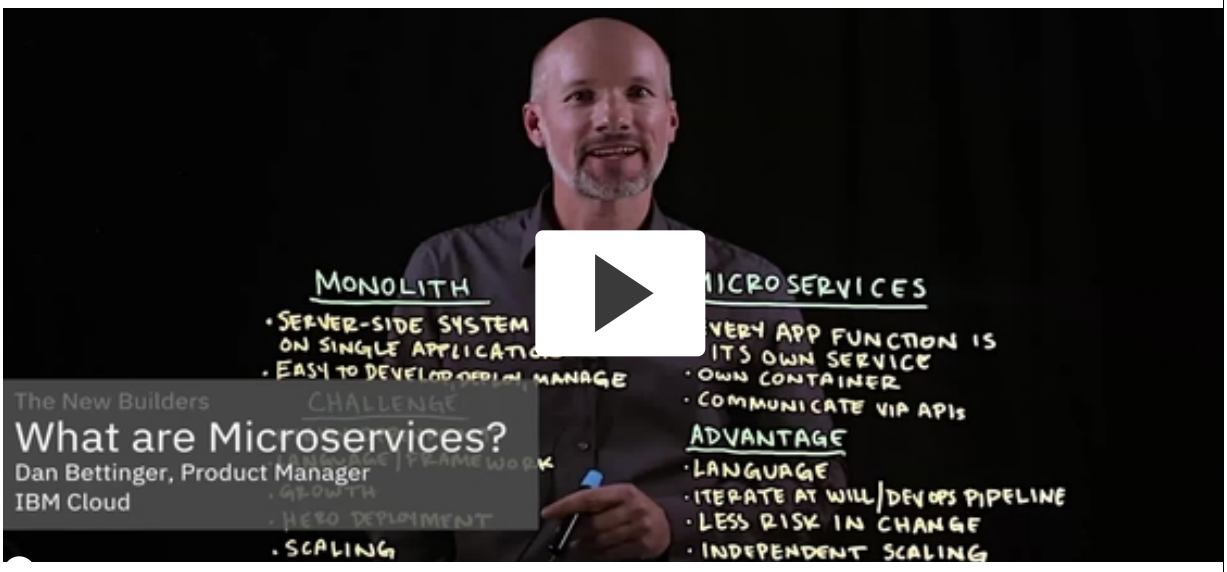
- **Improved developer agility and productivity** by enabling developers to incorporate new technologies into one part of an application without touching or 'catching up' the rest of the application.

- **Simpler, more cost-effective scalability** by enabling any component to be scaled independently of others, for the fastest possible response to workload demands and the most efficient use of computing resources.

The same granularity that microservices bring to application design can be brought to integration, with similar benefits. This is the idea behind agile integration, which breaks up the ESB into fine-grained, decentralized integration components, without interdependencies, that individual application teams can own and manage themselves.

For a deeper dive into all things microservices, check out "Microservices: A Complete Guide," "SOA vs. Microservices: What's the difference?" and watch Dan Bettinger's video, "What are Microservices?":

What are Microservices



ESB and IBM Cloud

As your company shifts its IT infrastructure toward a hybrid cloud approach, there's a high likelihood you'll be transforming a variety of workloads, including those based on SOA and ESB patterns, to more lightweight and flexible deployment models.

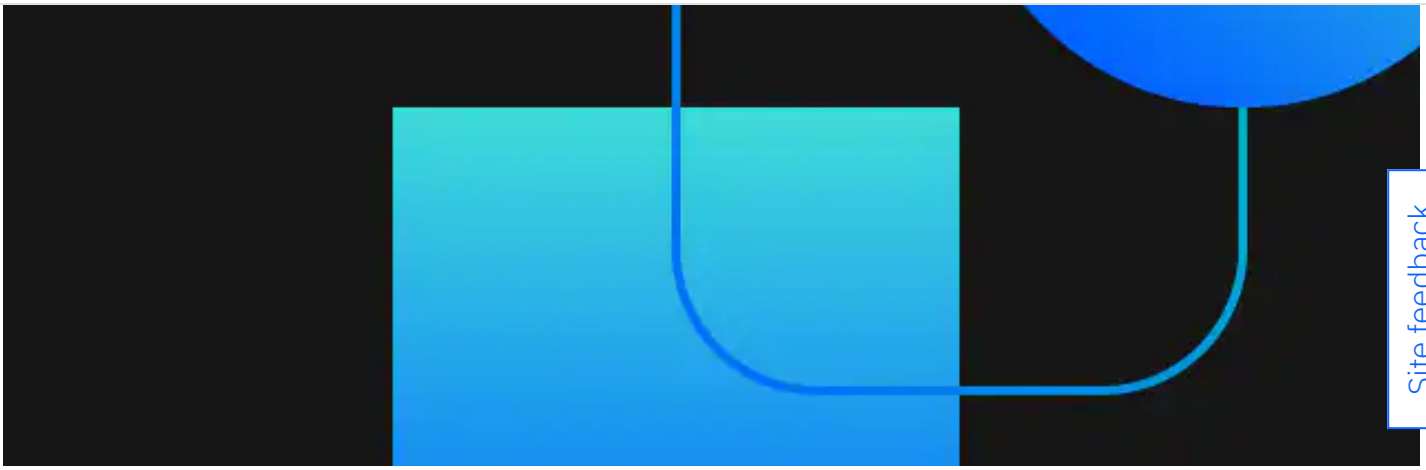
projects, which you can then scale and optimize for other processes and in other parts of your organization.

Working with IBM, you'll have access to AI-powered automation capabilities, including prebuilt workflows, to help accelerate innovation by making every process more intelligent.

Take the next step:

- See how you can leverage, extend and modernize your middleware investments with IBM Cloud Pak for Integration, a hybrid integration solution that provides an automated and closed-loop lifecycle across multiple styles of enterprise integration.
- Learn how you can connect all of your applications and data across multiple private and public clouds for personalized customer experiences by viewing IBM Cloud integration solutions.
- Get the IBM Application Modernization Field Guide to learn how to accelerate modernization, improve developer productivity and enhance operational efficiency and standardization.
- Take our integration maturity assessment to evaluate your integration maturity level across critical dimensions and discover the actions you can take to get to the next level.
- Download our agile integration guide, which explores the merits of a container-based, decentralized, microservices-aligned approach for integrating solutions.
- Read about the five "must-haves" for automation success (link resides outside IBM) in this HFS Research report.

Get started with an IBM Cloud account today.



Featured products

IBM Cloud Pak for Integration

App Connect

IBM API Connect

Why IBM Cloud

Why IBM Cloud

Hybrid Cloud approach

Trust and security

Open Cloud

Products and Solutions

Cloud Paks

Cloud pricing

View all products

View all solutions

Site feedback

Learn about

What is Hybrid Cloud?

What is Cloud Computing?

What is Confidential Computing?

What is a Data Lake?

What is a Data Warehouse?

What is Machine Learning?

What is DevOps?

What is Kubernetes?

What is Microservices?

Resources

Get started

Docs

Architectures

IBM Garage

Training and Certifications

Partners

Cloud blog

Let’s talk