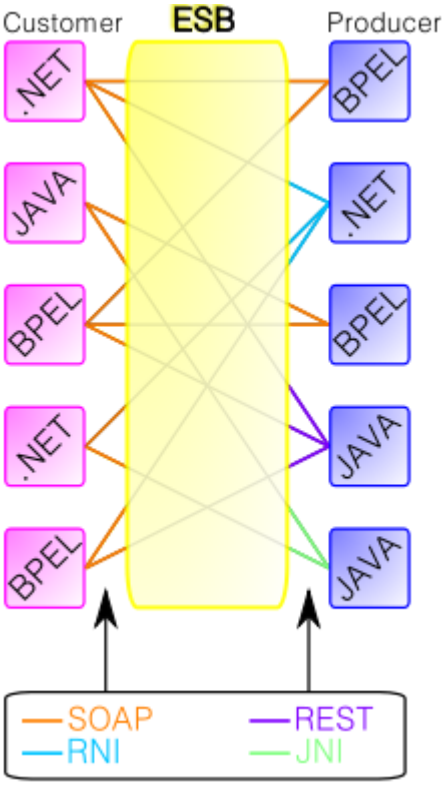
WIKIPEDIA

Enterprise service bus

An **enterprise service bus** (**ESB**) implements a communication system between mutually interacting software applications in a service-oriented architecture (SOA). It represents a software architecture for distributed computing, and is a special variant of the more general client-server model, wherein any application may behave as server or client. ESB promotes agility and flexibility with regard to high-level protocol communication between applications. Its primary use is in enterprise application integration (EAI) of heterogeneous and complex service landscapes.



All customer services communicate in the same way with the ESB: the ESB translates a message to the correct message type and sends the message to the correct consumer service.

Contents

Architecture

The concept of the enterprise service bus is analogous to the bus concept found in computer hardware architecture combined with the modular and concurrent design of high-performance computer operating systems. The motivation for the development of the architecture was to find a standard, structured, and general purpose concept for describing implementation of loosely coupled software components (called services) that are expected to be independently deployed, running, heterogeneous, and disparate within a network. ESB is also a common implementation pattern for service-oriented architecture, including the intrinsically adopted network design of the World Wide Web.

No global standards exist for enterprise service bus concepts or implementations.[1] Most providers of message-oriented middleware have adopted the enterprise service bus concept as *de facto* standard for a service-oriented architecture. The implementations of ESB use event-driven and standards-based

message-oriented middleware in combination with message queues as technology frameworks.[2] However, some software manufacturers relabel existing middleware and communication solutions as ESB without adopting the crucial aspect of a bus concept.

Functions

An ESB applies the design concept of modern operating systems to independent services running within networks of disparate and independent computers. Like concurrent operating systems, an ESB provides commodity services in addition to adoption, translation and routing of client requests to appropriate answering services.

The primary duties of an ESB are:

- Route messages between services
- Monitor and control routing of message exchange between services
- Resolve contention between communicating service components
- Control deployment and versioning of services
- Marshal use of redundant services
- Provide commodity services like event handling, data transformation and mapping, message and event queuing and sequencing, security or exception handling, protocol conversion and enforcing proper quality of communication service.

History

The first published usage of the term "enterprise service bus" is attributed to Roy W. Schulte from the Gartner Group 2002 and the book *The Enterprise Service Bus* by David Chappell. Although a number of companies take credit for coining the phrase, in an interview, Schulte said that the first time he heard the phrase was from a company named Candle and went on to say: "The most direct ancestor to the ESB was Candle's Roma product from 1998"[3] whose Chief Architect and patent application holder was Gary Aven. Roma was first sold in 1998 making it the first commercial ESB in the market, but that Sonic's product from 2002 was also one of the early ESBs on the market.[4]

- Service - denotes non-iterative and autonomously executing programs that communicate with other services through message exchange
- Bus - is used in analogy to a computer hardware bus
- Enterprise - the concept has been originally invented to reduce complexity of enterprise application integration within an enterprise; the restriction has become obsolete since modern Internet communication is no longer limited to a corporate entity

ESB as software

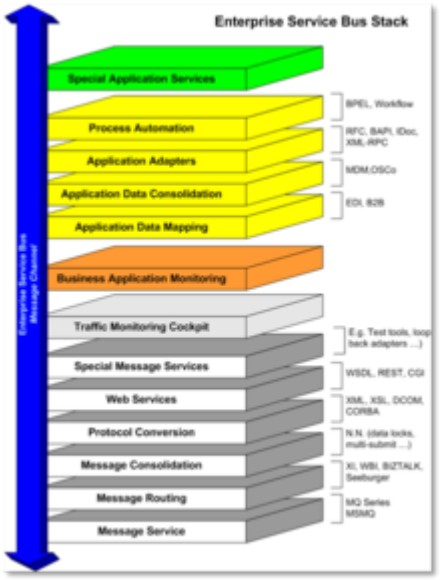
The ESB is implemented in software that operates between the business applications, and enables communication among them. Ideally, the ESB should be able to replace all direct contact with the applications on the bus, so that all communication takes place via the ESB. To achieve this objective, the ESB must encapsulate the functionality offered by its component applications in a meaningful way. This typically occurs through the use of an enterprise message model. The message model defines a standard set of messages that the ESB transmits and receives. When the ESB receives a message, it routes the

message to the appropriate application. Often, because that application evolved without the same message model, the ESB has to transform the message into a format that the application can interpret. A software adapter fulfills the task of effecting these transformations, analogously to a physical adapter.[5]

ESBs rely on accurately constructing the enterprise message model and properly designing the functionality offered by applications. If the message model does not completely encapsulate the application functionality, then other applications that desire that functionality may have to bypass the bus, and invoke the mismatched applications directly. Doing so violates the principles of the ESB model, and negates many of the advantages of using this architecture.

The beauty of the ESB lies in its platform-agnostic nature and the ability to integrate with anything at any condition. It is important that Application Lifecycle Management vendors truly apply all the ESB capabilities in their integration products while adopting SOA. Therefore, the challenges and opportunities for EAI vendors are to provide an integration solution that is low-cost, easily configurable, intuitive, user-friendly, and open to any tools customers choose.

Characteristics



ESB hive of commodity components

Category	Functions
Invocation	support for synchronous and asynchronous transport protocols, service mapping (locating and binding)
Routing	addressability, static/deterministic routing, content-based routing, rules-based routing, policy-based routing
Mediation	adapters, protocol transformation, service mapping
Messaging	message-processing, message transformation and message enhancement
Process choreography[1]	implementation of complex business processes
Service orchestration[2]	coordination of multiple implementation services exposed as a single, aggregate service
Complex event processing	event-interpretation, correlation, pattern-matching
Other quality of service	security (encryption and signing), reliable delivery, transaction management
Management	monitoring, audit, logging, metering, admin console, BAM (BAM is not a management capability in other words the ESB doesn't react to a specific threshold. It is a business service capability surfaced to end users.)
Agnosticism	general agnosticism to operating-systems and programming-languages; for example, it should enable interoperability between Java and .NET applications
Protocol Conversion	comprehensive support for topical communication protocols service standards
Message Exchange Patterns	support for various MEPs (Message Exchange Patterns) (for example: synchronous request/response, asynchronous request/response, send-and-forget, publish/subscribe)
Adapters	adapters for supporting integration with legacy systems, possibly based on standards such as JCA
Security	a standardized security-model to authorize, authenticate and audit use of the ESB
Transformation	facilitation of the transformation of data formats and values, including transformation services (often via XSLT or XQuery) between the formats of the sending application and the receiving application
Validation	validation against schemas for sending and receiving messages
Governance	the ability to apply business rules uniformly
Enrichment	enriching messages from other sources
Split and Merge	the splitting and combining of multiple messages and the handling of exceptions
Abstraction	the provision of a unified abstraction across multiple layers
Routing and Transformation	routing or transforming messages conditionally, based on a non-centralized policy (without the need for a central rules-engine)
Commodity Services	provisioning of commonly used functionality as shared services depending on context

[1] *Some do not regard process choreography as an ESB function. For example, see M.Richards.*[6]

[2] *While process choreography supports implementation of complex business processes that require coordination of multiple business services (usually using BPEL), service orchestration enables coordination of multiple implementation services (most suitably exposed as an aggregate service) to serve individual requests.*

These solutions often focus on low-level ESB functions, such as connectivity, routing and transformation, and require coding or scripting to implement orchestration.[7] Developers operating at a project or tactical level, e.g., just trying to fix a problem, often gravitate toward lightweight service bus technologies, but there is often ongoing tension between these initiatives and an enterprise architecture whose goal it is to optimize infrastructure across multiple projects.[8]

If the message broker, the ESB software, translates a message from one format to another, then as with any translation, there is the issue of semantics of the message. For example, a record can be translated from JSON to XML, but the same set of fields can be interpreted differently by different applications, specifically in the case of the various corner cases that are usually known only to developers that have extensive experience with the application that is connected to the ESB. For the known corner cases the number of tests that cover all corner cases increases exponentially with every application that is connected to the ESB, because every ESB-connected application must be tested against every other application that is connected to the ESB.

Key benefits

- Scales from point-solutions to enterprise-wide deployment (distributed bus)
- More configuration rather than integration coding
- No central rules-engine, no central broker
- Easy plug-in and plug-out and loosely coupling system

Key disadvantages

- Slower communication speed, especially for those already compatible services
- Single point of failure, can bring down all communications in the Enterprise
- High configuration and maintenance complexity

Products

Notable products include:

- Proprietary
  - Candle's Roma ESB - bought by IBM and became WebSphere ESB
  - IBM App Connect, formerly IBM Integration Bus and IBM WebSphere ESB
  - InterSystems Ensemble
  - Information Builders iWay Service Manager
  - Microsoft Azure Service Bus
  - Microsoft BizTalk Server
  - Mule ESB
  - Oracle Enterprise Service Bus
  - Progress Software Sonic ESB (acquired by Trilogy)
  - SAP Process Integration
  - TIBCO Software ActiveMatrix BusinessWorks
  - webMethods enterprise service bus (acquired by Software AG)
  - Sonic ESB from Aurea

- Open-source software
  - Apache Camel
  - Apache ServiceMix
  - Apache Synapse
  - Fuse ESB from Red Hat
  - JBoss ESB
  - NetKernel
  - Open ESB
  - Petals ESB
  - Spring Integration
  - UltraESB
  - WSO2 ESB

See also

- Enterprise Integration Patterns
- Event-driven messaging
- Java Business Integration
- Business Process Management
- Universal Integration Platform
- Enterprise application integration
- Business Service Provider
- Message Oriented Middleware
- Complex event processing
- Event Stream Processing
- Event-driven programming
- Comparison of Business Integration Software
- Comparison of BPEL engines
- Comparison of BPMN 2.0 Engines
- Composite application
- Event-driven SOA
- Integration Platform as a service (iPaaS)

References

1. Lapeira, Raul. "ESB is an architectural style, a software product, or a group of software products?" (https://web.archive.org/web/20140808053149/http://www.consultoriajava.com/articulos/esb_arquitectura_software_product.jsp). Artifact Consulting. Archived from the original (http://www.consultoriajava.com/articulos/esb_arquitecture_software_product.jsp) on 2014-08-08. Retrieved 2010-04-16. "The first thing a ESB architect should have in mind is that as of 2010 there is no global standard for ESB."
2. Curry, Edward. 2004. "Message-Oriented Middleware" (http://www.mendeley.com/download/public/1652511/4338215212/cce0f06f047faa57879a1fc36a8e8d6d754d2f6a/dl.pdf). In *Middleware for Communications*, ed. Qusay H. Mahmoud, 1-28. Chichester, England: John Wiley and Sons. doi:10.1002/0470862084.ch1 (https://doi.org/10.1002%2F0470862084.ch1). ISBN 978-0-470-86206-3

3. McKendrick, Joe. "The great ESB squabble of 2005" (https://www.zdnet.com/article/the-great-esb-squabble-of-2005/). *ZDNet*. Retrieved 2020-12-31.
4. "Difference between a Message Broker and an ESB" (https://stackoverflow.com/questions/773503/difference-between-a-message-broker-and-an-esb). Retrieved 2017-07-19.
5. http://shop.oreilly.com/product/9780596006754.do
6. Richards, Mark. "The Role of the Enterprise Service Bus (presentation)" (http://www.infoq.com/presentations/Enterprise-Service-Bus). Retrieved 2009-06-04. "I do not consider process choreography part of an ESB, if we consider an ESB as a high-speed messaging middleware. However, I do consider process choreography part of the ESB *platform*. Fortunately most ESB vendors separate out these major components into different products, but package them under a consolidated ESB offering. So, in the strictest sense of the word, no, I would not consider it as part of an ESB. It is a related capability."
7. Feraga, Matthias (6 Jun 2011). "How to: choosing between lightweight and traditional ESBs" (http://blog.octo.com/en/choosing-between-lightweight-and-traditional-esbs/). Octo. Retrieved 23 April 2014.
8. Fulton, Larry (12 Sep 2007). "Learn How to Embrace Lightweight ESBs" (http://www.forrester.com/Learn+How+To+Embrace+Lightweight+ESBs/fulltext/-/E-RES43339). Fo2014.

Further reading

- David Chappell, "Enterprise Service Bus" (O'Reilly: June 2004, ISBN 0-596-00675-6)
- Binildas A. Christudas, "Service-oriented Java Business Integration" (Packt Publishers: February 2008, ISBN 1-84719-440-0; ISBN 978-1-84719-440-4)
- Michael Bell, "Service-Oriented Modeling: Service Analysis, Design, and Architecture" (2008 Wiley & Sons, ISBN 978-0-470-14111-3)

External links

- "Lasting concept or latest buzzword?" (http://searchwebservices.techtarget.com/tip/1,289483,sid26_gci913058,00.html) (Nicolas Farges, 2003)
- Enterprise service buses hit the road: Infoworld Test Center (http://www.infoworld.com/article/2671356/application-development/enterprise-service-buses-hit-the-road.html) (July 22, 2005)
- JSR-208: Java Business Integration (http://www.jcp.org/en/jsr/detail?id=208) (August 2005)
- The Role of the Enterprise Service Bus (InfoQ - Video Presentation) (http://www.infoq.com/presentations/Enterprise-Service-Bus) (October 23, 2006)
- ESB Roundup Part One: Defining the ESB (InfoQ) (http://www.infoq.com/articles/ESB-Roundup-Part1-Defining-ESB) (July 13, 2006)
- ESB Roundup Part Two: Use Cases (InfoQ) (http://www.infoq.com/news/ESB-Roundup-Part-Two--Use-Cases) (July 5, 2006)
- "Services Fabric—Fine Fabrics for New Era Systems" (https://web.archive.org/web/20071219235821/http://www.iasahome.org/c/portal/layout?p_l_id=PUB.1.266) (Binildas A. Christudas, 2007)
- "ESBs in 2007: Taking the Open Source Bus to SOA" (http://www.ebizq.net/hot_topics/esb/features/8480.html) (Dennis Byron, September 20, 2007)
- Aggregate Services in ServiceMix JBI ESB: PACKT Publishers (http://www.packtpub.com/article/aggregate-services-in-servicemix-jbi-esb) (Binildas A. Christudas, November 30, 2007)
- ESB Topology alternatives (http://www.infoq.com/articles/louis-esb-topologies) (InfoQ, A. Louis, May 23, 2008)
- Rethinking the ESB: Building a Simple, Secure, Scalable Service Bus with a SOA Gateway (http://www.computerworld.com/s/article/9219205/Rethinking_the_ESB_Building_a_simple_secure_scalable

_Service_Bus_with_an_SOA_Gateway?taxonomyId=157&pageNumber=1) (Computerworld, J. Ryan, 2011)
- Louis, Adrien; Marc Dutoo (2010-07-02). "Choosing between Routing and Orchestration in an ESB" (http://www.infoq.com/articles/louis-dutoo-esb-routing). *InfoQ*. Retrieved 2009-07-02.
- The Enterprise Service Bus, re-examined (http://www.ibm.com/developerworks/websphere/techjournal/1105_flurry/1105_flurry.html) (IBM developer Works, Greg Flurry and Kim Clark, May 2011)
