

# OPEN SOURCE MIDDLEWARE REFERENCE ARCHITECTURE

LEVERAGING JBOSS ENTERPRISE MIDDLEWARE ACROSS YOUR  
ENTERPRISE TO INCREASE PRODUCTIVITY AND REDUCE COSTS

---

- 2 EXECUTIVE SUMMARY
- 3 REFERENCE ARCHITECTURES
- 4 APPLICATION AND SERVICE RUNTIME
- 6 PROCESS MANAGEMENT AND SERVICE INTEGRATION
- 8 DATA INTEGRATION AND BUSINESS INTELLIGENCE
- 9 USER INTERACTION SERVICES
- 10 INTEGRATED DEVELOPMENT TOOLING  
AND SYSTEMS MANAGEMENT
- 11 SUMMARY

## EXECUTIVE SUMMARY

---

Open source software has proven its value in companies of all sizes. As large enterprises have become increasingly comfortable with the open source model, exemplified by the Red Hat® Enterprise Linux® operating system, they have also begun adopting open source middleware for developing and deploying mission-critical applications. With continuing technology development that benefits from the innovative open source community, open source providers such as Red Hat continue to deliver new alternatives for additional levels of the software stack. These encompass not only operating systems, but also middleware for application hosting, application integration, and data integration, including the development tools, management tools, and monitoring tools to fully support the stack.

With the acquisition of JBoss in 2006, Red Hat entered the Java middleware market. Today Red Hat goes far beyond Java application containers (JBoss® Enterprise Application Platform) to provide a broad set of middleware capabilities, including:

- Process management
- Service integration and high-speed messaging (JBoss Enterprise SOA Platform)
- Data integration (JBoss Enterprise Data Services Platform)
- User interaction services (JBoss Enterprise Portal Platform)
- Systems management and monitoring (JBoss Operations Network)
- Integrated development tooling (JBoss Developer Studio)

In addition to these currently available platforms, the JBoss.org community is developing additional infrastructure components, including metadata management, governance, and security. Beyond extending support to multiple layers of the stack, open source software also provides advantages for a wide range of applications and development approaches. Red Hat's current generation of open source Java middleware provides support for multiple Java application development techniques, recognizing that while some application development efforts require J2EE/EJB, others are better suited to more nimble programming languages and lighter-weight Java platforms.

Each JBoss component captures the best thinking of a vibrant open source community to deliver a powerful and reliable set of middleware capabilities. In addition, the set of components works together to deliver an integrated platform that is much greater than the sum of its parts. These platforms have been tested for interoperability and certified, allowing organizations to deploy new applications more efficiently, confident that their platforms are supported by the industry-leading JBoss customer support team. Support of this caliber enables organizations to take advantage of the cost-effectiveness and value of open source software, while still leveraging mission-critical reliability and the full array of support services needed for large-scale enterprise applications.

This paper describes a comprehensive open source middleware reference architecture, an implementation of which is exemplified by the JBoss Enterprise Middleware portfolio. The ultimate goal of this reference architecture is to provide a roadmap of open source software capabilities across a diverse set of application requirements. In each focus area of this reference architecture, JBoss Enterprise Middleware provides the foundation for openness, flexibility, and scalability that enterprises need to develop and deploy applications efficiently, make the most of their data, implement SOA, and ensure efficient operations.



## REFERENCE ARCHITECTURES

---

The open source reference architecture that Red Hat provides and supports via JBoss Enterprise Middleware is comprehensive in scope, ranging from Java application development initiatives to business process management to data integration to presentation layer services that visually tie application components together for the user.

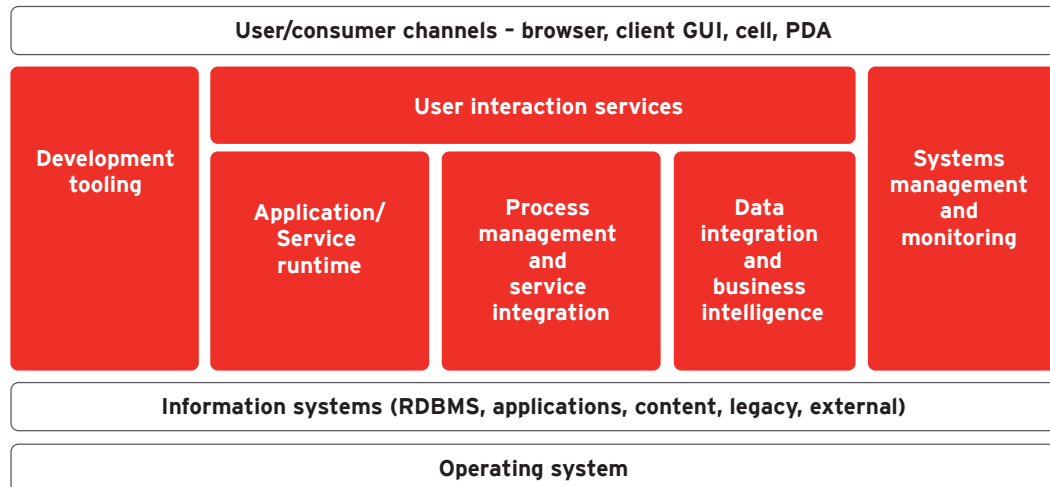
As seen in the high-level diagram below, the reference architecture includes the following focus areas:

- Application/Service runtime – This area is focused on the basic-yet-critical requirement to host the applications that “run the business,” including both custom-developed applications and packaged solutions that are purchased.
- Process management and service integration – This area is focused on the problem of tying many separate applications inside an organization into a set of more streamlined business processes
- Data integration and business intelligence – This area is focused on bringing the data from many separate application databases back together, in order to provide consolidated views of critical information such as customer and partner relationships.
- User interaction services – This area is focused on enabling and managing user access to the multiple applications, databases, and content that users need to get their job done more efficiently.

The JBoss reference architecture incorporates valuable lessons from early adopters and competitors, such as the need for interoperability and integration of its solutions across these focus areas. This clearly includes the need for common components – such as caching, persistence, transactions, and clustering – to be re-used across platforms. But it also extends to the development environment, where standard (Eclipse-based) tooling is necessary to maintain a high level of developer productivity and to minimize version mismatches between development, test, and production environments. Interoperability and integration also extend to the management environment, where consistent configuration management capabilities and monitoring tools are critical to lowering the cost of ownership during the typically long operational phase following an application’s initial development phase.

JBoss is unique in its ability to provide open source solutions that span all of the focus areas shown in the diagram below. By combining an open source approach with support for open standards, JBoss Enterprise Middleware is able to provide a technology solution in each focus area that enables enterprises to avoid vendor lock-in. As a result of taking this approach, enterprises are free to leverage JBoss implementations of a particular standard or to substitute their own or another vendor’s implementation (whether proprietary or open) where they see fit. Given the scope of technology encompassed by each focus area, it’s inevitable that industry standards and expectations of features will change over time. The JBoss approach provides enterprises with technology options at every phase of an application’s lifecycle, while helping them avoid the economic lock-in and narrow range of choices inherent to proprietary offerings.

## OPEN SOURCE MIDDLEWARE REFERENCE ARCHITECTURES



The detailed architectures for each of these focus areas (shaded in red) vary greatly, and are described further in each of the following sections.

## APPLICATION AND SERVICE RUNTIME

In the world of Java application development, there has always been a natural tension between application developers and operations staff. Application developers need to adopt the latest programming techniques and frameworks. The operations staff needs to provide a stable operating environment for those same applications.

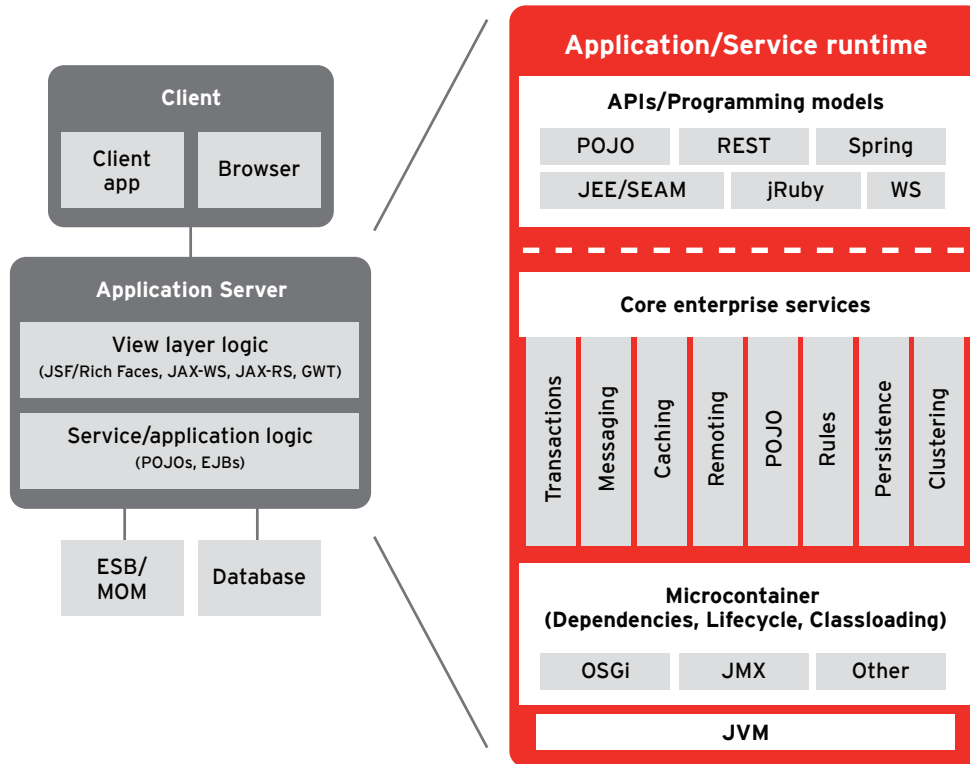
Developers need agility and choice in their application development frameworks, languages, and programming paradigms. The developer's ideal operating environment requires the flexibility to support more versions more often. But system administrators want stability and predictability. They don't want developer choices to disrupt operational procedures.

The reference architecture implemented by JBoss (shown below) benefits both developers and administrators by creating developer agility without operational disruption. This is in contrast to other application server architectures that fail to decouple the operational aspects from the developer aspects. This gives rise to specialized approaches (e.g. IIS/Tomcat for simple web applications, JEE for moderate applications, TPMs for high-end transaction processing).

The JBoss reference architecture provides instead for one core run-time that has the ability to support multiple application "personalities." The architecture provides support for standards that are popular today (e.g., JEE, OSGi) but is flexible enough to support tomorrow's programming models, frameworks, and component models.



## APPLICATION AND SERVICE RUNTIME



This new architecture's main benefit for the enterprise is that one core platform can provide support for all of an organization's Java applications. It will be able to support a more broad open source application stack, including popular web frameworks like Spring, Google Web Toolkit, Struts, or RichFaces. The footprint can be customized for specific application requirements, including:

- Basic and simple servlet deployments
- Rich Internet applications (e.g., EE 6 "Medium" profile)
- Lightweight Java applications
- Enterprise mission-critical applications, including support for EJBs, clustering, transactions, messaging, and web services

The new architecture of the JBoss Enterprise Application Platform (EAP) was created specifically to host a wider variety of Java workloads and support other languages to run on the VM. JBoss EAP provides a lightweight server runtime environment that is modular (choose only the components you need), extensible (add your own services and deployment types), and dynamic (add/remove components at runtime with no pre-compilation steps). In the past, this was achieved via a JMX-based kernel, but the current architecture has a POJO-based kernel (the Microcontainer) that has a smaller footprint. It also includes new aspectized deployers, a new configuration API, and a new class-loading approach that supports versioning.

## PROCESS MANAGEMENT AND SERVICE INTEGRATION

---

Today, many organizations are faced with increasingly rapid change in their business requirements. To respond, they're striving to implement more agile service-oriented architectures for their information systems. Although it is difficult to quantify the number of enterprises that have successfully implemented SOA, it is clear that SOA has evolved enough for most to agree on its basic technology components.

SOA implementations are often targeted at executing business processes and responding to business events in a faster, more consistent manner. A process orchestration component is therefore typically required. The individual steps of a business process require the execution of business logic, which may include the creation of new business-rule services or the encapsulation of that logic in existing code or web services.

An enterprise service bus component provides a set of flexible messaging features that enable processes and business logic to communicate with new and existing systems. Data service components can provide re-usable, abstracted interfaces to data in existing systems, hiding the complex data structures of those systems from higher-level components. Adapters and other connectivity components provide the basic connections to existing databases, custom and packaged applications, legacy systems, and external partners. These layered components are able to leverage other common components, such as message transformation, security, and registry functions, as shown in the generic representation of a SOA stack on the left side of the diagram below.

The agility that SOA promises requires flexibility—not just in the features of the technology, but also flexibility in the choice of vendor technologies to implement the various components of the SOA stack. Many organizations that have begun to implement SOA have either been forced to build much of it themselves—due to the prohibitively high cost of today's commercial SOA offerings—or have found themselves laboring under inflexible, complex, and expensive SOA software suites.

The reference architecture has brought together a combination of business process management, business rules, service registry, message mediation, and enterprise service bus capabilities. The architecture provides the capabilities and the open standards support that are needed for an enterprise SOA solution, as shown in the right side of the diagram below. The JBoss Enterprise SOA Platform provides a cohesive solution for this architecture's requirements, and with its greater cost-effectiveness and open source approach, offers a way to reduce the financial and technical pain created by closed source, monolithic platforms.

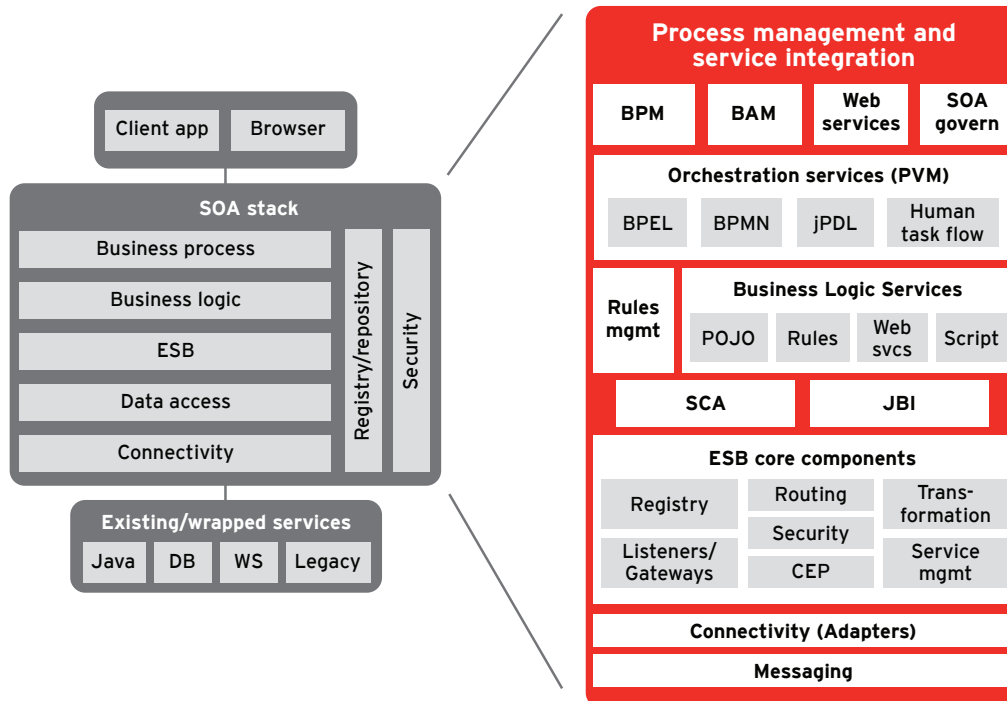
The JBoss Enterprise SOA Platform provides a next-generation enterprise service bus (ESB). It includes the necessary capabilities of a high-performance, reliable messaging infrastructure that provides protocol gateways and adapters with embedded routing and transformation. But it also extends beyond these core capabilities by bundling process management, human workflow, business rules, security, governance, and service registry functions with the ESB. This combined functionality seamlessly blends logic from web applications, legacy systems, and databases in new SOA-based applications. It then extends those services with new logic for business process orchestration, business rules, and event-driven message correlation. In addition, because of the ready access to the open source components, the JBoss Enterprise SOA Platform is paired with a vibrant partner ecosystem that magnifies the value of the technology itself.



The open source model encourages a greater degree of openness and flexibility than traditional, closed source platforms. This open architecture and open source model redefines “SOA” as “simple, open, and affordable.” The platforms, frameworks, and components are simpler to procure and deploy, because JBoss has taken an approach that allows developers, ISVs, and enterprises to implement solutions using only the components required, without having to learn the details of the entire platform. This enables organizations to create solutions that realize the benefits of SOA more quickly.

Open source software is more malleable and flexible than closed-source alternatives, and the JBoss SOA Enterprise Platform delivers standards-based service infrastructure that further supports its open focus and flexibility. Regarding affordability, JBoss subscriptions eliminate expensive license fees and deliver higher quality developer assistance and production support in SOA deployments. The benefit to the typical enterprise is the ability to reallocate budget for higher value activities, ensuring more successful deployments with superior customer satisfaction.

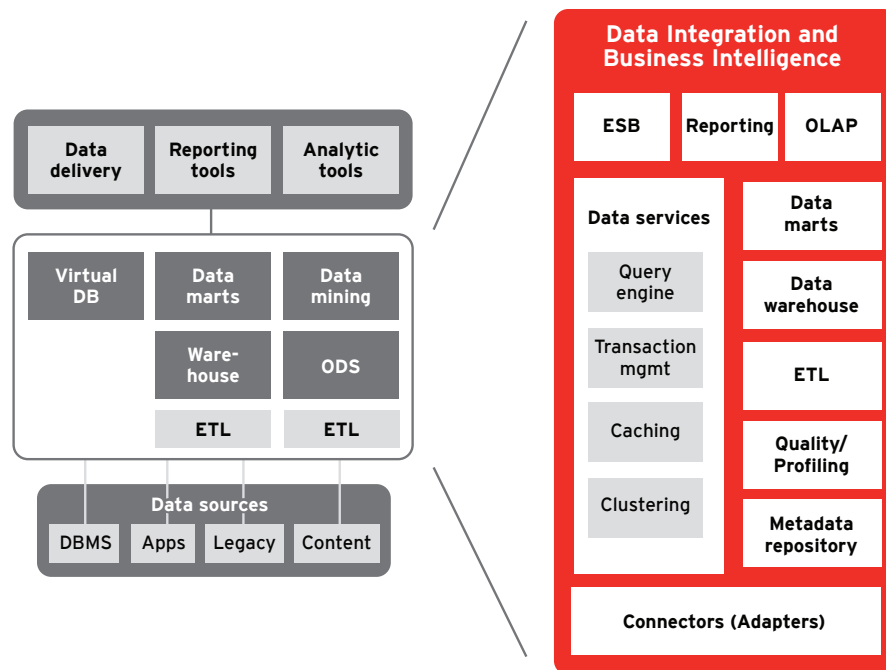
## PROCESS MANAGEMENT AND SERVICE INTEGRATION



# DATA INTEGRATION AND BUSINESS INTELLIGENCE

When multiple databases must be used together, organizations need to bridge a number of different technical and semantic gaps to get the data they need. Organizations have increasing data volumes, increasing needs for consolidated data views to drive real-time business operations, and increasing drive toward interoperability and standards support. The ability to bridge data gaps in a more straight-forward, streamlined, and scalable way is becoming an urgent need.

## DATA INTEGRATION AND BUSINESS INTELLIGENCE



Each of the data integration components identified in the reference architecture above is necessary for a comprehensive approach to data management and data integration. While the ETL style of data integration is used to create a consolidated copy of the source data, data services typically provide data integration in real-time without making a copy of the integrated result sets. Data services are key components of a flexible architecture that provides applications with the data views they need while hiding federated data source details and managing data source access.

Historically some of these components have been proposed as “either/or” components – *either* data services tools *or* ETL tools, for example. The reality is that organizations need multiple data integration tools and approaches in order to meet various quality, latency, performance, and technology API requirements. In fact, data services capabilities can be combined with an organization’s existing BI and ETL investments to accelerate the movement of data architectures toward more service-oriented approaches. The requirement for accelerated access to critical business information has forced organizations to rethink their data integration strategies to accommodate both retrospective analyses and real-time business activity monitoring.

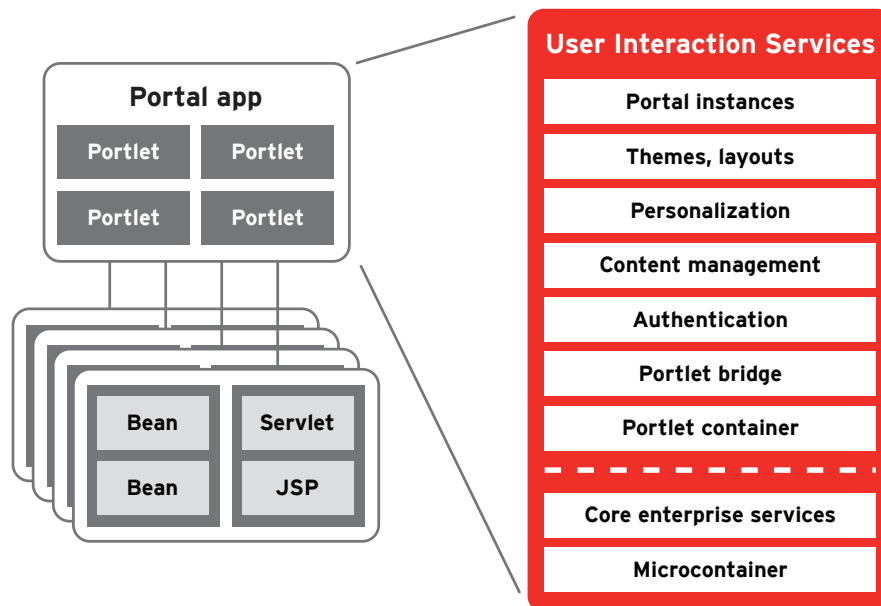


With the JBoss Enterprise Data Services Platform, Red Hat provides the capability to create, deploy, execute, and manage data services—collectively known as a data service management system. Creating data services (rather than custom coding data manipulations into each application) enables developers to bridge semantic gaps across federated data sources more efficiently and create reusable data access modules that isolate the impact of changes in data structures. JBoss provides a powerful data service management system that enables rapid creation, deployment, and management of data services and creates virtual data structures that meet the needs of the application and the business. As a result, applications and developers are insulated from the details of physical data sources.

## USER INTERACTION SERVICES

As organizations continue the drive to become more service-oriented, there's an associated drive toward improving the consumability of applications and services for end users. Lines of business will not be satisfied with the cost of SOA investment without enhanced end user services to achieve business objectives. They expect that reuse and modularity will occur at all levels, from site-wide to the component level, enabling the delivery of more personalized business applications. The population of power users within organizations will significantly expand as more technically savvy workers enter the workforce. This requires a new generation of user interfaces that allow power users to rapidly configure custom, time-sensitive dashboard applications without the need for an IDE or programming skills. In addition, growth in SaaS and cloud computing is driving a mix of on-premise and off-premise systems. All of these trends heighten the demand for portal platforms to provide a highly integrated user experience while promoting the reuse of common site elements, security, UI components, branding, and presentation layer infrastructure. The reference architecture that highlights these user interaction components is shown on the right in the diagram below.

### USER INTERACTION SERVICES

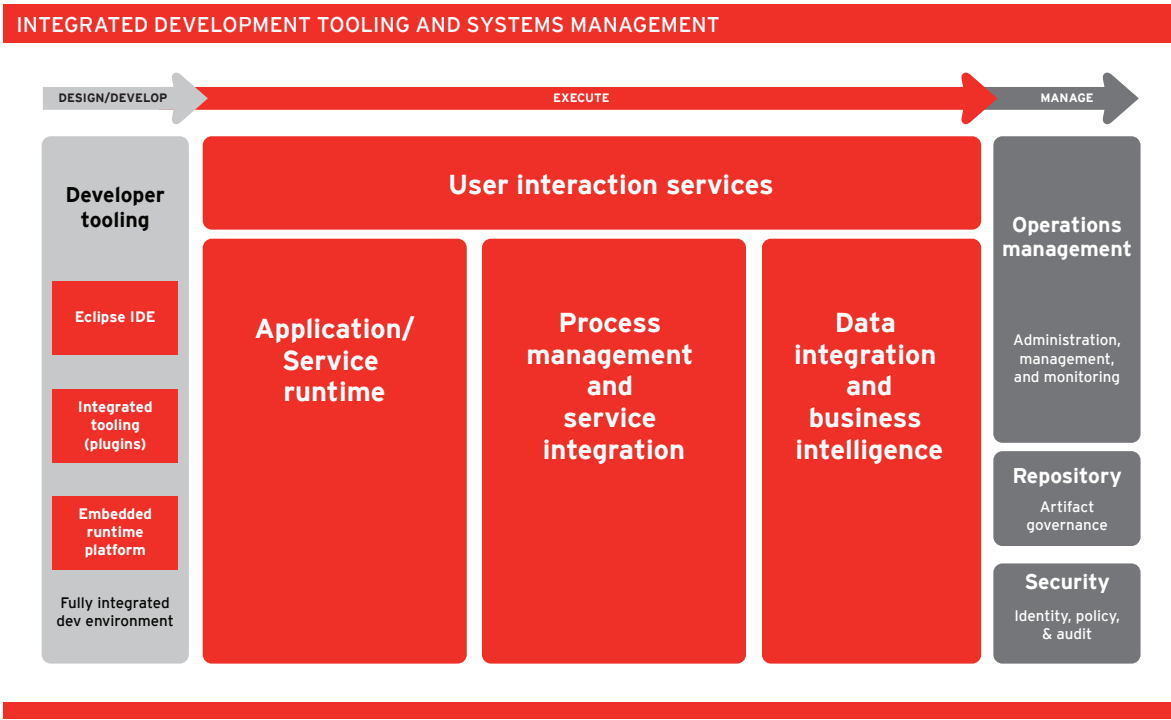


The JBoss Enterprise Portal Platform provides an open source platform for hosting and serving a portal's web interface, publishing and managing its content, and delivering personalized user experiences. While similar to other portal offerings in its ability to help enterprises launch portals more quickly, the JBoss Enterprise Portal Platform is unique in delivering the benefits of a fully-featured, open source solution combined with a flexible and scalable underlying platform. Portal Platform focuses on the features that enterprises really need, rather than on UI bells and whistles that make for great demonstrations, but aren't useful in actual deployments.

By focusing on portal standards, a lightweight footprint with simple installation, and scalable clustering to support high transaction volumes, the JBoss Enterprise Portal Platform can provide value at a much lower price point. Enterprises that deploy it gain the agility of an extensible site architecture that enables quicker site deployments and that can mash together legacy and new applications while taking advantage of new concepts within the fabric of their existing IT investments.

## INTEGRATED DEVELOPMENT TOOLING AND SYSTEMS MANAGEMENT

The significant breadth of the JBoss reference architecture implies that interoperability and integration of the component solutions in each focus area are fundamental concerns. Integrated development tooling and integrated systems management are two ways in which the JBoss reference architecture alleviates those concerns.



Consistent, integrated developer tooling begins with a standard, Eclipse-based development environment that maximizes developer productivity. The use of Eclipse enables extensibility via additional plug-ins as more heterogeneous technologies are required. By integrating the developer tooling with an instance of the runtime platform, JBoss minimizes the version mismatches between development, test, and production environments. This integration, coupled with the lightweight footprint of the runtime platform, provides the benefit of reduced iteration times between development and test efforts.

Integrated systems management is required to handle the administration and maintenance challenges found in typical datacenters, where applications are deployed across development, QA, staging, and production. The results of these challenges are applications that have multiple versions and potentially inconsistent configurations, access privileges, and release cycles. The architecture of the JBoss integrated management solution helps to solve these challenges by including a centralized management server and auto-discovery agents for the inventory of distributed resources. Organizations with complex environments benefit from administration features that include the start/stop deployment of services and applications, patch management, monitoring, and alerts. Because the JBoss reference architecture spans multiple technologies, the types of resources that are monitored range from low-level OS statistics and detailed application processes (e.g., EJBs, URL response times) to business processes, message queues, and data access requests. Data center operators benefit from this improved manageability and visibility, which provides more consistency and efficiency in managing application systems and results in lower cost of ownership. Organizations also benefit from the governance of system artifacts and security policies, ensuring that critical system information is stored, versioned, and audited, as well as retrievable by development, operations, and management teams.

## SUMMARY

---

This open source reference architecture is a maximally flexible, multi-purpose architecture. As with other reference architectures, it can provide a template for evaluating technology choices in each of the focus areas that have been highlighted, including application and service runtime, process management and service integration, data integration, and user interaction services. These templates can aid in evaluation efforts by providing a common vocabulary for the components, features, and functionality that are expected in vendor solutions. The reference architecture can also help promote best practices, to the extent that the architecture represents a high-level framework for organizations to follow in their implementations of new systems. Best practices may also be gleaned from open source and open standards, two approaches that maximize the enterprise's ability to control its own technology roadmap without relying too much on any one vendor. Finally, the reference architecture serves to highlight the extent to which open source software has evolved to satisfy a much broader set of enterprise middleware requirements.





## **JBoss SALES AND INQUIRIES** NORTH AMERICA

---

1-888-REDHAT1  
[www.jboss.com](http://www.jboss.com)