

# IT Infrastructure Architecture Building Blocks

---

*Ramesh Radhakrishnan, Sun Professional Services*

*Rakesh Radhakrishnan, Sun Professional Services*

*May 2004*

Copyright 2003 Sun Microsystems, Inc. 4150 Network Circle, Santa Clara, California 95045 U.S.A. All rights reserved.

Sun Microsystems, Inc. has intellectual property rights relating to technology embodied in the product that is described in this document. In particular, and without limitation, these intellectual property rights may include one or more of the U.S. patents listed at <http://www.sun.com/patents> and one or more additional patents or pending patent applications in the U.S. and in other countries.

This product or document is protected by copyright and distributed under licenses restricting its use, copying, distribution, and decompilation. No part of this product or document may be reproduced in any form by any means without prior written authorization of Sun and its licensors, if any. Third-party software, including font technology, is copyrighted and licensed from Sun suppliers.

Parts of the product may be derived from Berkeley BSD systems, licensed from the University of California. UNIX is a registered trademark in the United States and other countries, exclusively licensed through X/Open Company, Ltd.

Sun, Sun Microsystems, the Sun logo, Sun BluePrints, and Solaris are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. in the US and other countries. Products bearing SPARC trademarks are based upon an architecture developed by Sun Microsystems, Inc.

The OPEN LOOK and Sun™ Graphical User Interface was developed by Sun Microsystems, Inc. for its users and licensees. Sun acknowledges the pioneering efforts of Xerox in researching and developing the concept of visual or graphical user interfaces for the computer industry. Sun holds a non-exclusive license from Xerox to the Xerox Graphical User Interface, which license also covers Sun's licensees who implement OPEN LOOK GUIs and otherwise comply with Sun's written license agreements.

U.S. Government Rights—Commercial use. Government users are subject to the Sun Microsystems, Inc. standard license agreement and applicable provisions of the Far and its supplements.

DOCUMENTATION IS PROVIDED "AS IS" AND ALL EXPRESS OR IMPLIED CONDITIONS, REPRESENTATIONS AND WARRANTIES, INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT, ARE DISCLAIMED, EXCEPT TO THE EXTENT THAT SUCH DISCLAIMERS ARE HELD TO BE LEGALLY INVALID.

---

Copyright 2003 Sun Microsystems, Inc., 4150 Network Circle, Santa Clara, Californie 95045 Etats-Unis. Tous droits réservés.

Sun Microsystems, Inc. a les droits de propriété intellectuels relatants à la technologie incorporée dans le produit qui est décrit dans ce document. En particulier, et sans la limitation, ces droits de propriété intellectuels peuvent inclure un ou plus des brevets américains énumérés à <http://www.sun.com/patents> et un ou les brevets plus supplémentaires ou les applications de brevet en attente dans les Etats-Unis et dans les autres pays.

Ce produit ou document est protégé par un copyright et distribué avec des licences qui en restreignent l'utilisation, la copie, la distribution, et la décompilation. Aucune partie de ce produit ou document ne peut être reproduite sous aucune forme, par quelque moyen que ce soit, sans l'autorisation préalable et écrite de Sun et de ses bailleurs de licence, s'il y en a. Le logiciel détenu par des tiers, et qui comprend la technologie relative aux polices de caractères, est protégé par un copyright et licencié par des fournisseurs de Sun.

Des parties de ce produit pourront être dérivées des systèmes Berkeley BSD licenciés par l'Université de Californie. UNIX est une marque enregistrée aux Etats-Unis et dans d'autres pays et licenciée exclusivement par X/Open Company Ltd.

Sun, Sun Microsystems, le logo Sun, Sun BluePrints, et Solaris sont des marques de fabrique ou des marques déposées, ou marques de service, de Sun Microsystems, Inc. aux Etats-Unis et dans d'autres pays. Toutes les marques SPARC sont utilisées sous licence et sont des marques de fabrique ou des marques déposées de SPARC International, Inc. aux Etats-Unis et dans d'autres pays. Les produits portant les marques SPARC sont basés sur une architecture développée par Sun Microsystems, Inc.

L'interface d'utilisation graphique OPEN LOOK et Sun™ a été développée par Sun Microsystems, Inc. pour ses utilisateurs et licenciés. Sun reconnaît les efforts de pionniers de Xerox pour la recherche et le développement du concept des interfaces d'utilisation visuelle ou graphique pour l'industrie de l'informatique. Sun détient une licence non exclusive de Xerox sur l'interface d'utilisation graphique Xerox, cette licence couvrant également les licenciés de Sun qui mettent en place l'interface d'utilisation graphique OPEN LOOK et qui en outre se conforment aux licences écrites de Sun.

LA DOCUMENTATION EST FOURNIE "EN L'ÉTAT" ET TOUTES AUTRES CONDITIONS, DECLARATIONS ET GARANTIES EXPRESSES OU TACITES SONT FORMELLEMENT EXCLUES, DANS LA MESURE AUTORISEE PAR LA LOI APPLICABLE, Y COMPRIS NOTAMMENT TOUTE GARANTIE IMPLICITE RELATIVE A LA QUALITE MARCHANDE, A L'APTITUDE A UNE UTILISATION PARTICULIERE OU A L'ABSENCE DE CONTREFAÇON.



Please  
Recycle



Adobe PostScript

# IT Infrastructure Architecture Building Blocks

---

This article introduces building blocks for architecting IT infrastructures to provide web services. The primary audience for this article is a beginning/junior architect who has at least one year of experience with infrastructure platforms. This article recommends properties and criteria for evaluating and selecting the building blocks that best fit your environments. Examples throughout this article and a use-case scenario at the end apply the information and recommendations to realistic environments.

The article covers the following topics:

- “Defining Architecture Frameworks, Building Blocks, Architecture Patterns, and Design Patterns” on page 2
- “Providing or Outsourcing Services” on page 3
- “Using Building Blocks” on page 4
- “Assessing Building Block Properties” on page 5
- “Using Hard Architectural Building Blocks” on page 6
- “Using Soft Architectural Building Blocks” on page 12
- “Considering a Use Case Scenario” on page 19
- “Building Blocks and Sun's N1 Architecture” on page 22
- “About the Authors” on page 23
- “Related Resources” on page 24
- “Ordering Sun Documents” on page 24
- “Accessing Sun Documentation Online” on page 25

---

# Defining Architecture Frameworks, Building Blocks, Architecture Patterns, and Design Patterns

To explain where building blocks fit among other architecture concepts, in this section we define the terms IT architecture framework, building blocks, architecture patterns, and design patterns in the context of IT infrastructure architecture.

IT architecture framework refers to a concept and organizing principle that addresses and aligns technologies prevalent in application development, application middleware, management tools, networking, computing, and storage. The framework includes common architectures in each one of these areas and shows the synergies between these architectures.

Building blocks extend the concept of a framework to architect an IT environment. A building block approach helps categorize the components of building an IT architecture into hard, soft, and connector building blocks. Hard building blocks are a combination of software and hardware, which can further be divided into systemic and application tier building blocks. Soft building blocks are software entities like Enterprise Java Beans (EJBs). Connector building blocks are the glue that connects all the components. Building blocks and architectures using building blocks might use one or more architecture patterns.

Architecture patterns are well known ways to put together building blocks in an IT environment. An architecture pattern can address an entire layer of an IT architecture for a given service. As an example, a storage area network (SAN) architectural pattern can address the architecture for the storage infrastructure layer, and a message bus architecture is a pattern for architecting the application infrastructure layer.

Design patterns address problems with a layer and do not have to be an architecture for the entire layer. For example, the VLAN/VNET pattern is a design pattern within the network infrastructure layer that plays a key role in the design of a network that offers virtualization capabilities. Another example is the N+1 HA pattern, which is a design pattern within the compute infrastructure layer that plays a key role in the design of a HA cluster.

For architects working in large IT environments, their focus is typically on standardization, consolidation, efficiency, discipline, cost reduction, and total cost of ownership. And, of course, key to their success is meeting business requirements, increasing revenues, and satisfying both internal and external customers. By using building blocks, architecture patterns, and design patterns in architecture frameworks, less experienced architects can achieve their objectives.

Even when using existing products to build a system, architects are often faced with too many choices and possible combinations. This article offers a way for you to apply order to the often chaotic process of evaluating and selecting technologies to architect systems for web services.

---

## Providing or Outsourcing Services

One of the early considerations for architecting a system is to decide whether to provide services internally or to outsource them. You may have some of the following as your requirements:

- Plan ahead to accommodate new web services and other related technologies
- Improve service levels such as availability, scalability, security, efficiency, flexibility, and performance
- Reduce total cost of ownership
- Improve ease of deploying new services

Whether your enterprise provides or outsources services, or chooses a combination approach, we suggest that you require services to be based on open and industry standards.

---

# Using Building Blocks

Extending the concept of building blocks to architecting an IT environment for web services, we propose that most IT environments can be standardized using some or all of the architectural building blocks addressed in this article.

For example, you could treat a database server as a resource tier building block. (The resource tier is usually comprised of a data store or connectivity to a legacy system.) Resource tiers are application tiers; see TABLE 1 on page 6 for a list of application tiers. In large enterprises, especially service providers, there are many business units and each business unit typically owns a database server. Many of these database servers are designed and configured differently, managed by different groups of people, and maintained and tuned separately. This approach results in a significant management challenge as well as higher costs associated with deploying and maintaining these servers.

Using a dramatically different approach, you could think of these same database servers as building blocks, meaning that multiple database servers are created out of the same hardware, OS, patches, RDBMS, etc. The only differences necessary would be a few configuration, customization, and optimization variances, based on unique business requirements of the business units.

To determine what variances are needed, you could obtain the following information from each business unit:

- How many current users and what is the growth rate?
- Is usage pattern OLTP, DSS, or both?
- Security requirements?
- Current size requirements and growth rate?
- Names of the application database administrators who work with the business units to create and change data.
- Tables within the database based on business requirements.
- Any special optimization requirements.

Now, you could centralize all the database servers so they can be maintained by one group of administrators.

In this article, we describe each type of building block and their subcategories. Also, we provide guidance on what properties to look for when evaluating and selecting building blocks.

In this article, we divide the architectural building blocks into two primary types:

- Hard architectural building blocks
- Soft architectural building blocks

---

# Assessing Building Block Properties

What are the essential properties to look for when evaluating and selecting architectural building blocks? Although your environment may require additional properties, we recommend the following as a start:

- Scalable horizontally—It is possible to replicate the building block multiple times to scale the level of service it provides. For example, if a directory server can support up to 10,000 users, additional LDAP slaves can be deployed when the number of users have reached about 9000.
- Standards based—The building block is able to support many different kinds of applications. For example, an Oracle RDBMS server can be built as an architectural building block and any application that uses a database can use it.
- Customization—The building block can be used to deploy new applications with minimal customizing. For example, a security server such as the Checkpoint Firewall-I can be used to deploy new applications.
- Reusable—Soft service building blocks can be reused to build other applications. For example, a group of Enterprise Java Beans (EJBs) can be used to enable order entry processing.
- Portable—Soft building blocks are easily portable to other platforms.
- Integrable—The building blocks have standardized interfaces that make them easy to integrate with other architectural components.
- Asynchronous—Ideally, the result of using a building block allows for asynchronous communication. Building blocks facilitate communication by setting standards for interfaces and establishing common communications flows. Some examples are JMS and MDBs, which are described later in this article.

---

# Using Hard Architectural Building Blocks

Hard building blocks are a combination of software and hardware components. We view them essentially as servers that consist of either a combination of all the infrastructure layers and one application tier, or a subset of all the infrastructure layers and one application tier.

The hard building blocks are subdivided into the following building block categories:

- Systemic components
- Application tiers

TABLE 1 lists examples of hard building blocks for both systemic components and application tiers.

TABLE 1 Sample Hard Building Blocks

Systemic Components	Application Tiers				
	Resource	Integration	Business	Presentation	Client
Security servers	Database servers	EAI servers	Application servers	Portal servers	Cell Phones
Load balancing servers	Legacy systems	Directory servers	Calendar servers	Web servers	Pagers
Certificate servers	Directory servers	Wireless servers	Mail servers	Caching servers	PDAs
Monitoring servers	FTP servers	EII servers	Vending machines	WAP servers	Web browsers



# Systemic Building Blocks

Systemic building blocks create systemic qualities such as scalability, manageability, availability, reliability, and security. Systemic qualities are pervasive across all the application tiers and all infrastructure layers described in 3D Methodology (3DM) architectures. Examples of systemic building blocks are load balancers, certificate servers, firewall servers, and monitoring servers.

Using load balancing as an example, let's evaluate it as a systemic building block. Load balancing optimizes performance and increases capacity through horizontal scaling. Load balancing needs to be achieved across:

- Many of the application tiers, such as the web server in the presentation tier and the database server in the resource tier.
- Infrastructure layers, such as at the network and the upper platform layers.

We can achieve load balancing by using commercial off the shelf (COTS) software or hardware with pre-built software or firmware. We look for the following properties in a load balancing solution:

- Redirects a connection if a target server fails.
- Shares load across several target servers.
- Provides a highly scalable solution. For example, manages thousands of connections per hour.
- Provides a highly available solution. For example, administrators can add or remove building blocks.
- Runs without any interruptions to the services using load balancing.
- Allows different load-sharing algorithms to be selected by users.
- Provides redundancy, in the event of a failure.
- Provides a secure and easily manageable user interface. For example, provides a secure socket layer (SSL) web browser-based management interface.
- Aids in providing session persistence between a client and a server.
- Enhances a systemic quality in multiple tiers of an application service, such as the presentation, business, and resource tiers.
- Provides a highly secure solution, such as prevention of Denial of Service (DoS) attacks.

A firewall server is an important building block to introduce security within a service. In addition to being used for creating DMZs and filtering traffic specific to a zone, it supports application-specific protocols within incoming requests. As an example, if a new application is introduced within a network with the expectation of incoming requests using SIP as the protocol, a firewall building block with support for SIP can be deployed.

# Application Tier Building Blocks

Application tier building blocks are specific to a single tier and enhance the systemic qualities in a single tier. Examples of application tier building blocks are database servers, directory servers, mail servers, web servers, and web browsers.

The application tier building blocks are subdivided into the following categories:

- Resource
- Integration
- Business
- Presentation
- Client services

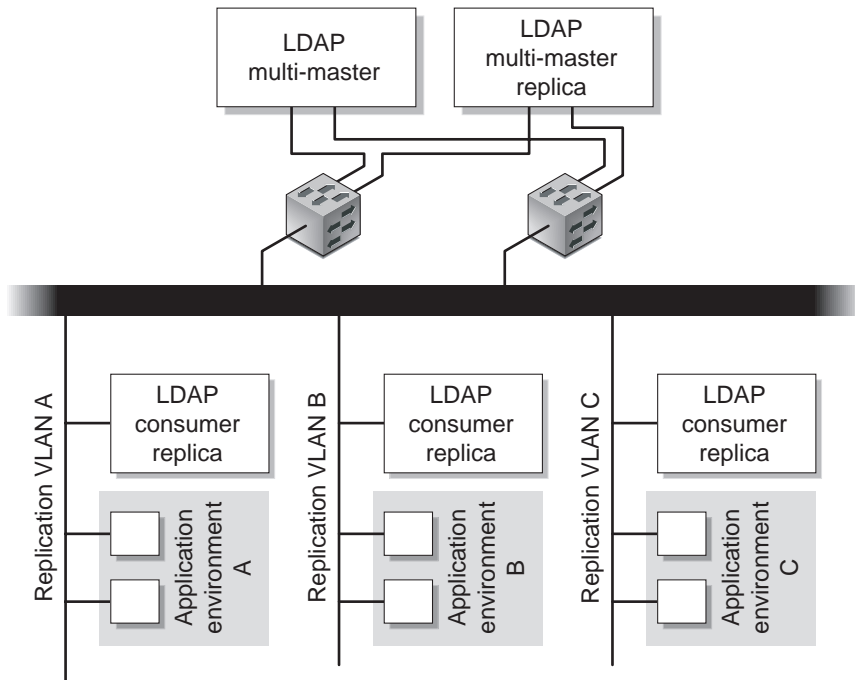
## Example 1

Using a directory server as an example, let's evaluate it as an integration tier building block. A directory server stores user profiles. If an enterprise has several services, and if the services use applications that comply with the Lightweight Directory Access Protocol (LDAP), then each compliant application does not need to store a separate set of user profiles. All user profiles can be stored in a group of directory servers that all services and applications access for authentication and authorization. We look for the following properties in a directory server solution:

- Complies with the open standard LDAP.
- Facilitates a loosely coupled architecture. For example, allows chaining, replication, and distribution.
- Helps lower the cost of managing and synchronizing multiple application-specific directories.
- Allows for adding, deleting, and updating user profiles for most applications used within the enterprise.
- Provides application programming interfaces (APIs) to develop cross-platform applications.
- Provides security and enables corporate-wide security.
- Allows very high scalability, especially for extranets.
- Allows logical partitioning. For example, it allows storing data in multiple databases. This property allows for partitioning (or splitting) the master LDAP logical directory information tree (DIT) in a large LDAP environment into multiple smaller DITs.
- Enables developing solutions that are interoperable in heterogeneous environments.

- Integrates with other building blocks such as application servers, web servers, calendar servers, and certificate servers.
- Is easy to monitor, administer, and manage.
- Enables development of asynchronous applications. For example, if an application updates the LDAP master when a user changes her password, then it should not verify the password immediately by doing a read operation. If the application does this, then the LDAP master does not have enough time to update the local replicas and authentication could fail. The application should send the update to the LDAP master, trust the LDAP master to do its job, and forget about it. This expectation is typical in an asynchronous operation.
- Integrates with applications that are LDAP compliant. The applications should use the LDAP protocol, which is the open standard for directory lookups.

FIGURE 1 shows how a directory server building block, with the properties listed previously, would be used to create a corporate-wide directory strategy. This approach uses the directory server as a building block to provide an integrated authentication and authorization mechanism.



**FIGURE 1** Example Showing the Use of Directory Server Building Block (The LDAP Master and Replicas are the Building Blocks.)

You could use this building block to achieve the same strategy on corporate extranets. Very high levels of scalability are required for extranets, and some of the properties described previously support the required scalability. As shown in FIGURE 1, user profiles for all application environments are stored in the LDAP multi-master server and are pushed out to the local replicas in each environment. The local replicas help keep the performance of the applications at an optimum level. Updates from each application environment are sent directly to an LDAP multi-master. The multi-master allows for seamless failover for the LDAP write-master server. For performance and scalability reasons, we believe that it is better to use additional building blocks (such as load balancers) for protection, as shown in the previous figure.

You could also configure a directory server to serve multiple roles, for example, as a building block for both the integration and resource tiers.

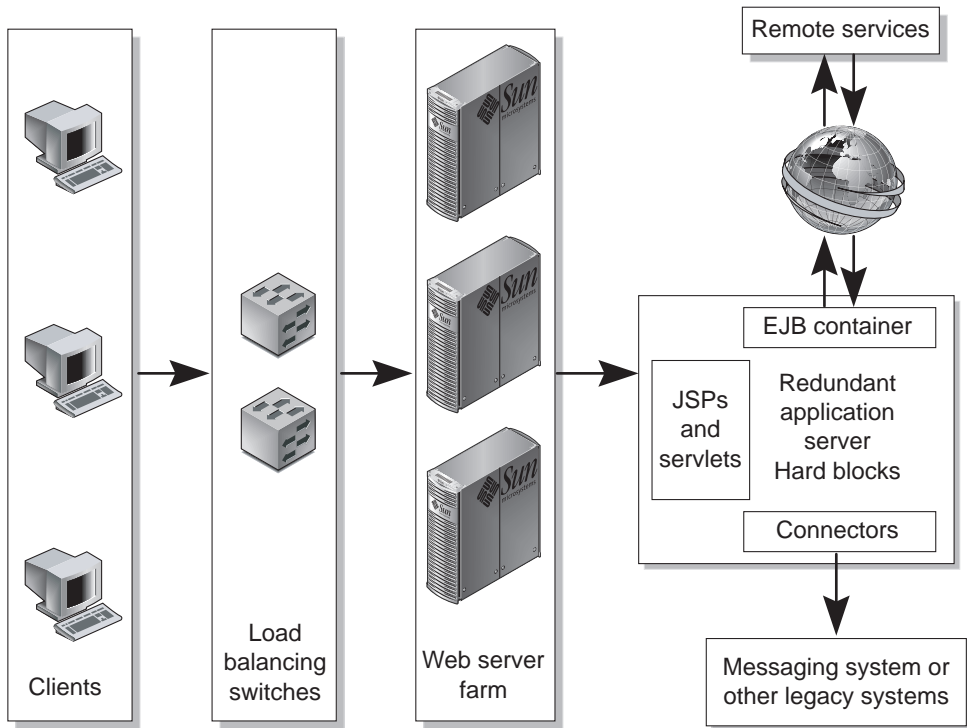
## Example 2

Let's consider another application tier building block example. The application server can be used out-of-the-box or customized for specific applications. The application server typically has an EJB container with several EJBs that execute the business logic for various services. Many custom EJBs are available commercially, and you can use them with the application server building block.

This building block is a key part of architecting most IT environments. For many engineers, when they write an application to take care of a set of complex business problems, it is typically hard for them to focus on other functionality in the application, like making it highly available, highly scalable, highly secure, etc. They would rather focus on developing the core business logic within the application. The application server as a building block becomes very useful to these engineers, because it frees them to repeatedly write code for "middleware" functions such as:

- Resource pooling
- Transaction processing
- Threading
- Caching
- Security
- Load balancing
- Remote method invocation (RMI)
- Availability (transparent fail-over)
- System management
- Caching
- Logging

FIGURE 2 shows an example of using an application server as a building block.



**FIGURE 2** Example Application Server as a Building Block

---

# Using Soft Architectural Building Blocks

These building blocks are software components only. Evaluating, selecting, and creating soft building blocks is an important part of architecting an IT environment. Soft building blocks are subdivided into the following building blocks to support the important architectural concept of “separation of concerns:”

- Services
- Connectors
- Presentations

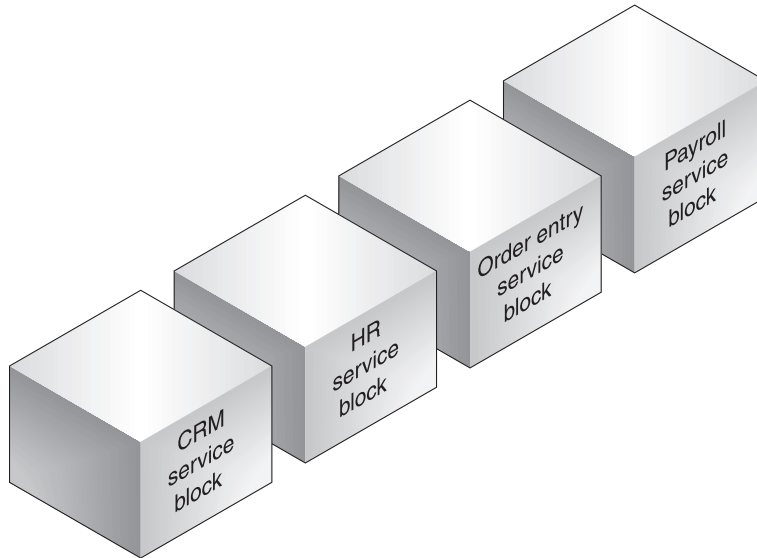
**TABLE 2** Sample Soft Building Blocks

<b>Services</b>	<b>Connectors</b>	<b>Presentations</b>
Internet bill presentment and payment (IBPP)	Java Naming and Directory Interface (JNDI)	HTML pages
Payment processing	Java database connectors (JDBC)	Java server pages (JSP)
Order entry	Java messaging service (JMS)	Java servlets
Provisioning	Java Native Interface (JNI)	
Remote service	Remote method invocation (RMI-IIOP)	
Employee services (for example, human resources and payroll)	Java API for XML parsing (JAXP)	
Customer resource management (CRM)	Java connector architecture (JCA)	
Enterprise resource planning (ERP)		

# Service Building Blocks

Service building blocks are most commonly EJBs corresponding to each business requirement. The EJB building blocks enable rapid and simplified development of distributed, transactional, secure, and portable applications. The EJBs are fully deployable units.

Some service building blocks take care of a single business requirement and others satisfy multiple business requirements. Note that the EJB building block is not a complete solution; you have to use several EJB building blocks with hard building blocks, such as an application server, to create a complete solution. A typical use of service building blocks is for web services. FIGURE 3 shows an example of EJBs as part of a complete solution.



**FIGURE 3** EJB Service Blocks As Part of a Solution

TABLE 3 lists functions of sample service building blocks.

**TABLE 3** List of Functions of Sample Service Building Blocks

<b>Services</b>	<b>Function</b>
Internet bill presentment and payment (IBPP)	Allows customers to view, store, and pay bills via the Internet. Presenting bills over the IBPP integrates bill presentment and payment into a single service.
Payment processing	Handles electronic payment transactions.
Order entry	Handles all the accounting and record keeping functions necessary to effectively process online orders entered by customers.
Provisioning	Ensures that the inventory levels are maintained, and keeps the inventory records up-to-date.
Remote service	Provides any of the other services offered externally and referred to by the UDDI registry.
Employee services	Provides human resource services like employee relationship, 401K, and payroll management.
CRM	Provides customer call centers and allows self service over the Internet for customers. CRM is a meta category encompassing many of the other services.
ERP	Provides HR, payroll, and accounts payable services used by employee services personnel.



# Connector Building Blocks

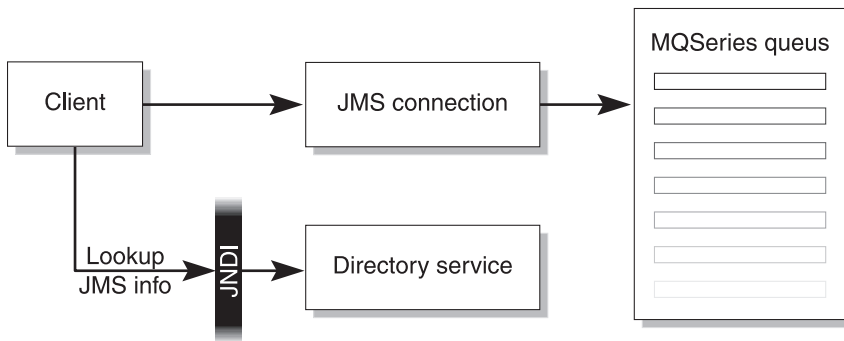
Connector building blocks are soft blocks that are typically used to connect the resource tier to other services. For example, a Java Database Connector (JDBC) is a connector building block. It is an open standard API used to connect a service or EJB to any database. These building blocks connect services to legacy systems, databases, and other services.

TABLE 4 lists the functions of connector building blocks.

**TABLE 4** Functions of Connector Building Blocks

Connector	Function
Java Naming and Directory Interface (JNDI)	Services (EJB components) connect with directory servers in the integration tier using JNDI.
Java Database Connectors (JDBC)	Services (EJBs) in the business tier request data in a database via the JDBC connector and return the response via JDBC. There are specific JDBC for different RDBMS vendors.
Java Messaging Service (JMS)	Services (EJBs) use JMS to communicate with existing message oriented middleware (MOMs), such as MQSeries.
Java Native Interface (JNI)	Provides interface for writing java native methods and embedding Java Virtual Machine (JVM) into native applications. JNI is required for situations where existing libraries or code written in another language has to be accessed from a Java program.
Remote Method Invocation (RMI-IIOP)	Presentation Tier soft blocks like JSP use RMI/IIOP to communicate with services (EJB components).
Java API for XML Parsing (JAXP)	An API used by EJBs in the business tier and servlets in the presentation tier when performing interactions with other services on the internet (B2B interactions).
Java Connector Architecture (JCA)	JCA is an important connector block used to integrate with existing applications. Connectors are already available for SAP/R3, CICS, VSAM datastore, PeopleSoft, etc.

The JMS API connects to message oriented middleware (MOM) services such as JMQ and MQSeries. A client can use JMS to connect directly to a MOM service as shown in FIGURE 4.



**FIGURE 4** Using JMS API as a Connector Building Block

The advantages of using JMS are as follows:

- Allows asynchronous messaging, which results in loosely coupled systems.
- Clients are not tied to a single server.
- Systems developed using JMS are more resilient to failures.
- Systems developed using JMS are more extensible, allowing development of additional features.
- Provides an effective means of transmitting events between applications.

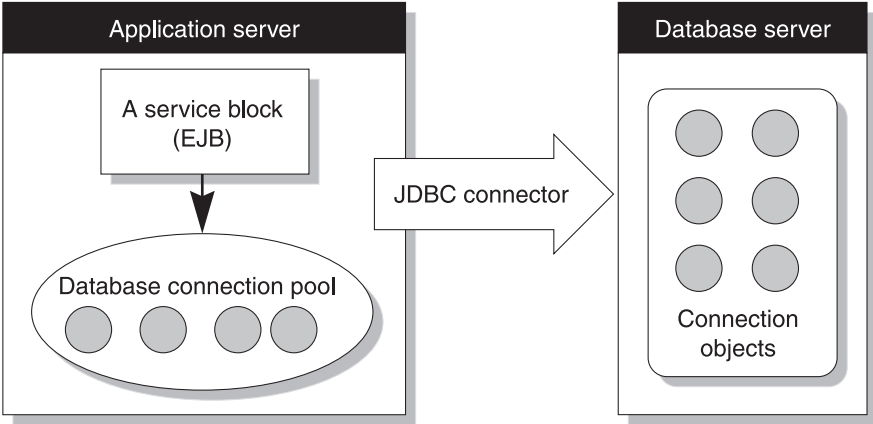
The JMS connector block can connect EJBs directly to MOM services. This approach provides great advantages of being able to integrate EJBs with JMS. The architectural advantages of integrating EJBs with JMS are as follows:

- EJBs become part of a loosely coupled system.
- Clients become non-blocking clients. For example, they do not have to wait for the result of a request before going on to other activities.
- Enables *n*-ary communications. For example, a client can request something that it wants from *n* number of servers, rather than depend on one server.

Also, JMS allows clients to communicate with Message Driven Beans (MDBs) a special type of EJB building block. MDBs are decoupled from clients. The clients can communicate with MDBs through the JMS API. The MDBs were created mainly to enable messaging. An MDB is invoked by an EJB container within an application server when the server receives a JMS message. Messages are received in message queues or “topics,” and MDBs are message queue listeners or receivers who read from message queues.

The JDBC building block connects business tier building blocks to the resource tier. Connection pooling is an important architectural property of this building block. Establishing and cleaning up connections to a database can be very time consuming for an application, slowing its performance. Instead of creating new connections as

needed, it is better for a system to use an existing pool of connections that are held open and available at all times. Connection pooling can be used between the presentation and business tiers too. COTS connection pooling building blocks are available from Silver Stream and WebLogic. FIGURE 5 shows connection pooling between the business and resource tiers.



**FIGURE 5** Sample Connection Pooling Between Tiers

# Presentation Building Blocks

Presentation building blocks are soft blocks in the presentation tier. These building blocks run within hard blocks such as web servers, portal servers, etc., in the presentation tier.

TABLE 5 lists functions of presentation building blocks.

**TABLE 5** Functions of Presentation Building Blocks

Presentation	Function
HTML pages	Displays static web pages.
Java server pages (JSP)	Pulls information from session objects and creates HTML web pages for displaying results.
Java servlets	Routes device independent requests from client tier (devices, browsers etc) to business tier (for example, EJBs in the application server).

Java servlets and Java server pages (JSP) provide a simplified and fast way to create dynamic web content. JSP allows fast development of server and platform-independent web based applications. JSP and servlets are widely used for developing web-based interactive applications. JSP and servlets run within web servers and application server hard building blocks. When a user makes a request from an enterprise web site, the servlets create the information that the user requests, sometimes calling other entities such as the EJBs. The JSP and servlets are generally stored in containers within the web server.

Some architectural advantages of Java servlets and JSPs are as follows:

- Modularity
- Platform independence
- Enhanced performance
- Separation of logic from the design and actually displaying information
- Extensibility
- Administration and ease of use
- Scalability
- Available as “ready to use” building blocks

---






**Note** – “Ready to use” building blocks in this case refers to items such as Apache Web Server, SunONE Web Server containers, BEA WebLogic application server, IBM WebSphere application server, and SunONE application server containers.

---





# Considering a Use Case Scenario

Using a realistic example of purchasing a book from an online book store, let's apply a set of business functional requirements to architect a solution for a use case scenario. For this use case scenario, we use both hard and soft building blocks, described earlier in this article.

**TABLE 6** Use Case Scenario: Purchasing a Book Online

<b>Description</b> 	Anyone with Internet access can browse a catalog at this book store and purchase a book.
<b>Actor(s)</b> 	Purchaser
<b>Assumptions</b> 	The purchaser is a registered user of this online book store. She is using a new credit card, and her credit information has to be verified. The purchaser has just been authenticated and authorized to use this feature.
<b>Steps</b> 	<ul style="list-style-type: none"> <li>• Purchaser searches for a book by author's name and book title.</li> <li>• System searches the inventory and displays the book details, including price.</li> <li>• Purchaser clicks "Buy" option and a form is displayed requesting credit card information.</li> <li>• Purchaser enter credit card information and clicks "Send."</li> <li>• System displays a message confirming the purchase order confirmation number.</li> <li>• System sends an email to the purchaser confirming the order and order number.</li> <li>• Purchaser expects to receive the book in "x" number of days, per the order confirmation email.</li> </ul>
<b>Variations</b> 	Several variations are possible such as the book is out of stock, unavailable at this store, out of print, or not matching the author's name entered. The purchaser is suggested to search based on title only, etc.

**TABLE 6** Use Case Scenario: Purchasing a Book Online (*Continued*)

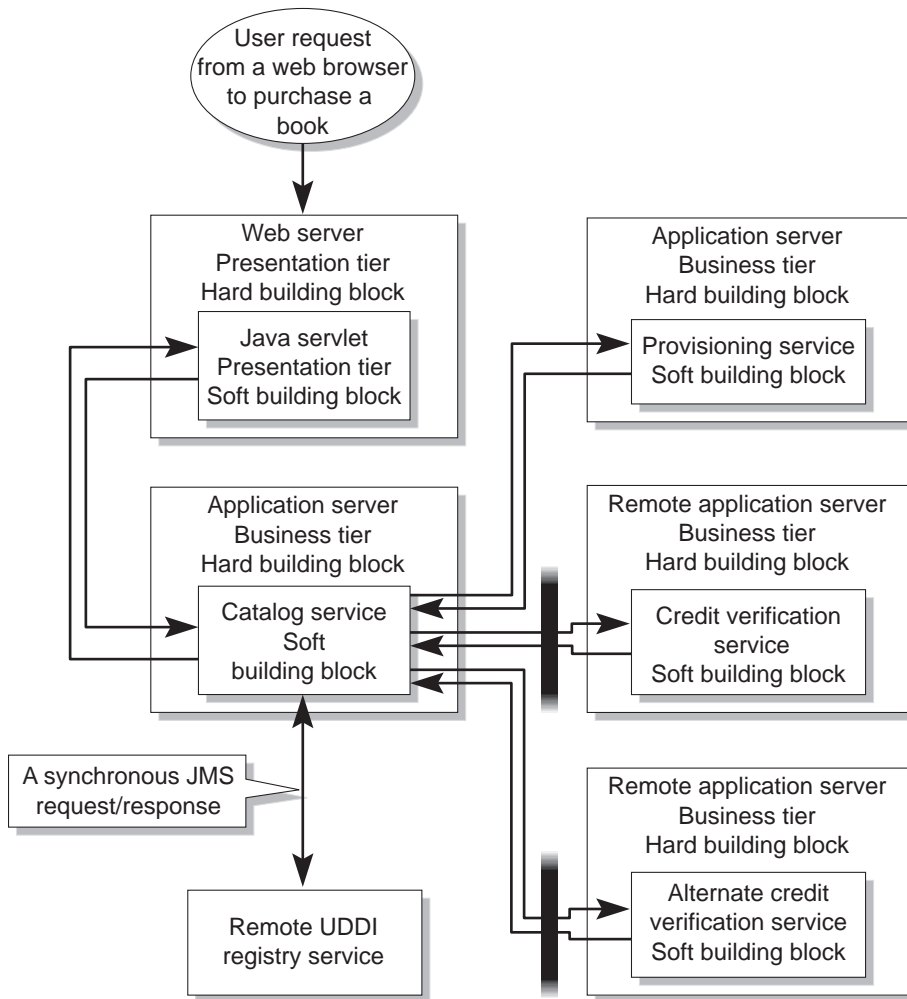
<b>Description</b> 	Anyone with Internet access can browse a catalog at this book store and purchase a book.
<b>Nonfunctional or Issues</b> 	Must update inventory data after this transaction.
<b>Related Use Cases</b> 	Authenticate and authorize a purchaser.
<b>Priority</b> 	Critical

Assuming that the purchaser is authenticated and authorized using other building blocks (such as directory servers and identity servers), a Java servlet within the web server building block sends a search request to the catalog service EJB in an application server building block. The catalog service EJB in turn requests a provisioning service block (could be an EJB) to check for the requested book in the inventory.

If the book is available, the catalog service sends a JMS request to a known credit verification service. To receive asynchronous JMS messages, the credit verification service must be made up of MDBs.

If the credit verification service is unavailable, the JMS request queries a well-known Universal Description Discovery and Integration (UDDI) registry service for more information about a reliable local credit verification service. The UDDI registry sends a JMS message back to the catalog service EJB, with detailed information about the credit verification service EJB running on a remote application server building block.

The JMS interactions are asynchronous. FIGURE 6 below shows the interactions between the local and remote service building blocks. It demonstrates how these building blocks work together; it does not show details about redundancy, scalability, security, etc. A real-world architecture would use load balancers, certificate servers, directory servers, and other features like platform clustering and session state management to achieve higher levels of systemic qualities.



**FIGURE 6** Interactions Between Local and Remote Service Blocks

---

# Building Blocks and Sun's N1 Architecture

N1 is Sun's new approach to the datacenter that redefines the meaning of a system. With N1, Sun becomes a provider of networked systems built from Internet-attached components. N1 virtualizes compute, network, and storage resources. It automates systems operations and manages services instead of servers. N1 uses the concepts associated with building blocks, system service containers, and application service containers; it aligns these containers, taking into account an application's affinity to certain hard building blocks.

The following are the advantages of understanding the building block approach if you are planning to adopt N1 architecture and products.

- N1 uses architecture building blocks, patterns, and frameworks with software to virtualize all compute, network, and storage resources.
- Understanding the building block approach helps you while implementing N1 in your IT environment. N1 software displays the building blocks available to you. You then use those building blocks to create a service based on well-known architectural patterns.
- You can put these building blocks together using the N1 Provisioning Server (NPS) software, which is GUI based. N1 puts the building blocks together for you behind-the-scenes, using techniques such as soft cabling (physically wire-once and rewire logically, as needed).



---

## About the Authors

Ramesh Radhakrishnan has been an IT Architect and Consultant at Sun Microsystems for the past three years. He conducts availability and architecture assessments, and he designs IT environments for several of Sun's mission-critical customers. Before joining Sun, he worked as a system administrator, IT consultant, and ClearCase Consultant. He has a master's degree in Computer Science from Old Dominion University. Over the years, he has gained experience in the areas of backup and recovery architecture, disaster recovery architecture, and IT processes, along with many other IT infrastructure management areas. Ramesh is currently part of a team developing an architecture basics course for Sun engineers.

Rakesh Radhakrishnan is a Sr. Technical IT Architect with Sun Professional Services. He works in the Communication Market Area -Professional Services Organization and has led multiple IT Architecture, Architecture Assessment, and Architecture Workshop projects for telecommunication and service provider customers like Telcel (Mexico), Telefonica (Argentina), Verizon, Cingular, Southwestern Bell Communications (SBC), and American Online Time Warner (AOL TW). He is an active member of the Global PS Architecture Council and Service Technology Council. Also, he is the Chair for S1onN1 working group, defining a patent pending technology (container alignment engine) based on current and future advances in application containers and system containers. Rakesh is a frequent speaker at both Sun and Industry Conferences (SuperG, SunNetwork, JavaONE, OMG, CMG, etc.) and has presented/published more than 12 papers at these conferences.

---

## Related Resources

### Publications

- *Dot-Com and Beyond: Breakthrough Internet Based Architectures and Methodologies*, by Sun Microsystems Press/Prentice Hall PTH, ISBN 0-13-062297-4, June 2001.
- Maier, Mark W. and Rehtin, Eberhardt, *The Art of Systems Architecting*, CRC Press, 2nd edition, ISBN 0849304407, June 2000.
- “Take a 3D Approach to Architecture Design,” by Sun Professional Services.Com Consulting, Sun Microsystems, Inc., Sun Journal, Volume 5, No. 1:  
<http://www.sun.com/executives/sunjournal/v5n1/feature2.html>
- Cockroft, Adrian and Walker, Bill, *Capacity Planning for Internet Services*, Sun BluePrints, Sun Microsystems, Prentice Hall, 1st edition, ISBN 0130894028, January 2001.
- “Overview of Solaris Patch System Testing and Performance Regression Testing Overview,” SunSolve, Sun Microsystems, Inc: <http://sunsolve.sun.com>. Click on the Patch Portal link.

### Web Sites

- Application tier building blocks: [http://www.sun.com/software/product\\_categories/application\\_integration\\_messaging.html](http://www.sun.com/software/product_categories/application_integration_messaging.html)
- Enterprise Java Beans: <http://java.sun.com/products/ejb/docs10.html>
- Soft building blocks: <http://java.sun.com/blueprints>
- SunONE architecture: <http://sunonedev.sun.com/platform/architecture/>

---

## Ordering Sun Documents

The SunDocs<sup>SM</sup> program provides more than 250 manuals from Sun Microsystems, Inc. If you live in the United States, Canada, Europe, or Japan, you can purchase documentation sets or individual manuals through this program.

---

# Accessing Sun Documentation Online

The `docs.sun.com` web site enables you to access Sun technical documentation online. You can browse the `docs.sun.com` archive or search for a specific book title or subject. The URL is `http://docs.sun.com/`

To reference Sun BluePrints OnLine articles, visit the Sun BluePrints OnLine Web site at: `http://www.sun.com/blueprints/online.html`