Overview Guide

iPlanet Application Server

Version 6.0

806-3491-01
May 2000

Printed in the United States of America 00 99 98 5 4 3 2 1

Contents

Preface

This preface contains the following topics:

- Using the Documentation
- About This Guide
- How This Guide is Organized
- Documentation Conventions

Using the Documentation

The *Overview Guide* is only one of several documents that help you develop, deploy, and manage web-based enterprise applications. The following table lists the tasks and concepts that are described in the iPlanet Application Server (iAS), iPlanet Application Builder (iAB) printed manuals and online release notes. If you are trying to accomplish a specific task or learn more about a specific concept, refer to the appropriate manual.

Note that the printed manuals are also available as online files in PDF and HTML format at *http://docs.iplanet.com*

Table 1

For information about	See the following	Shipped with
Late-breaking information about the software and the documentation	readme.htm	iAS6.0 iAB 6.0
Installing iPlanet Application Server and its various components (Web Connector plug-in, iPlanet Application Server Administrator), and configuring the sample applications	Installation Guide	iAS6.0
Installing iPlanet Application Builder	install.htm	iAB 6.0

Table 1

For information about	See the following	Shipped with
Basic features of iAS, such as its software components, general capabilities, and system architecture	Overview	iAS6.0, iAB 6.0
Administering one or more application servers using the iPlanet Application Server Administrator tool to perform the following tasks: • Deploying applications with the Deployment Manager tool • Monitoring and logging server activity • Setting up users and groups • Administering database connectivity • Administering transactions • Load balancing servers • Managing distributed data synchronization	Administration & Deployment Guide	iAS6.0
Migrating your applications to the new iPlanet Application Server 6.0 programming model from version 2.1 and version 4.0, including a sample migration of an Online Bank application provided with iPlanet Application Server	Migration Guide	iAS6.0, iAB 6.0
Creating iAS6.0 applications within an integrated development environment by performing the following tasks: • Creating and managing projects • Using wizards • Creating data-access logic • Creating presentation logic and layout • Creating business logic • Compiling, testing, and debugging applications • Deploying and downloading applications • Working with source control • Using third-party tools	User’s Guide	iAB 6.0

Table 1

For information about	See the following	Shipped with
Creating iAS6.0 applications that follow the new open Java standards model (Servlets, EJBs, JSPs, and JDBC), by performing the following tasks: • Creating the presentation and execution layers of an application • Placing discrete pieces of business logic and entities into Enterprise Java Beans (EJB) components • Using JDBC to communicate with databases • Using iterative testing, debugging, and application fine-tuning procedures to generate applications that execute correctly and quickly	Programmer’s Guide (Java)	iAB 6.0
Using the public classes and interfaces, and their methods in the Netscape Application Server class library to write Java applications	Server Foundation Class Reference (Java)	iAS6.0, iAB 6.0
Creating C++ applications using the iAS class library by performing the following tasks: • Designing applications • Writing AppLogics • Creating HTML templates • Creating queries • Running and debugging applications	Programmer's Guide (C++)	Order separately
Using the public classes and interfaces, and their methods in the iAS class library to write C++ applications	Server Foundation Class Reference (C++)	Order separately

About This Guide

The *Overview Guide* describes the product components, features, and system architecture of iPlanet Application Server (iAS). The guide contains useful background information on the application servers, Java 2 Enterprise Edition and products that integrate with iAS.

How This Guide is Organized

This guide is divided into 5 chapters. Chapter 1 gives an executive overview of iPlanet Application Server. Chapter 2 and 3 focuses on architecture, Chapter 4 gives an overview of J2EE and Chapter 5 expands on the iPlanet product family.

Documentation Conventions

Files and directory paths are given in the Windows format with backslashes separating directory names. For Unix versions, the directory paths are the same, except slashes are used instead of backslashes to separate directories.

This guide uses URLs of the form: *http://server.domain:port/path/file.html*

Where *server* is the name of the server on which you run your application; *domain* is your internet domain name; *path* is the directory structure on the server; and *file* is the individual filename. Italic items in URLs are placeholders.

This guide uses the following font conventions:

- The `monospace` font is used for sample code and code listings, API and language elements (such as function names and class names), file names, path names, directory names and HTML tags.
- *Italic* type is used for book titles, emphasis, variables and placeholders, and words used in the literal sense.

Chapter 1

iPlanet Application Server

This chapter summarizes the main features and components of iPlanet Application Server (iAS).

Executive Overview

iPlanet Application Server 6.0 provides the most robust e-commerce platform for delivering innovative and leading edge application services to a broad range of servers, clients and devices. Built on a heritage of world-class scalability, reliability and performance, the iPlanet Application Server delivers on the requirements for global e-commerce success by rapid time-to-market, the ability to leverage information systems and business processes across the extended enterprise. With thousands of customers, the iPlanet Application Server enables enterprises, service providers and dot.com businesses to exploit the enormous opportunities of the Net Economy.

iPlanet Application Server delivers on the requirements for e-commerce success, including the ability to:

- Develop and deliver applications based on Java 2 Platform Enterprise Edition (J2EE) standards
- Handle unplanned successes through a highly scalable, reliable, and available architecture
- Provide business-to-business, business-to-consumer, and business-to-employee solutions on a single platform
- Leverage and integrate information assets both internal and external to the enterprise
- Automate business processes across suppliers, customers, and partners

iPlanet Application Server provides robust support for the J2EE standards. At virtually every level of development and deployment, iPlanet Application Server features interfaces and functionality that conform to the J2EE specification. In addition, iPlanet Application Server provides additional key capabilities through services such as load-balancing, failover, and high-availability applied to these same J2EE standards. For example, iPlanet Application Server delivers load-balancing services across JavaServer Pages and servlets.

Based on an “open tools” strategy, application development for the iPlanet Application Server platform can be performed with a number of different toolkit options, including:

- iPlanet Application Builder offers wizard-based development and tight integration with the application server.
- Support for Forte for Java, Enterprise Edition featuring enterprise-class development for team development capabilities, including distributed debugging.
- Third party products, such as WebGain VisualCafe, IBM VisualAge, Macromedia Dreamweaver, and Inprise JBuilder plug into the iPlanet Application Server architecture for tight integration.
- The iPlanet Application Server solution includes the ability to automate business processes across J2EE business logic and legacy/enterprise business logic. The iPlanet Process Manager is a comprehensive web-based solution for designing, deploying and managing business processes on the iPlanet Application Server.

For comprehensive integration with multiple back-end information systems, iPlanet Application Server includes a Unified Integration Framework and Enterprise Connectors which extend the application and data assets for ERP environments, such as IBM CICS, BEA Tuxedo, SAP R/3, and PeopleSoft, into dynamic Web services.

New Features

iPlanet Application Server 6.0 provides significant enhancements over previous versions of the product. This includes full compliance with the Java™ 2 Platform, Enterprise Edition (J2EE™) APIs and specifications, improvements in performance, scalability and reliability, and enhanced manageability and administration. The iPlanet Application Server platform provides a new integration framework that simplifies and accelerates the development of applications that integrate across multiple information systems, including SAP R/3, CICS, Tuxedo, and PeopleSoft. The product family includes an enterprise class tool suite, as well as support for industry-leading developer and deployment tools. iPlanet Application Server is backward compatible with previous versions of iPlanet Application Server.

Certified Compliance with Java 2 Enterprise Edition

iPlanet Application Server 6.0 features strong support for a wide range of industry standards, including Java™ 2 Platform, Enterprise Edition (J2EE™) and is certified with the Compliance Test Suite (CTS). J2EE provides a complete, secure foundation and describes a rich set of standards for security, development, deployment, code re-use and portability that allows the enterprise to create applications that are portable and vendor independent. J2EE consists of the following components and APIs:

Enterprise JavaBeans™ (EJB) 1.1 The server-side component architecture for the J2EE platform. EJB enables rapid and simplified development of distributed, transactional, secure, and portable Java applications.

JavaServer Pages™ (JSP) 1.1 The JavaServer Pages technology provides a simplified, fast way to create dynamic web content. The JSP technology enables rapid development of web-based applications that are server- and platform-independent

Java Servlets 2.2 Provides web developers with a simple, consistent mechanism for extending the functionality of a web server and for accessing existing business systems.

JDBC™ 2.0 (Core and Standard Extensions, and Optional Package 2.0) Provides access virtually any tabular data source from the Java programming language. It provides cross-DBMS connectivity to a wide range of SQL databases, and now, with the new JDBC API, it also provides access to other tabular data sources, such as spreadsheets or flat files.

Java Transaction API (JTA) 1.0 Specifies standard Java interfaces between a transaction manager and the parties involved in a distributed transaction system: the resource manager, the application server, and the transactional applications.

Java Naming and Directory Interface™ (JNDI) 1.2 Works in concert with other J2EE technologies to organize and locate components in a distributed computing environment.

RMI-IIOP 1.0.1 Remote Method Invocation (RMI) over Internet Inter-Orb Protocol (IIOP) delivers Common Object Request Broker Architecture (CORBA) compliant distributed computing capabilities to the Java™ 2 platform and to the Java Development Kit (JDK™) 1.2. This enables Rich Clients such as stand alone Java applications to communicate directly with the application server.

Java Message Service (JMS) 1.0.2 Provides a set of standard Java language interfaces to enterprise messaging systems, often called message-oriented middleware. These interfaces are implemented by products called JMS providers. The JMS API and provider framework enables the development of portable, message-based applications in the Java programming language.

JavaMail™ 1.1 Provides a set of abstract classes that model a mail system. The API provides a platform independent and protocol independent framework to build Java technology-based mail and messaging applications.

JavaBeans™ Activation Framework (JAF) 1.0 Enables developers to take advantage of standard services to determine the type of an arbitrary piece of data, encapsulate access to it, discover the operations available on it, and to instantiate the appropriate bean to perform said operation(s). For example, if a browser obtained a JPEG image, this framework would enable the browser to identify that stream of data as an JPEG image, and from that type, the browser could locate and instantiate an object that could manipulate, or view that image.

Extensible Markup Language (XML) XML is used as the standard for describing all J2EE Deployment Descriptors. iPlanet Application Server use XMLfor application assembly and deployment.

Other standards include:

- CORBA 2.3 enables cross-platform interoperability
- HTML is the universal language of the Web
- LDAP (Lightweight Directory Access Protocol) enables access to large scale directories, providing authentication and access control
- SNMP provides an interface which enables status monitoring by network management systems
- IMAP/POP3 are standard electronic e-mail protocols
- XA is a standard protocol for distributed database transactions

J2EE Enhancements

More than just complying with the J2EE standard, iPlanet Application Server enhances the functionality of the J2EE specification:

- Overall availability is improved with failover for key J2EE components:
  - Stateful EJB failover
  - Rich client failover
- Rich client load balancing over RMI-IIOP
- JSP load-balancing — JSPs now can have GUIDs:
  - GUIDs uniquely identify each JSP process. This enables JSP to be load-balanced, similarly to servlets, and applogics in previous versions of the application server.
  - This also enables JSPs to use results caching
  - There is continued support for JSPs without GUIDs, as unregistered JSPs
- There is a conversion tool provided to transform JSP 0.92 pages (Netscape Application Server 4.0) to JSP 1.1.
- JSP Page Caching — calls to JSP pages are cached, providing similar functionality as results caching in NAS 4.0.
- JMS Connection Pooling and User Mapping — Connection Pooling enhances the performance and reliability of iPlanet Application Server applications using JMS via the creation and management of pools of connections from iPlanet Application Server to a JMS-enabled messaging product. User mapping speeds iPlanet Application Server application development and eases administration by supporting the easy mapping of users — authenticated at the web application level — to users, groups, and roles authorized by the JMS-enabled messaging provider.
- There is a performance boost for compositional JSPs.
- Support for JSP 1.1 tag library extensions.

Improved Performance, Scalability and Reliability

The iPlanet Application Server provides a fifth generation architecture with market proven performance, scalability, and reliability leadership at leading e-commerce sites. For customers requiring high transaction integrity and continuous up-time, the iPlanet Application Server eliminates single points of failure through application failover at every level of the J2EE development environment, including JavaServer Pages, Java Servlets, and Enterprise JavaBeans failover. iPlanet Application Server also ensures that user information and application data are not lost during a failure by distributing the state and session information of a transaction across multiple servers.

iPlanet Application Server features enhanced load balancing capabilities. For example, there is weighted round robin load balancing — administrators can assign characteristics such as numbers of CPUs and CPU speed to determine how servers can balance requests. iPlanet Application Server supports load balancing among CPUs within a multiprocessor server, and provides load-balancing capabilities across the J2EE processes.iPlanet Application Server also provides response time and server-load based load balancing. Load balancing is also available for JSP processes, and rich clients.

iPlanet Application Server provides high performance features including connection caching and pooling, results caching, data streaming, optimized web server communication, and a multi-threaded, multiprocessing architecture. The iPlanet Application Server also provides application partitioning to ensure that application logic is run on the server with the most capacity. iPlanet Application Server also features performance improvements with Web and Directory services. For example, this release features high-performance connectivity with the iPlanet Web Server and iPlanet Directory Server products.

iPlanet Application Server integrates the Encina transaction monitor as core feature of the server for optimal performance, reliability, and manageability. The Encina transaction monitor provides reliability for distributed transactions.

Many iPlanet Application Server components, including JSPs and applications, can be deployed without shutting down the relevant servers. This provides maximum availability while offering up-to-date application functionality.

Reacting to “unplanned success” is the most critical requirement for e-commerce platforms on the internet where demand can scale from hundreds to millions of users over-night. The iPlanet Application Server provides superior performance required to react to such high volume by allowing companies to scale applications across multiple CPUs within a physical hardware server, as well as across multiple machines.

Enterprise-Class Development and Deployment Tools

iPlanet Application Server supports the Java programming environment for new application development, as well as existing Netscape Application Server 4.0 C++ applications. However, Java applications are easier to develop and maintain, because they can take advantage of the enhanced, standards-based application model.

Application developers can choose from among various tools to build applications. These tools can range from simple text editors, to visual Java editors and visual HTML editors, to integrated development environments (IDEs). iPlanet Application Server's workbench plug-in framework enables a best-of-breed approach — developers can choose the development tool of preference for creating iPlanet Application Server applications. J2EE compliance provides simple plug-in capabilities for applications and components that are externally developed or purchased on the open market.

The tools include iPlanet Application Builder and iPlanet Extension Builder. These tools are tightly integrated with iPlanet Application Server but are packaged separately.

iPlanet Application Builder

iPlanet Application Builder is an Internet application development tool designed to simplify the creation of multi-tiered enterprise-class applications that run on iPlanet Application Server. iPlanet Application Builder provides an intuitive and productive web development environment that enables developers to leverage the rich, prebuilt application and infrastructure services of Application Server. By targeting the distributed multitier application model of iPlanet Application Server, iPlanet Application Builder enables developers to rapidly build sophisticated, business-critical web applications for the Internet. iPlanet Application Builder also interoperates with third-party tools such as WebGain VisualAge, Inprise JBuilder, and Macromedia Dreameaver.

Third Party Tools

Forte for Java Enterprise Edition is available for customers who are deploying large (10-200) development teams, and working from a large pool of reusable components. Forte for Java has strong versioning capabilities, making it ideal for environments where there is a need to manage and control frequent application changes.

iPlanet Application Server also supports sophisticated Object to Relational (OR) mapping middleware. CocoBase, from Thought, Inc., provides a dynamic repository based Object to Relational mapping tool which delivers Container Managed Persistence (CMP) and Bean Managed Persistence.

Enhanced Management and Administration

iPlanet Application Server eases application management by providing integrated management facilities.

iPlanet Application Server Administration Tool

iPlanet Application Server Administration Tool (iASAT) manages one or more iAS machines or applicationsis. iASAT is a Java application with a graphical user interface that manages such tasks as performance settings, load balancing parameters and configuring devices. iASAT enables the following capabilities:

- Remote management of multiple servers and distributed applications.
- Dynamic deployment and scaling of applications.
- Performance tuning and optimization of the server environment.
- Management and tuning involves tasks such as adjusting database connection threads, adjusting load-balancing parameters, configuring web servers, and managing roles
- Event Logging and Failure Analysis
- Security features including viewing and management of security roles
- Transaction Management features for local or global transactions
- Application management features for J2EE applications

Dynamic Application Management

iPlanet Application Server's architecture allows partitioned applications to run even if one or more servers fail. In a load-balanced server configuration, application logic can be replicated on multiple servers. If a server fails, the load balancing module dynamically directs requests to other available servers, thus preventing application-wide failure.

Because the iPlanet Application Server architecture promotes high availability of applications, server administrators can use iPlanet Application Server Administrator to perform a variety of tasks in real time, without interrupting an application's operation. These tasks include:

- monitoring, reconfiguring, or replacing servers
- swapping out or updating application components

iAS Deployment Tool

The iPlanet Application Server Deployment Tool Features:

- Support for creation and deployment (and enhanced XML editing) of J2EE modules
- Support for J2EE assembly and automated deployment of J2EE applications and components Configuration of security roles, authentication for the application, and binding in LDAP

iAS Installation Tool

iPlanet Application Server Installation Tool provides four types of installation options:

- Express provides the quickest installation. Most configuration parameters are set to default.
- Typical installation lets the administrator override some of the default configuration parameters.
- Custom offers a complete installation. Administrators can configure databases, transactions, and server processes and connections.
- Silent is a non-interactive installation process. Once an administrator has determined the optimal parameters for a given server type, these can be re-used to quickly set up additional installations.

Event Logging and Failure Analysis

iPlanet Application Server provides facilities for logging requests from Web servers and logging system-level and application-level events on iPlanet Application Servers. For deployed applications, system administrators can use contemporaneous logs to assist with failure analysis and to detect attempted security breaches.

Event logging occurs in multiple ways on the iPlanet Application Server:

- Application developers can enable logging in their application logic to assist with failure analysis. For example, an application can log messages like “Transaction succeeded” or “Transaction failed” depending on conditions and events at run-time.
- System administrators can enable automatic event logging to record the messages generated by dynamically loadable modules (DLMs) and application execution.
- System administrators can enable HTTP request logging to record and monitor the requests received by a Web server. Administrators can specify brief, normal, or detailed logging. If logging is enabled, iPlanet Application Server logs information about the HTTP requests to a specified target database. Administrators can then analyze the logs, generate custom reports, and so on. HTTP request logging requires NSAPI or ISAPI Web Connectors.

Support for Third-Party Management Tools

iPlanet Application Server provides the ability to be monitored and managed via SNMP agents such as HP OpenView. SNMP is a protocol used to exchange data about system and network status and activity.

iPlanet Application Server stores variables pertaining to network management in a tree-like hierarchy known as the server's management information base (MIB). Through this MIB, iPlanet Application Server exposes key management information to third-party tools that run SNMP. As a result, iPlanet Application Server can integrate with an enterprise's server management tools, thereby allowing other solutions for remote administration.

Tight Integration with iPlanet Directory Server and iPlanet Web Server

iPlanet Application Server includes the run-time versions of the iPlanet Directory Server and iPlanet Web Server, Enterprise Edition.

iPlanet Directory Server

iPlanet Directory Server, which is packaged with iPlanet Application Server, is iPlanet's implementation of the Lightweight Directory Access Protocol (LDAP). iPlanet Application Server uses iPlanet Directory Server not only to store iPlanet Application Server configuration data but also as a central repository for user and group information. A single iPlanet Directory Server can support multiple instances of iPlanet Application Server. This means that administrative data for all iPlanet Application Server installations can be centralized in one place. iPlanet Directory Server automatically monitors any updates made to iPlanet Application Server clusters or applications. This feature can reduce administration and ensure the most recent applications are loaded.

The iPlanet Application Server Administrator acts as an LDAP client and can access information about users and groups. As a result of this integration with LDAP, iPlanet Application Server provides unified management of users, groups, and roles across the enterprise.

iPlanet Web Server

iPlanet Web Server plug-in for iPlanet Application Server provides performance improvements over other web server platforms; iPlanet Web Server supports servlet and Java ServerPages standards, enabling J2EE application partitioning for performance improvements, load-balancing capabilities, and component fail-over services.

Key Features

Application Model

An application model is the conceptual division of a software application into functional components. The iPlanet Application Server application model promotes code reusability and faster application deployment.

The iPlanet Application Server application model divides an application into multiple layers: presentation, business logic, and data access. Presentation is further separated to distinguish page layout and presentation logic. Data access refers to both databases and other data sources.

The iPlanet Application Server application model is component-oriented. For Java applications, the application model has been enhanced to be fully compliant with the J2EE standard.

The following table lists the main components that make up the functions of an application in the iPlanet Application Server environment:

Functionality in Application	**Application Components**
Presentation logic	Servlets
Page layout	JavaServer Pages
Business logic	Enterprise JavaBeans
Database access	Enterprise JavaBeans using JDBC; query files
Access to other data sources	Connectors, JMS

Industry-Standard Application Components

For developing Java applications, we recommend that you use standard components whenever possible. These standards-based components include servlets, JavaServer Pages (JSPs), and Enterprise JavaBeans (EJBs), described as follows:

- Servlets are Java classes that define page logic and page navigation. Servlets also support the creation or invocation of business components.
- JSPs are web browser pages written in a combination of HTML, JSP tags, and Java.
- EJBs encapsulate an application's business rules and business entities.

In addition, application components can invoke JDBC calls. JDBC is a standard API for database connectivity.

High Scalability

iPlanet Application Server has a scalable architecture. This means that applications can be built to meet the needs of initial deployment. Applications can then be scaled as business needs grow.

Application scaling is accomplished in two main ways: either by adding more servers to a cluster of servers, or by adding more CPUs to a multi-CPU system. Application logic can then be deployed to the new servers. Developers do not need to change any application logic as the user base grows.

In addition, application tasks can be assigned to the server best able to process the request efficiently. This is accomplished either through application partitioning or through dynamic load balancing.

Application Partitioning

The iPlanet Application Server architecture supports application partitioning, which allows logic to be distributed across servers as an application scales to accommodate heavier loads. Using iPlanet Application Server Administration Tool, system administrators can partition an application into functional areas.

For example, in an online catalog application, the application logic for order processing, inventory management, and checkout processing can reside on different servers. Regardless of how applications are partitioned and distributed, the application functions as a single, cohesive unit.

Application logic can also be grouped where each group consists of related operations. For example, a group might contain all logic associated with order processing. Each application component can belong to one or more groups. Applications can also share application logic.

System administrators can deploy these groups of application logic objects locally or globally across application servers in the following ways:

- Portions of an application might uniquely reside on different iPlanet Application Servers, yet still run as a single application. In this way, application logic can be stored on the server that can run it most efficiently. For example, data-intensive application logic can be run on the server that is closest to the data source to avoid latencies associated with accessing remotely located data.
- For load-balanced applications, the same group of application logic objects can be stored on multiple servers. This allows an application to run the application logic more efficiently on the server with the most available resources.
- Applications might dynamically share certain application logic objects. For example, all applications in a network might share the same application logic for user login and authentication, or for credit card authorization.
- Application partitioning gives system administrators tremendous flexibility to scale and tune the performance of applications. In addition, having application components stored on multiple servers helps ensure high application availability in the event of a server shutdown.

Dynamic Load Balancing

In an environment with multiple iPlanet Application Server installations, incoming requests first pass through the Load Balancing System. The Load Balancing System directs the request to the server best suited to process it. iPlanet Application Server offers many load balancing methods, including server load, response time, round robin and weighted round robin mechanisms. These are covered in detail in the Administration section. Administrators can choose the among these load balancing mechanisms, as well as tune the parameters as appropriate.

High Performance

iPlanet Application Server is a high performance, multi-threaded, and multiprocessing application server. iPlanet Application Server can handle a high number of concurrent requests, database connections, and sessions, and provides high performance even under heavy loads.

In addition to the high-performance architecture, iPlanet Application Server offers many features which offer enhanced efficiency.

iPlanet Application Server delivers high performance between web servers, other iPlanet Application Server machines, and heterogeneous back-end data sources through the following features:

- JSP caching
- Result caching
- Database connection caching
- Data streaming
- Multi-threaded capabilities
- Optimized Communication with Web servers

Aside from the application server, other factors affecting application performance include network topology, network and server hardware, database architecture, and application programming.

JSP Caching

iPlanet Application Server 6.0 provides a new feature called JSP Caching, which aids in development of compositional JSPs. This provides functionality to cache JSPs within the Java engine, thereby making it possible to have a master JSP which includes multiple JSPs (similar to a portal page), each of which can be cached using different cache criteria. JSP caching is in addition to result caching.

Result Caching

iPlanet Application Server improves application performance by caching the results of application logic execution. Developers can optionally enable this feature in their applications.

If caching is enabled, iPlanet Application Server saves the application logic's input parameters and results in the cache. The next time the iPlanet Application Server executes the same request, the server can first check the cache to determine whether the input parameters match the cached input parameters. If they match, the server retrieves the results from the cache instead of executing the request again. Result caching is especially effective for large data requests that involve lengthy processing time and for frequently accessed application logic.

iPlanet Application Server has the ability to cache the results of a servlet in order to make subsequent calls to the same servlet faster. iPlanet Application Server caches the results of a request (i.e. a servlet’s execution) for a specific amount of time, so that if another call for that data happens, it can just return the cached data rather than having to perform the operation again.

Database Connection Caching

To improve performance, the iPlanet Application Server caches database connections so that commonly used, existing connections are re-used rather than re-established each time. Connection caching avoids the overhead involved in creating a new database connection for each request.

Application developers can enable database connection caching in their application logic. At run time, when a request creates a new connection to a database, iPlanet Application Server stores the connection in the cache. When the request has completed using the connection, the connection is marked as free in the cache. If a new request is made to the same database by the same user, iPlanet Application Server can first check the cache and use an existing free connection rather than creating a new one. If the freed connection remains unused after a specified time-out period, it is released by the server.

System administrators can use the iPlanet Application Server Administration Tool to specify server-wide settings for database connection caching, such as the initial number of slots in the cache and the time-out limit for free connections, and so on.

Using iPlanet Application Server Administration, system administrators can monitor server performance and tune the number of available connections in the cache. This ensures an optimal ratio of cached connections to system resources.

Data Streaming

iPlanet Application Server provides data streaming facilities. Streaming improves performance by allowing users to begin viewing results of requests sooner, rather than waiting until the complete operation has been processed. Application developers can explicitly control what data is streamed, or allow the system to provide automatic streaming.

Streaming is especially useful for large data sets involving lengthy queries. For example, suppose a user requests a price list containing 10,000 items. The application can process the query and display items to the user as they become available, for example, 40 at a time (typically one full page view), rather than wait until all 10,000 items have been retrieved from the database.

Multi-threaded Capabilities

iPlanet Application Server supports the multi-threading capabilities of the host operating system. An application can optimize performance by processing requests on multiple threads, which maximizes CPU resource utilization.

Application developers automatically take advantage of multi-threading in their applications. In addition, developers can run database operations such as queries, inserts, updates, deletes, and so on, asynchronously. Asynchronous operations allow an application to do other work while a time-consuming operation, such as a large query, runs in the background.

System administrators can use iPlanet Application Server Administration tool to specify settings for multi-threading, such as the following:

- Minimum and maximum number of threads to handle all requests
- Minimum and maximum number of threads to handle asynchronous database requests

Typically, administrators will monitor server performance and tune the number of available threads to achieve an optimal ratio of threads to system resources.

Optimized Communication with Web Servers

iPlanet Application Server optimizes application performance through tighter integration with web servers. This integration occurs using Web Connector Plug-ins and corresponding Listeners. iPlanet Application Server supports NSAPI, ISAPI, and optimized CGI for iPlanet, Microsoft, and CGI-compatible Web servers, respectively.

Session and State Management

iPlanet Application Server supports state and session management capabilities required for web-based applications. iPlanet Application Server provides a number of classes and interfaces that application developers can use to maintain state and user session information.

State and session information is stored on each server in a distributed environment. For example, an application can display a login screen, prompt users to enter their user name and password, then save this information in a session object. Thereafter, the application uses this same information to log in to multiple databases without prompting the user to type it in again.

Similarly, in an online shopping application, a session object can store a list of products selected for purchase (such as quantity, price, and so on) and persistent variables (such as running order totals).

State and session management is especially important for applications that have complex, multi-step operations. In an environment where application logic is partitioned across different servers, system administrators can use the iPlanet Application Server Administration to optionally designate a single server to act as the central repository for all state information. As in previous versions, iAS maintains and replicates distributed user session-information and distributed application-state information.

High Availability

Many enterprise applications must be accessible (available) to users 24 hours a day, 7 days a week. iPlanet Application Server provides a highly available and reliable solution through the use of load balancing and dynamic failover (also called failure recovery).

iPlanet Application Server enables you to distribute all or part of an application across multiple servers. As a result, if one server goes down, the other servers can continue to handle requests. iPlanet Application Server minimizes downtime by providing automatic application restarting. In addition, iPlanet Application Server maintains and replicates distributed user-session information and distributed application-state information. Information is maintained as long as more than one iPlanet Application Server installation is running in a cluster with the server that crashed.

Developers need not be concerned with building recovery and scalability features into their application. The application inherits these features simply by being hosted on the runtime environment.

iPlanet Application Server 6.0 features a set of failover capabilities that promote application availability. These include:

- **Stateful session bean failover** A session bean implements business logic. For example, a bean may hold the contents of an online shopping cart. If there are unexpected fatal problems with the server, the bean fails over to another server, and the user picks up where they left off. Supporting failover for stateful session beans is an iPlanet Application Server value-added feature. J2EE programs do not need any modification to support this iPlanet Application Server failover feature.
- **Rich Client failover** The Rich Client Corba Executive Service (CXS) acts as a bridge between Rich Clients that use the Internet Inter-ORB Protocol (IIOP) and the EJBs on iPlanet Application Server's Java engine(s). If the CXS server within iPlanet Application Server crashes, the state of the bridge objects for all EJBs are restored to that before the crash.

Security

Planet Application Server supports all J2EE security requirements, including role-based authentication, certificate authentication, and form-based authentication.

iPlanet Application Server provides secure web server communication and supports SSL, HTTPS, and HTTP challenge-response authentication to clients. To bridge the security gap between browsers and data sources, iPlanet Application Server supports user authentication, cookies, and database access controls for the secure handling of transactional operations. Event logging and tracking enables detection of, and protection against, unauthorized access.

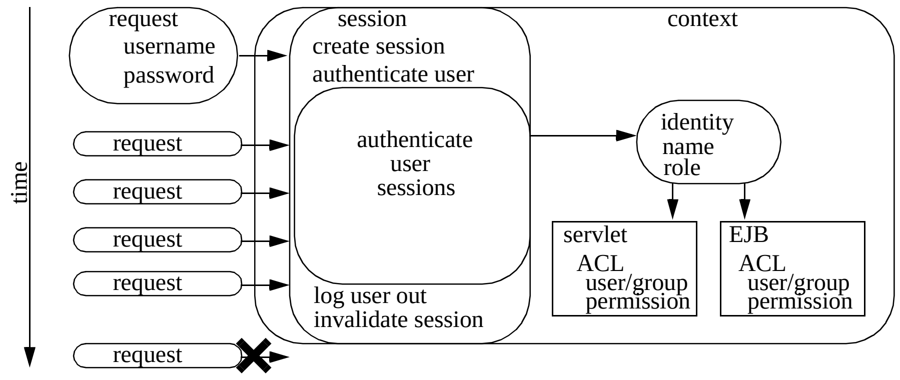
Security Using Access Control Lists

iPlanet Application Server 6.0 also provides for security through Access Control Lists (ACL). In this model, secure applications depends on two kinds of validation:

You authenticate users by comparing a user name and password provided by the user with a user name and password stored in the directory server using LDAP. Authentication is held in the user’s session and remains available until the session expires or the user logs out. Components can retrieve the identity in order to ensure that the caller is authentic.

You create secure components by setting up access control lists (ACLs) that define permissions granted to specific users and groups. These lists are also stored in the Directory Server using LDAP. Components can test for a user’s membership in these groups.

The following diagram shows the basic steps in the security model:



- First, the application establishes a session for the user, and prompts the user to supply a user name and password.

- A servlet authenticates the user name and password by comparing them with the values in the Directory Server, and then “logs the user in” by authenticating the user’s session.
- The server’s context, a programmatic view of the state of the server, recognizes the authenticated session by managing an identity object to components. Components can test authenticity by examining this identity.
- Additionally, components can test the identity to see whether it has permission to perform certain tasks. These permissions are defined in an access control list, which resides in the Directory Server.
- If the user logs out or the session expires, the context can no longer supply the identity, so authentication fails for any requests that require security.

The identity can be used to determine whether a user or group is a member of a certain “role”, so that application flow can be controlled based on a user’s function in the application paradigm. For example, if part of your application is restricted to paying customers only, those customers can be associated with a group called PayingCustomers. You can then write your components to perform tasks based on a user’s membership in that group.

Additionally, each component can have an access control list (ACL) that defines the permissions given to various users or groups with respect to that component. You define the permissions and the users/groups to which they apply either from the iPlanet Application Server Administration Tool or in component configuration files. For example, an EJB that represents an employee database can contain an ACL specifying that members of the group *Employee* can only read the data, while members of the group *Manager* can also update the data.

JMS - Java Message Server

Java^{TM} Message Service (JMS) 1.0.2 provides a set of standard Java language interfaces to Enterprise Messaging Systems, often called Message Oriented Middleware. These interfaces are implemented by products called JMS Providers. The JMS API and provider framework enables the development of portable, message based applications in the Java programming language.

JMS provides connection Pooling and User Mapping:

- Connection Pooling enhances the performance and reliability of iAS applications using JMS via the creation and management of pools of connections from iAS to a JMS-enabled messaging product.
- User mapping speeds iAS application development and eases administration by supporting the easy mapping of users authenticated at the web application level to users, groups, and roles authorized by the JMS-enabled messaging provider

Next Generation Applications with XML

iPlanet Application Server provides complete support for building the next generation of vertical applications using XML. iAS is bundled with the Apache XML parser (Xerces) and XSL processor (Xalan). The rich generating and validating capabilities allow the Xerces-J Parser to be used for:

- Advanced vertical applications that use XML as their data format.
- Instant validation for creating XML editors.
- Creating and maintaining integrity of e-business data expressed in XML.
- Internationalizing XML applications.
- Creating XML aware Web servers.

Xerces provides excellent XML parsing and generation. It conforms to theW3C XML, XML schema and DOM (Level 1 and 2) standards, as well as the defacto SAX (version 2) standard for fully-validating Java parsers. The Xerces parser is component based, modular and can be easily configured.

XML documents are converted to HTML, text and other XML document types using the XSL processor, Xalan. for transforming XML documents into HTML, text, or other XML document types. Xalan-J version 1.0.1 conforms to the Java W3C Recommendations for XSL Transformations (XSLT) and the XML Path Language (XPath).

Chapter 2

Application Server Architecture

This chapter summarizes the design elements of iAS and describes the Application Server structure.

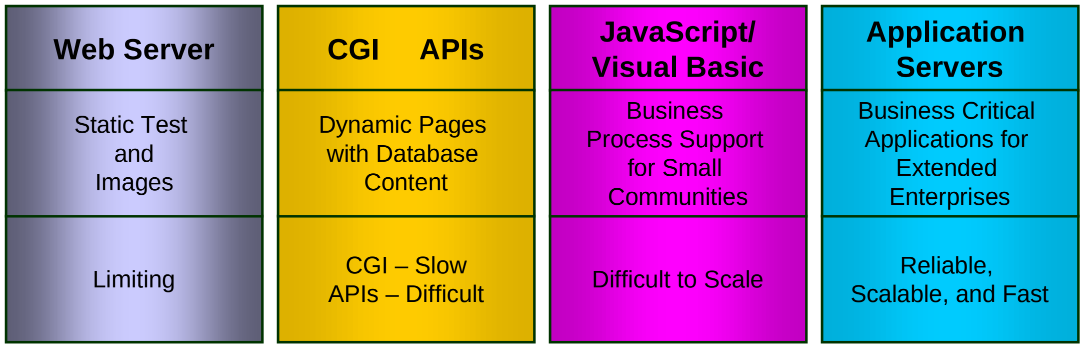
This chapter contains the following sections:

- Application Server Overview
- The Multitiered Environment

Application Server Overview

An Application Server runs the software between a browser and data. For example, when a customer enters an order from a browser, a web server sends the request to the application server which executes logic and also retrieves and updates customer data from back-end sources. The application server runs the business programs instead of the client (browser, rich client), web server or back-end system. It sits in the middle between a client and an enterprise's data and other applications. It physically separates out the business logic from the client and the data into an architecture known as multitier computing. Application servers enables a business to develop and deploy applications quickly and easily and increase the quantity of their users without reprogramming. It can do this because it is on a separate tier.

Application Server Evolution



Web Server	CGI APIs	JavaScript/ Visual Basic	Application Servers
Static Test and Images	Dynamic Pages with Database Content	Business Process Support for Small Communities	Business Critical Applications for Extended Enterprises
Limiting	CGI – Slow APIs – Difficult	Difficult to Scale	Reliable, Scalable, and Fast

Application servers are the next logical step in enterprise web development. They developed from the need to have mission critical applications consistently available to an ever growing number of clients. Additionally these applications needed to be secure and reliable so that regardless of the number of people accessing the system or the source of data, the application server would always be up and running. Prior to application servers, web applications were often run on web servers which are really only designed to serve up web pages. Running and developing applications was slow and complex.

Application servers are part of a multi-tiered architecture. This is an architecture where there is a physical separation between the client which requests information, the programs that process the request and the data that is operated on. The multi-tiered architecture evolved from the mainframe where client, data and process were centralized in the one place. GUI interfaces were rare and remote multiple database access was difficult. Client/Server computing followed in the mid 1980's, where processing was divided between the Client (a PC) and a Server (a mainframe) and requests were usually handled in queries by a relational database system. The presentation and business logic was applied by the PC after receiving processed data from the mainframe. This system allowed for modular development and a GUI but deployment proved problematic.

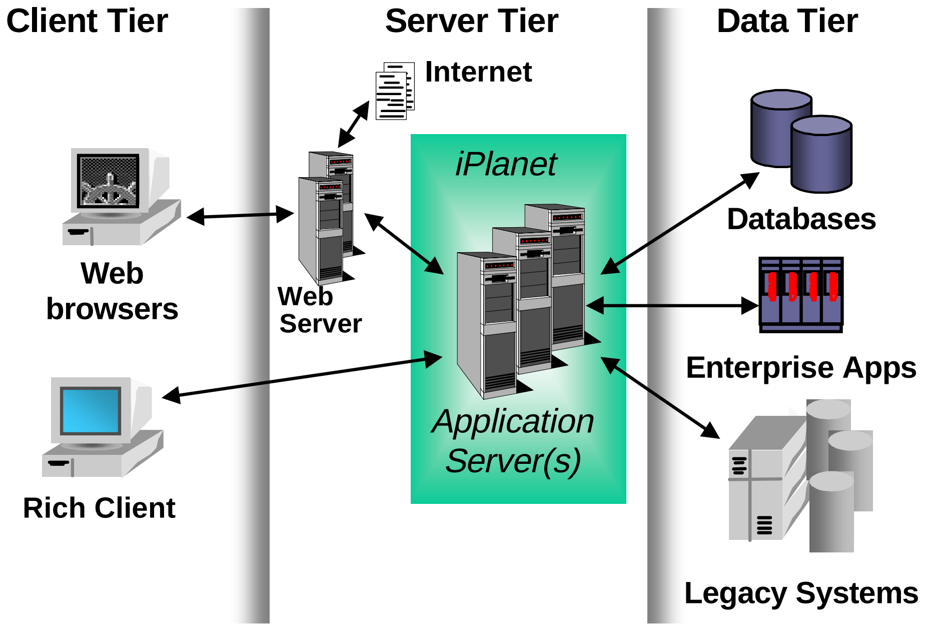
Three-tier computing then divided the presentation logic from the business logic. This separation meant that the business code was independent of how it was presented and where. The business logic layer, now in the middle tier, need not be concerned with what type of client displayed the data. Three-tier was more portable, worked across different types of platforms and allowed for the balancing of requests from the client across multiple servers. Security was easier to implement as the application software was now off the client and costs were substantially reduced. But providing the underlying functions of the middle layer such as transaction processing, security and accessing of the data layer was still complex. The emergence of development tools and a runtime environment to solve this problem came together as the Application Server.

The Multitiered Environment

iPlanet Application Server is the middleware between enterprise data sources and the clients that access those data sources. Business code is stored and processed on iPlanet Application Server rather than on clients. An application is deployed and managed in a single location, and the application is accessible to large numbers of heterogeneous clients.

iPlanet Application Server applications run in a distributed, multitiered environment. This means that an enterprise system might consist of several application servers — computers running the iPlanet Application Server software — along with multiple database servers and web servers. Application code can be distributed among the application servers. Overall, the machines and software involved are divided into three tiers:

- a client tier; the user interface. Requests for data originate here, represented by web browsers or rich clients (such as a Java application).
- a server tier represented by a web server such as iPlanet Web Server Enterprise Edition and an application server which runs the business code such as iPlanet Application Server.
- a data tier, represented by relational databases or other back-end data sources such as Oracle or SAP R/3.



- End users interact with client software, typically a web browser, to use the application.
- When a request originates from a web browser, it is sent to the web server. Assuming the request requires application processing or data access, the web server forwards the request to iPlanet Application Server (iAS).
- When a request originates from a java client, it is sent to iAS by way of an RMI/IIOP link.
- iAS handles requests by running the appropriate application code (and accessing data sources if needed). iAS returns the results to the web server, which in turn forwards the reply back to the client.

Industry Standard Components

iPlanet Application Server is 100% compliant with the Java 2 Platform Enterprise Edition. J2EE is a component based architecture for multi-tiered computing. With J2EE, business logic can be reused and the application can be run on other platforms. Transaction management, life- cycle management, and resource pooling are built into the J2EE platform and provided automatically to the components it supports. Component and application developers are free to focus on specifics such as business logic and user interfaces. The J2EE application model encapsulates the layers of functionality in specific types of components. Business logic is encapsulated in Enterprise JavaBean (EJB) components. Client interaction can be presented through HTML, Java Server Pages (JSP), Servlets and stand alone applets. For more information on J2EE see Chapter 4 “J2EE Concepts” Following is a list of specific standards and components that iPlanet Application Server 6.0 supports:

- JDK 1.2 Specification
- Java Servlet 2.2 Specification
- Enterprise JavaBeans 1.1 Specification
- JavaServer Pages 1.1 Specification
- JDBC 2.0 Core Specification
- JDBC 2.0 Standard Extensions Specification
- JTA 1.0 Specification
- JNDI 1.2 Specification
- RMI-IIOP 1.0.1 Specification
- JavaMail 1.1 Specification

- Java Message Service1.0.2 Specification
- JAF 1.0 Specification
- Corba 2.3 Specification
- HTML
- XML
- LDAP
- SNMP
- XA

Note that for building application components written in C++, iPlanet Application Server provides the Foundation Class Library.

Chapter 3

Architectural Details

This chapter describes the processes, systems, and services that make up the architecture of iAS.

The chapter contains the following sections:

- Server Processes
- System Components

Server Processes

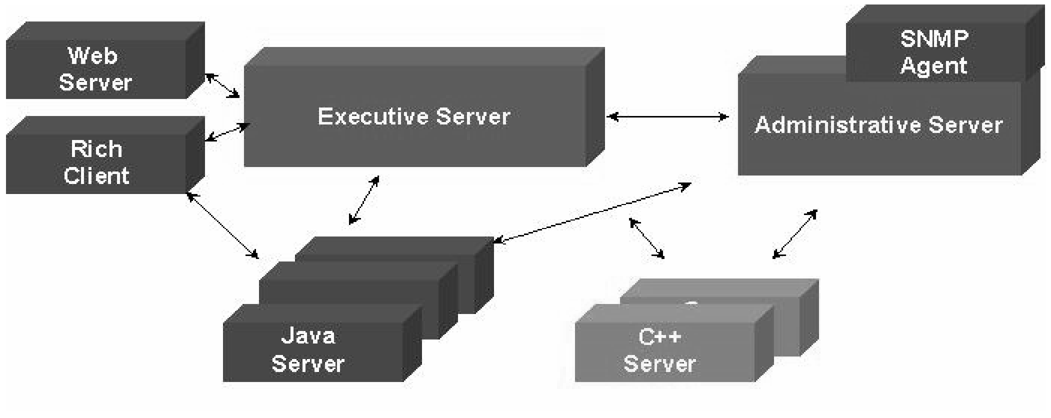
The architecture of iAS includes three main types of internal servers, which are often called engines or processes. They are responsible for all processing within iPlanet Application Server. The following table summarizes the internal servers:

Internal Server	Process Name	Description
Executive Server	KXS	Provides most system services such as Load balancing Dsync, repository and failover.
Administrative Server	KAS	Provides system services for iAS administration and failure recovery.
Java Server	KJS	Provides services to Java applications.

Note that C++ applications are supported in iPlanet Application Server 6.0 for compatibility. All new applications should be developed using J2EE tools.

Summary of Process Interactions

The following figure shows how the four iPlanet Application Server processes interact, and the relationship of the application server to other complementary services:



When a web server forwards requests to iAS, the requests are first received by the Executive Server process (KXS). The KXS process forwards the request either to a Java Server process (KJS). A KJS process runs Java programming logic. Note that for migration purposes a KCS, C++ engine exists and runs C++ programming logic. KCS runs in a similar manner to the KJS engine.

Each KJS process maintains a specified number of threads and runs the programming logic to completion on those threads. The results are returned to the web server and sent on to the client browser.

The iAS technique of processing application requests is the key to reducing the load on a web server, thereby providing faster response time. A server administrator can configure the iAS environment for best performance by

- adding any number of iAS machines.
- specifying any number of KJS processes.
- maintaining any number of threads on each process.

In addition to providing high performance, the internal processes make it possible for iAS to remain available 24 hours a day, 7 days a week. If a KJS process goes down, the KXS process restarts it. And if the KXS process itself fails, then it is restarted by an additional process—the Administrative Server process (KAS). Additional monitoring in iAS makes sure that the KAS process is always running.

If all iAS processes go down, then other iAS machines in the cluster will take over, (this assumes a multiserver environment.) In addition, iAS can replace the index page with a redirected page and also send notifications by email and FAX to alert the system administrator and redirect the system to a different site.

The next several sections describe the internal servers in more detail.

The Executive Server

The Executive Server is the main engine in the iPlanet Application Server. The Executive Server is responsible for hosting many of the system-level services as they are needed by iAS.

The Executive Server process (KXS) also distributes application requests to the appropriate application process, Java Server.

For example, here is what happens when an application request comes into iPlanet Application Server:

1. The Executive Server invokes the request manager, a system-level service.
2. The request manager assigns a thread from the pool to the request and forwards the request to the appropriate Java Server which loads the application class if necessary and executes the application logic.
3. When the request completes, the thread is returned to the thread pool.

The Administrative Server

The Administrative Server enables administration of one or more iPlanet Application Servers. It registers with the appropriate server or servers all changes made to the system and application settings using iAS Administrator Tool (iASAT), the GUI administration tool.

The Administrative Server also hosts the failure-recovery service that restarts the other server processes if they become unavailable. This failure-recovery service provides a high degree of fault tolerance for iPlanet Application Server.

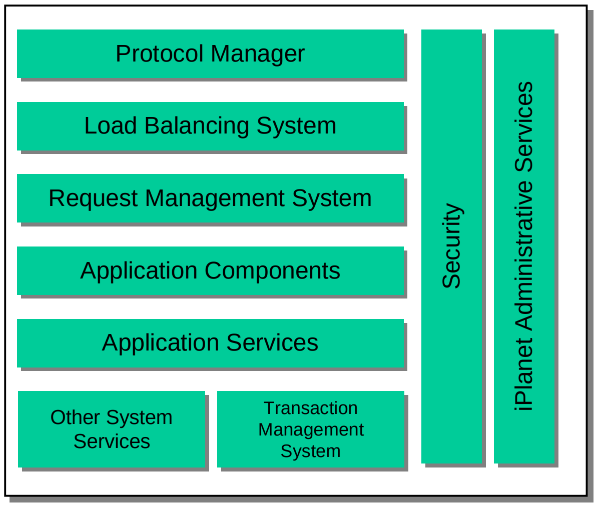
The Java Server

The Java Server processes is the application server. Business logic components written in Java are hosted in the Java Server. Business logic components are the core of the application, holding the compiled code instructions written by the developer.

The Java Server (KJS) also hosts the application-level services. These services are dynamically loaded into the appropriate process as needed by an application component. For example, when an EJB requires access to a database, the Java Server loads in the data access engine, uses its services, and then dismisses it. The database connection provided by the data access engine is cached in the Java Server active memory. If another request enters the system and accesses the same database, the cached connection is used. In this way, iAS reduces the need to invoke the data access engine, thereby increasing the performance of request processing.

System Components

This section describes the main internal systems that make up the iAS architecture. These systems are shown in the following figure:



This section describes the following topics:

- Protocol Manager
- Load Balancing System
- Request Management System
- Application Components
- Application Services
- System Services
- Transaction Management System
- Security
- Administrative Services

Protocol Manager

Communication between a web server and iPlanet Application Server occurs through NSAPI, ISAPI, and optimized CGI. Optimization and superior performance are achieved through optimized communication via a plug-in on top of the internal protocol. Internal components manages different protocols as they are encountered.

When a request comes in from a web browser, the request is passed to the web server via the HTTP or HTTPS protocol. The request is processed by the appropriate web connector. Web connectors include the NSAPI web connector, ISAPI web connector, and optimized CGI web connector for iPlanet, Microsoft, and CGI-compatible web servers, respectively.

Load Balancing System

In an environment with multiple iAS installations, incoming requests first pass through the Load Balancing System. The Load Balancing System directs the request to the server best suited to process it. The Load Balancing System includes a Load Monitor and a Load Balancer. There are a number of types of load balancing:

iAS employs several load balancing methods for optimum performance:

- Weighted round robin load balancing
- Response time load balancing
- System load based load balancing

In addition, the following load balancing types are supported

- Intraserver process level load balancing
- JSP load balancing
- Rich client load balancing

Administrators can choose the best method and once implemented requests are routed to the most appropriate iAS machine.

Request Management System

Incoming requests are handled by the Request Management System. iAS is multi-threaded, and the Request Management System assigns threads from a dynamic thread pool to process the requests. The Request Management System enables the simultaneous processing of a high volume of requests. System administrators can configure thread pool parameters for optimal request processing. The Request Management System includes the following subsystems:

- The Thread Manager provides a dynamic pool of threads. From this pool, a thread is assigned to process the request.
- The Queue Manager gets involved when requests must be queued until a thread becomes available.The Queue Manager manages the list of pending requests and descriptive information. This information includes things such as the unique request ID and a request's current processing status, such as *waiting*, *in process*, *finished*, and so on.
- Request Logging, if enabled by the system administrator, keeps an information log of web server requests in a back-end database or in log files.

Application Components

Each incoming request identifies one or more application components. These components, in turn, are identified by their globally unique identifier, or GUID. GUIDs are checked against iAS's Global Directory Service (GDS) and iPlanet Directory Server, an LDAP server. If necessary, iPlanet Directory Server authenticates the request by checking against known role information. The appropriate application component is then executed to process the request. For example, the request may invoke a servlet, which in turn may call one or more EJBs.

For more information about application components, see the *Programmer's Guide*.

Application Services

Application services enable management of application functions, such as user sessions, application states, cookies, email notification, result caching, and so on. These services are invoked by application components using API calls. Application services are loaded into a KJS process. The following sections summarize the services available to Java applications:

Services Hosted by KJS

The following services are available to applications written in Java: Services Hosted by KJS only S

Application Service	Description
State and session management	Manages user session information, such as user login, page navigation information, and "shopping cart" selections. Manages persistent state information. Distributed iAS machines can use a state workspace to share information.
Cookie management	Generates HTTP cookies for cookie-aware web browsers. For non-cookie aware browsers, emulated cookies are embedded in URLs or hidden fields.
Data access management	Provides and manages access to databases.
Transaction management	Manages database transactions, providing commit and rollback support for those transactions. See "Transaction Management System" for more information.
Database connection pooling	Caches database connections so that future access for the same database is provided immediately.
Result caching	Caches result-set data so future requests can be processed more efficiently. If the request has been stored in the result cache, the previously computed result is returned immediately. Otherwise, the application logic is executed and the result is processed. System administrators can configure result cache settings such as the number of cache slots, time-outs, and cache cleaning interval.
Application events	Based on time criteria or other event criteria, allows applications to send and receive emails, to invoke Java applications,to invoke a servlet. This is useful for administration.
HTML streaming	Provides streaming of data back to HTML clients so as to return data more efficiently.
Connectors	Allow enterprise applications to integrate with applications deployed on iAS. Connectors are persistent modules that are dynamically loaded into iAS and are accessed by multiple EJBs over the life of the extension. Although connectors act as application services, connectors can also be considered as application components.

The following services are available only to applications written in Java:

Application Service	Description
JSP compiler	Interprets JSP tags, the HTML-like tags that determine the layout of pages sent to a web browser. The compiler supports Version 1.1 of the JavaServer Pages specification.
Servlet container	Contains and manages servlets through their life cycle by providing network services over which requests and responses are set, by decoding MIME-based requests, and by formatting MIME-based responses. Supports HTTP and HTTPS.
EJB container	Provides a home for EJBs and manages the beans it contains. Management involves registering beans, providing a remote interface for them, creating and destroying instances, checking security, managing their active state, and coordinating distributed transactions. The EJB container can also manage all persistent data within the bean and includes a full global transaction manager.
Distributed transactions	Supports transactions involving multiple databases (of different types or in different locations). A distributed transaction is invoked from an EJB and uses the built-in transaction processing manager of iAS.
LDAP support	Eases management and security by providing a central repository for information about users and groups.

System Services

System services increase the efficiency with which application requests are processed. These services are not directly used by applications, but rather provide additional application support outside the scope of application logic. There is no API access to system-level services. A description of these services is provided in the following table. (Note that some of the following services are described elsewhere in this chapter.)

Table 3-1

System Service	Description
Protocol management	Manages communications with clients by supporting the various protocols used by the iPlanet Application Server.
Request management	Manages requests as they arrive at the server, routing them to the proper processes (the Java Server) and managing request thread allocations.
Global Directory Service	Repository for all application server metadata information
JNDI	The Java Naming and Directory Interface (JNDI) is a standard extension to the Java platform. The JNDI API provides Java applications with a unified interface to multiple naming and directory services in the enterprise.
Event logging	Maintains a log of application logic execution. Application developers can enable logging in their application logic to assist with debugging and tuning. In addition, system administrators can enable automatic event logging, which records the messages generated by dynamically loadable modules (DLMs) and application logic objects when processing user requests. Event logging can run in all processes.
Load balancing	Determines how application request loads are balanced among multiple servers.
Application result caching	Caches application results so that future requests for the same application components by the same user are handled immediately.
Failure recovery	Restarts the Executive Server, or Java Server processes if they ever become unavailable.
Distributed data synchronization	Supports failure recovery by synchronizing data. Data is synchronized not only across all KJS processes running in iAS, but also across all iAS installations within a cluster.
SNMP support	Provides access to iAS via SNMP agents, thereby allowing remote management from third-party administration tools.

Table 3-1

System Service	Description
Kernel services	Provide low-level services to all other services and subsystems. Examples of kernel services include language-binding engines and the lock manager.

Transaction Management System

The transaction management system provides support for the EJB transaction model. It is also responsible for the transparent propagation of the transaction context across processes and two phase commit coordination.

Local versus Global Transactions

For J2EE applications, iAS supports both local transactions and global transactions.

Global transactions can span multiple databases of potentially heterogeneous types,. Global transactions are managed and coordinated by the transaction manager, and can span multiple databases and processes. The transaction manager typically uses the XA protocol to interact with database back-ends. Global transactions occur using a two-phase commit from Encina, a transaction manager built into iPlanet Application Server.

Local transactions involve access to a single database. They provide better application performance because they are less complex. Local transactions are native to a single database and are restricted within a single process.

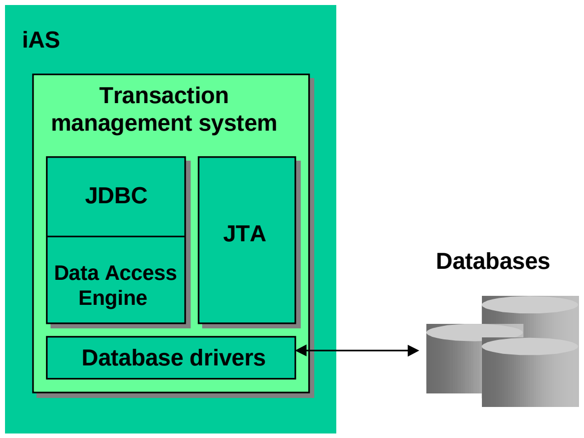
Global transactions can only be started declaratively through EJBs. By contrast, local transactions can only be executed programmatically, from either servlets, JSPs, or EJBs.

Both the JDBC and iPlanet Application Server APIs rely on the Data Access Engine to interact with database drivers. iPlanet Application Server provides native support for the following database drivers: Oracle, DB2, Informix, Sybase, and (on Windows NT only) SQLServer. There is also an ODBC driver. iPlanet Application Server can automatically configure the drivers if they are installed before iPlanet Application Server.

The transaction management system also includes the Java Transaction API (JTA). JTA is used for managing connections to a single database. JTA specifies local Java interfaces between the transaction manager and the other transaction elements (which include iPlanet Application Server and the transactional application). JTA provides a Java mapping of the industry-standard X/Open XA protocol, which is used for distributed database applications.

Architectural Details

The following figure shows the architectural details of the transaction management system:



Both the JDBC and iAS APIs rely on the Data Access Engine to interact with database drivers. iAS provides native support for the following database drivers: Oracle, DB2, Informix, Sybase, and (on Windows NT only) SQLServer.

The transaction management system also includes the user transaction interface specified in the JTA specification. This interface allows the application developer to explicitly demarcate transactions. For more information see the JTA specification and *Programmer's Guide*.

Security

Security in iAS is role based as specified in the J2EE specifications. Roles are defined by the application assemblers and are accorded permissions to access bean methods, servlets and JSPs. These roles are mapped at deployment time to LDAP users and groups. See the *Administration and Deployment Guide* or the *Programmer's Guide*.

Administrative Services

Administrative services run in KAS, the Administrative Server process. KAS enables remote administration of servers and applications. KAS also supports other services, such as application partitioning, event logging, request monitoring, and dynamic configuration of key server settings.

Clients that access administrative services include iAS Administration Tool, iPlanet Directory Server, and third-party SNMP agents. For more information, see the *Administration and Deployment Guide*.

Chapter 4

J2EE Concepts

This chapter describes the Java 2 Platform, Enterprise Edition and its role in the iPlanet Application Server.

This chapter contains the following sections:

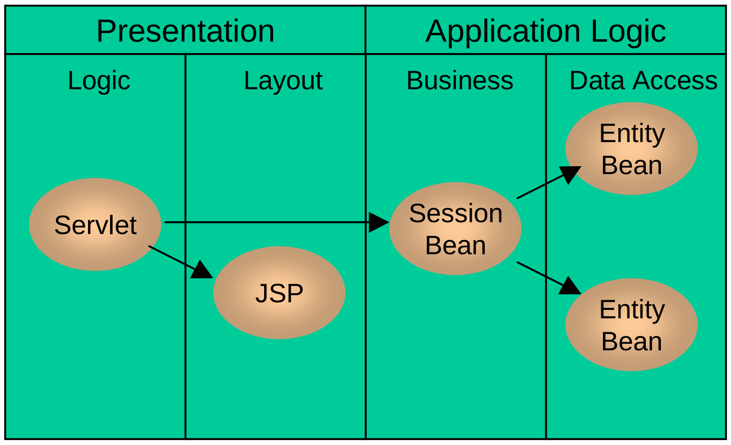
- J2EE Concepts
- J2EE Components
- The J2EE Programming Model

J2EE Concepts

J2EE solves the problem of the cost and complexity in developing multi-tiered services that are scalable, highly available, secure and reliable. It achieves this by providing an open standard architecture through the J2EE Platform and the J2EE Application Model. This platform allows developers to focus on the business logic while J2EE handles all the low level details. With J2EE, services are easily enhanced and rapidly deployed, allowing business to quickly react to competitive changes.

J2EE is an open environment for developing and deploying multi-tiered services where thin-client applications invoke business logic that executes on an application server such as the iPlanet Application Server. It comprises of a set of services, application programming interfaces and protocols. The Java programming language the Java Virtual Machine and Java Bean components are the foundation of J2EE.

The Multi-Tiered Model



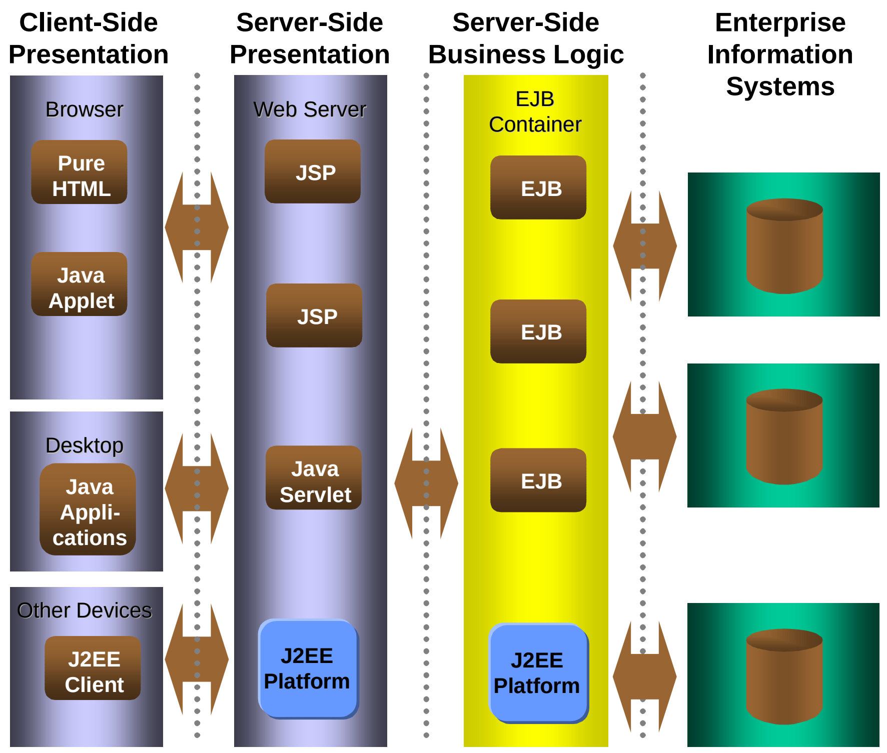
In the graphic above, the servlet receives the request, validates input, calls the session bean and calls the JSP. The JSP formats the HTML and responds to the client. The session bean validates the request, executes the process, and enforces transactions. The entity beans manage the data.

In a multi tiered model, the first tier is the client and is usually a web browser or stand alone Java application. This invokes the business logic on one or more middle tiers running on dedicated hardware, which in turn accesses data from the Enterprise Information Service on the third tier.

Developing a multi-tiered service requires client applications, business and presentation logic (the applications that get, update and present data) and infrastructure code. The infrastructure is the low level system components that access various databases, system resources and provide security. Infrastructure details are handled by J2EE in iAS so the developer needs only to develop the business and presentation logic.

In the middle tier, business logic is implemented as Enterprise Java Beans components (EJB), while the presentation logic is implemented as Java Server Pages (JSP) and Servlets. Servlets and JSPs allow the separation of the request processing from the presentation logic. The presentation layer of the model allows easy access to the middle-tier business functions. JSP technology allows developers to present dynamically created web pages. Servlets allow developers to create dynamic presentations to users completely in the Java programming language.

J2EE Benefits



- J2EE provides a seamless Java solution across all layers.
- Separation of tasks in the layered platform architecture into business logic away from system services and the user interface, placing business logic it in a middle tier between the two.
- Open set of standards— EJB, JSP, servlets, JDBC, JNDI, RMI make up the J2EE platform.
- Portability— The Java feature of “Write Once, Run Anywhere”(WORA) is in all these technologies. This means that any third party provider can write specially designed components for iAS, reducing the cost of development.
- Scalability— Scalability is the responsibility of the iAS - the application does not need to code for this. Enterprise application systems support high scalability by using a multitier, distributed application architecture with the integrated components.

- Components— The presentation logic, business logic, and data access logic are separated into suitable components and deployed on multiple servers. This enables s an application to take advantage of the high performance of multi threaded and multiprocessing systems.
- Language and API experiences can be leveraged when dealing with another area. For example, experience of writing a Java applet is similar to writing a servlet. Experience of writing a servlet is similar to writing an EJB.
- Sandbox, Garbage Collection, and Exception Handling automatically reduces the problem of one component blocking the operation of the server or another component. For example, servlets allow you to plug custom code into a Web server as a plug- in or extension library would. The difference however, is that servlets have garbage collection and exception handling abilities, so a problem in a servlet should not affect the server operation. EJBs function in the same way and can be deployed on a database or application server.

J2EE Components

Transaction management, life- cycle management, and resource pooling are built into the J2EE platform and provided automatically to the components it supports. Component and application developers are free to focus on specifics such as business logic and user interfaces. The J2EE application model encapsulates the layers of functionality in specific types of components. Business logic is encapsulated in Enterprise JavaBean (EJB) components. Client interaction can be presented through the following:

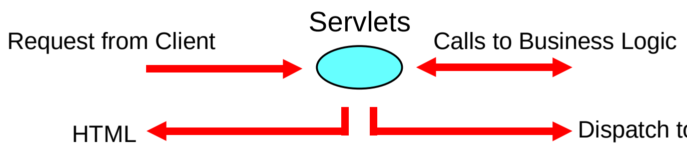
- Plain HTML Web pages
- Web pages powered by Java applets
- Java Servlets API
- JavaServer Pages technology
- Stand- alone Java applications

Components communicate transparently using various standards, including HTML, XML, HTTP, SSL, RMI, IIOP, and others.

J2EE is made of different components;

- **Servlets—** an efficient platform- independent replacement for CGI scripts responding to client requests.
- **JavaServer Pages (JSP)—** a type of server- side scripting, which can dynamically generate Web pages.
- **Enterprise JavaBeans (EJB)—** server- side session management, business logic encapsulating and abstractions for accessing persistent data.
- **Java Database Connectivity (JDBC)—** an API that describes a standard Java library for accessing data sources.
- **Transaction Support— declarative transactions for components where transactions can span components and processes.**
- **Java Naming and Directory Interface (JNDI)—** an abstract interface to name binding and directory search services.
- **Remote Method Invocation (RM/IIOP)—** an enabling technology for distributed object communication.
- **CORBA Compatible—** CORBA complements Java by providing a distributed objects framework, services to support that framework and interoperability with other languages.

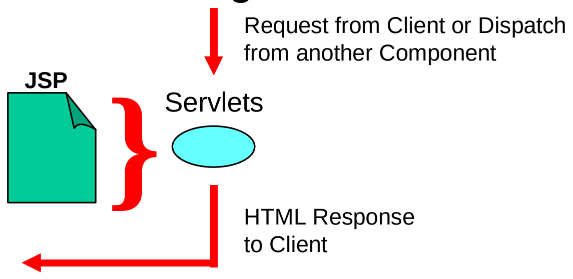
Servlets



In J2EE, servlets manage the presentation logic of an application by acting as a central dispatcher for applications by processing form input, invoking business logic components by accessing Enterprise JavaBeans, and formatting page output using JSPs. Servlets control the application’s flow from one user interaction to the next by generating content in response to a request from a user.

Servlets are used to handle request and response from browser clients. Servlets are like applets except they run on a server instead of a client.

Java Server Pages



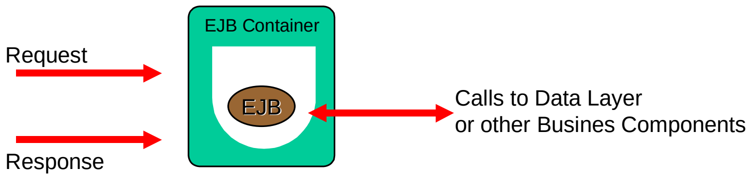
JavaServer Pages (JSPs) are the presentation layout mechanism. They are browser pages in HTML or XML. They can optionally contain Java code, which enables them to perform complex processing, conditionalize output, and communicate with other objects in your application. JSPs in iPlanet Application Server 6.0 are based on the JSP 1.1 specification

In iPlanet Application Server applications, you use JSPs as the individual pages that make up your application. You can call a JSP from a servlet to handle the output from a user interaction, or, since JSPs have the same access to the application environment as any other application components, you can use a JSP as a destination from an interaction.

JSPs are compiled into servlets, either when installed or the first time they are called. This makes JSPs available to the application environment as standard objects and enables them to be called from a client using a URL.

You can think of servlets and JSPs as opposite sides of the same coin: each can perform the tasks of the other. However, because JSPs are written as HTML files, with embedded Java code, they are best suited for layout tasks. Servlets are best suited as central dispatchers for incoming requests.JavaServer Pages (JSPs).

Enterprise Java Beans



Enterprise JavaBeans (EJBs) are applications. If servlets act as the central dispatcher for your application and handle presentation logic, EJBs do the bulk of your application's actual data and rules processing but provide no presentation or visible user- interface services. EJBs enable you to partition your business logic, rules, and objects into discrete, modular, and scalable units. Each EJB encapsulates one or more application tasks or application objects, including data structures and the methods that operate on them. Typically, EJBs also take parameters and send back return values. EJBs always work within the context of a “container,” which serves as a link between the EJBs and the server that hosts them. As an iPlanet Application Server developer, you need not worry about the container for your EJBs. The iPlanet Application Server software environment provides the container. This container provides all the standard container services denoted in the EJB 1.1 specification. It also provides additional services such as stateful session bean failover. In fact, the container can handle all remote access, security, concurrency, transaction control, and database access. Because the actual implementation details are part of the container, and there is a standard, prescribed interface between a container and its EJBs, the bean developer is freed from having to know or handle platform- specific implementation details. Instead, the enterprise bean developer can create generic, task- focused EJBs for use with any vendor’s products that support the EJB Standard.

Session Beans and Entity Beans

There are two kinds of EJBs: entity and session. Each of these bean types is used differently in a server application. An EJB can be an object that represents a stateless service, an object that represents a session with a particular client (and which automatically maintains state across multiple client-invoked methods), or can be a persistent entity object possibly shared among multiple clients. A session bean implements business logic. All functionality for remote access, security, concurrency, and transactions is provided by the EJB container. A session EJB is a private resource used only by the client that creates it. More information on session beans can be found in the EJB 1.1 specification.

Entity EJBs

Entity beans most commonly represent persistent data. This data is maintained directly in a database, or accessed through a back-end application as an object. A simple example of an Entity Bean would be one that is defined to represent a single row in a database table, and where each instance of the Bean represents a specific row. A more complex example would be an Entity Bean designed to represent complicated views of joined tables in a database where each instance of the Bean would represent a single customer’s shopping cart contents.

Unlike Session Beans, Entity Bean instances can be accessed simultaneously by multiple clients. The container, is responsible for synchronizing the instance’s state by using transactions. This delegation of responsibility to the container, means the Bean developer does not need to worry about concurrent access to methods from multiple transactions.

The persistence of an Entity Bean can either be managed by the bean itself, or by the bean’s container. When the Entity Bean manages it’s own persistence, it’s called Bean-managed persistence. When the bean delegates this function to the container, it’s called container-managed persistence (CMP).

Bean-managed Persistence. The bean developer must implement persistence code (such as JDBC calls) directly in the EJB class methods, if the bean is to manage it's own persistence. The possible downside of this implementation is the loss of portability, if a proprietary interface is used, and also the risk of tying the bean to a specific database.

Container-managed Persistence (CMP). The container provider uses the iPlanet Application Server Deployment Tool (iASDT) to generate the bean code to implement the container persistence process. The container manages, transparently to the bean, the persistence state. The bean developer does not need to implement any data access code in the bean's methods. Not only is this method simpler for the bean developer to implement, but it makes the bean fully portable, without any ties to a specific database.

Finally, any number of entity beans can be installed in a container. The container implements a home interface for each entity bean. The home interface enables a client to create, look up, and remove entity objects. A client can look up an entity bean's home interface through the Java Naming and Directory Interface (JNDI).

The J2EE Programming Model

The iPlanet Application Server 6.0 Application Server is Java 2 Platform, Enterprise Edition specification version 1.2 (J2EE 1.2) compliant and is based on standards developed by the Java community, including servlets, JavaServer Pages, and Enterprise JavaBeans. iPlanet Application Server 6.0 programming model is for Java applications only. C++ applications continue to use the NAS 2.1 model.

iPlanet Application Server 6.0 is backward compatible with NAS 2.1. applications. NAS 2.1 applications can run on iPlanet Application Server 6.0 without code alteration. NAS 4.0 applications are compatible with conversion to the J2EE standard and do need some conversion.

Application flow is similar between the iPlanet Application Server version 6.0 model and the previous version 4.0 and 2.1 models. Each user interaction is handled by one (or more) application components that process the inputs, perform business logic functions, interact with a database, and provide an output page that answers the input and sets up the next user interaction. The new programming model describes three tiers of application logic, each of which is represented by a set of components or APIs.

Programming Tier	**NAS 2.1 component**	**NAS 4.0 component**	**iAS 6.0 component**	**Description**
Presentation Logic	AppLogic	Java servlet and proprietary standards	Java servlet	Controls the application's interface to the user by processing requests, generating content in response, formatting and delivering that content back to the user. In 6.0, servlets process incoming requests and orchestrate the response. Business logic is normally offloaded to EJBs, and output is usually offloaded to JSPs.
Presentation Layout (part of Presentation Logic)	HTML template	JavaServer Page (JSP) and proprietary standards	JavaServer Page (JSP)	Controls the appearance of each page. Part of the presentation logic, usually handled by JavaServer Pages. JSPs are HTML pages that contain embedded Java, and thus are much more versatile and powerful than 2.1 HTML templates.
Business Logic	AppLogic	Enterprise JavaBeans (EJBs) and proprietary standards	Enterprise JavaBeans (EJBs)	Controls business logic. EJBs enable business logic to be persistent across calls, offer improved caching, and are designed to work closely with JDBC for database transactions.
Data Access Logic	DAE	JDBC and proprietary standards	JDBC	Controls database storage and retrieval. The JDBC API is available to all Java components, as are all APIs, though database transactions are usually controlled by EJBs in the 6.0 model.

Presentation Logic and Layout

Presentation logic describes the flow of an application from the perspective of each user interaction: request processing, followed by content generation and delivery. The goal of presentation logic is to create a logical answer to a request, and to prompt for another request. The goal of presentation layout is to display the content of this answer in a predetermined format. Application functions such as user sessions, security and user authentication, and input validation are also handled by the presentation logic.

In short, presentation logic involves everything related to the application's interface with the user.

In the NAS 2.1 programming model, presentation logic was controlled by an AppLogic, while layout was handled by an HTML template. At run-time, the AppLogic provided output to populate the template.

In the iAS 6.0 programming model, presentation logic is usually handled by a Java servlet. Layout is usually handled by a JSP. At runtime, the servlet uses a JSP to format the content generated by the business logic.

The two major alternatives to this basic model are as follows:

- Handle all presentation logic and layout for a given interaction in a JSP. This can be an easy way to control an interaction that has no business logic and little to process from the previous interaction. For example, the "front page" for an application often requires no processing at all.
- Handle all presentation logic and layout in a servlet. This can be efficient for interactions that have very little layout. For example, a simple database report might just list the rows retrieved from a database query. It doesn't make sense to incur the overhead of a JSP call when the page can be simply output from a servlet.

Business Logic

Business logic describes the activities that involve the generation of specific content: storing and retrieving data, and performing computations on that data. The goal of business logic is to perform the activities that generate or determine answers to questions posed by the presentation logic.

In short, business logic involves the content provided by and generated for the application.

In the NAS 2.1 programming model, business logic was controlled by the same AppLogic that handled the presentation logic for a given user interaction.

In the iAS 6.0 programming model, business logic is usually handled by one or more Enterprise JavaBeans (EJBs), which control database transactions and encapsulate the results. EJBs are powerful, reusable components that empower applications with a great deal of flexibility, since EJBs can be invoked or inspected from any other object and can be made to be persistent.

One alternative to this model is to handle business logic in the presentation logic (servlets and/or JSPs), much the same way that AppLogics handled business logic. This can be efficient for short, directed business events such as specific directory requests, but this approach lacks the flexibility and power that EJBs bring to the programming model.

Data Access Logic

Data access logic describes transactions with a database or directory server. The goal of data access logic is to provide an interface between an application and the set of data that concerns it. Data access is normally performed as a function of business logic.

In short, data access logic involves the storage and retrieval of the content collected or generated by business logic.

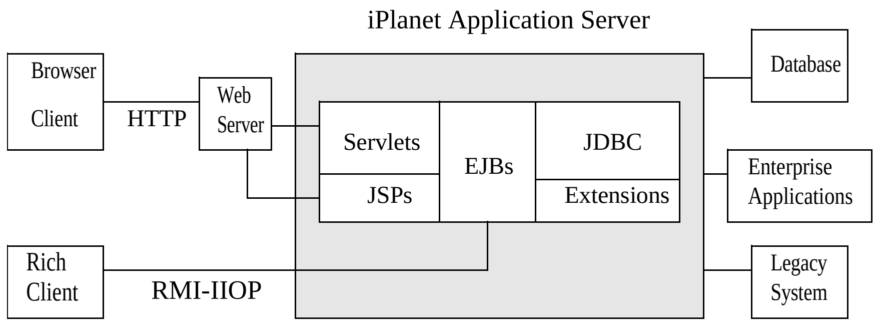
In the NAS 2.1 programming model, data access logic was controlled by calls made from an AppLogic using APIs from several classes and interfaces, including the `DataSet`, `DBDataSet`, and `DBStoredProcedure` classes and the `ICallableStmt`, `IColumn`, `IDataConn`, `IDataConnSet`, `IHierQuery`, `IHierResultSet`, `IListDataSet`, `IPreparedQuery`, `IQuery`, `IResultSet`, `ITable`, `ITrans`, and `IValList` interfaces.

In the iAS 6.0 programming model, data access logic is handled by the JDBC standard set of APIs. The previous APIs are all deprecated in iAS 6.0.

Rich Clients

The Rich Client is a stand-alone Java program that can directly access EJBs deployed on iPlanet Application Server. Traditionally, clients communicated with iPlanet Application Server through the web-path, i.e. by speaking HTTP to server components such as JSPs and Servlets which in turn had access to EJBs within the context of the server. The J2EE v1.2 specification, however, requires that stand-alone clients be able to talk to iPlanet Application Server using the RMI-IIOP standard. Chapter 9 of the J2EE v1.2 specification also requires that these stand-alone clients operate within the context of an Application Client Container (ACC) that isolates server-specific issues, leaving the clients completely portable. Through its Rich Client infrastructure, iPlanet Application Server allows Java clients to directly access EJBs on iPlanet Application Server. These clients could operate within an ACC that ships with iPlanet Application Server as required by the J2EE ACC specification or more straight forward direct access (non ACC path) the way Java programmers are used to writing them.

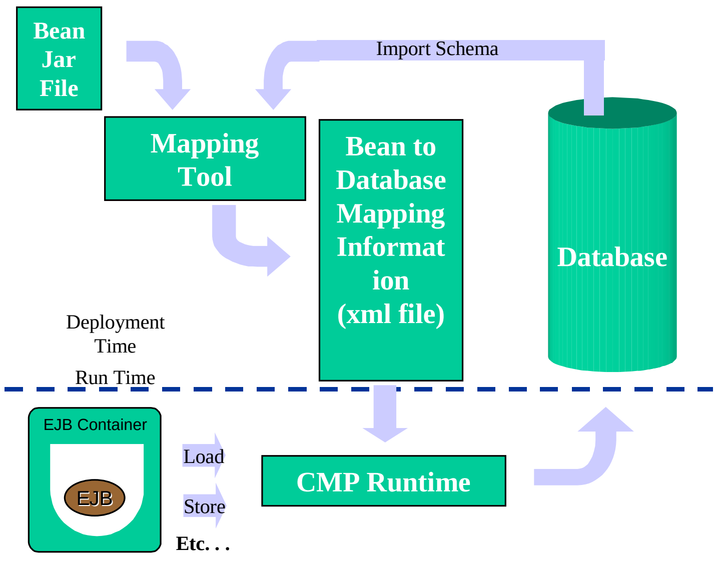
The diagram below is a schematic representation of the iPlanet Application Server architecture and illustrates the difference between the Rich Client and web paths. While browser clients communicate with iPlanet Application Server using HTTP through a Web Server, Rich Clients circumvent the Web server and directly access EJBs as RMI-IIOP clients. Failover support is also provided for entity bean failover in case of a Corba Executive Service (CXS) crash



The Rich Client is a first-tier program that executes in its own Java Virtual Machine (JVM), possibly in an ACC. Deploying the Rich Client requires the specification of deployment descriptors using XML.

Container Managed Persistence

If you want the EJB container to manage the storing of entity bean variables to an underlying resource manager, then the iPlanet Application Server Deployment Tool is used to configure the EJB container with container managed persistence (CMP) settings.



iPlanet Application Server implements CMP with:

- Support for the J2EE v 1.2 specification CMP model.
- Support of multiple pluggable pooling and persistence managers on a per bean basis.
- Provide user with a deployment tool (iASDT) to perform the object relational (O/R) mapping and create the separate CMP deployment descriptor XML files.
- Support for Commit Options B and C
- Support for sophisticated custom finder methods
- Pluggable Persistence Managers

Pluggable persistence managers allow users to set the persistence policies for a bean at deployment time. This flexible framework allows a customer to use a third party persistence manager of choice at deployment time. Factory classes for persistence manager can be set in the bean's deployment descriptor XML file. The classes specified in these fields will be used by the container to create either zero or one persistence managers for a bean.

iPlanet Application Server also supports sophisticated Object to Relational (OR) mapping middleware. CocoBase, from Thought, Inc., provides a dynamic repository based Object to Relational mapping tool which delivers CMP and Bean Managed Persistence.

RMI-IIOP

RMI over IIOP combines the best features of RMI with the best features of CORBA. Like RMI, RMI over IIOP speeds distributed application development by allowing developers to work completely in the Java programming language. When using RMI over IIOP to produce Java technology-based distributed applications, there is no separate Interface Definition Language (IDL) or mapping to learn. Like RMI, RMI over IIOP provides flexibility by allowing developers to pass any serializable Java object (Objects By Value) between application components. Like CORBA, RMI over IIOP is based on open standards defined with the participation of hundreds of vendors and users in the Object Management Group. Like CORBA, RMI over IIOP uses IIOP as its communication protocol. IIOP eases legacy application and platform integration by allowing application components written in C++, Smalltalk, and other CORBA supported languages to communicate with components running on the Java platform.

Using JDBC for Database Access

In iAS, Enterprise JavaBeans (EJBs) support database access primarily through the JDBC API. iPlanet Application Server supports all of JDBC 2.0 API including result set enhancements, batch updates, distributed transactions, row sets, and JNDI support for datasource name lookups.

While this section assumes familiarity with JDBC 2.0, it also describes specific implementation issues that may have programming ramifications. For example, the JDBC specifications do not make it clear what constitute JDBC resources. In the specifications, some JDBC statements—such as any of the `Connection` class methods that close database connections—release resources without specifying exactly what those resources are.

JDBC Overview

JDBC is a set of Java classes and methods that let you embed database calls in your server applications. That’s all you need to know in order to start using JDBC in your server applications.

More specifically, JDBC is a set of interfaces that every server vendor, such as iPlanet, must implement according to the JDBC specifications. iPlanet Application Server provides a JDBC type 2 driver which supports a variety of database back-ends. This driver processes the JDBC statements in your applications and routes the SQL arguments they contain to your database engines.

Chapter 5

iAS Product Family

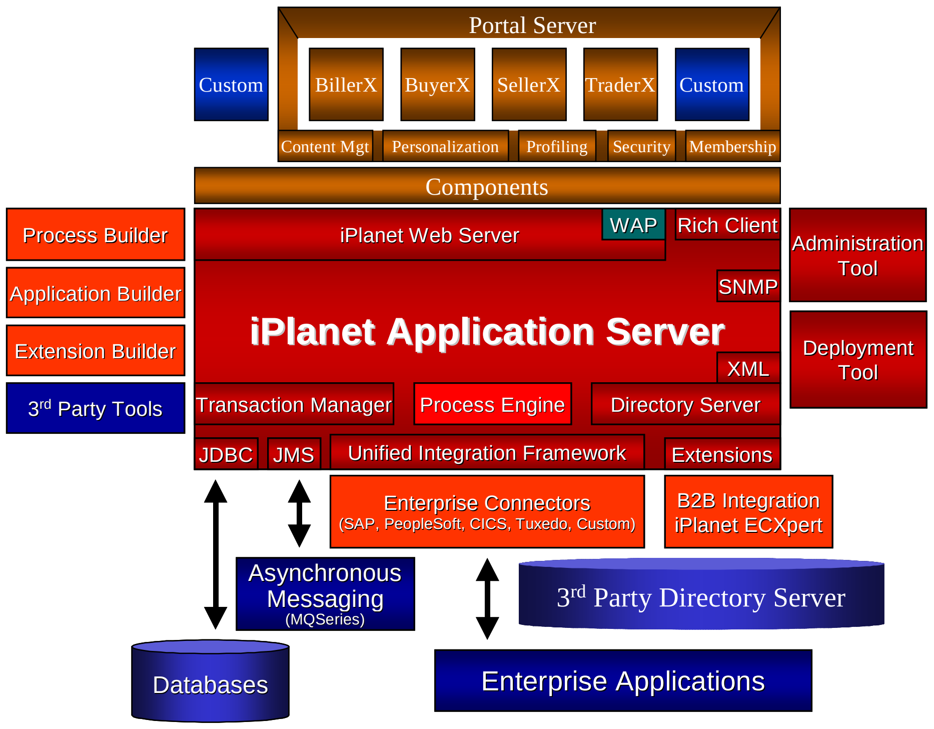
This chapter describes the applications and services that integrate with the iPlanet Application Server.

This chapter contains the following sections:

- iPlanet Application Builder
- iAS Deployment Tool
- iAS Administration Tool
- iAS Enterprise Connectors
- iPlanet Unified Integration Framework
- iPlanet Process Manager
- Sample Applications
- Encina Transaction Manager
- iPlanet Directory Server
- iPlanet Web Server Enterprise Edition

iAS in the iPlanet Framework

iAS operates seamlessly with other iPlanet and many third party products to provide a comprehensive enterprise solution.



iAS integrates with any J2EE third party product and with all iPlanet solutions, some of which are shown above. Solutions to track, profile and maintain customers as well as include all company relationships such as banks, customers, suppliers and employees are available such as the fully service extensible applications like Process Builder, Biller Xpert, Trader Xpert and Seller Xpert.

The iPlanet Application Server Product Family

In addition to the core application server, the iPlanet Application Server product line includes a comprehensive set of products and tools to help your organization quickly and efficiently create business critical applications. These include:

iPlanet Application Builder

An Internet application development tool designed to simplify the creation of multi-tiered enterprise-class applications that run on iPlanet Application Server. iPlanet Application Builder provides an intuitive and productive web development environment that enables developers to leverage the rich, prebuilt application and infrastructure services of Application Server. By targeting the distributed multitier application model of iPlanet Application Server, iPlanet Application Builder enables developers to rapidly build sophisticated, business-critical web applications for the Internet.

iPlanet Application Builder 6.0 can be used with third party development tools, including Symantec Visual Café, Macromedia Dreamweaver, Inprise JBuilder, WebGain Studio, and others.

iPlanet Application Builder makes application development easy by offering powerful programming integrated development environment, including the following functions:

- Java code editing with integrated builds and testing
- Debugging support using third-party IDEs
- WYSIWYG html editing
- Java code editing along with building, testing, and integrated debugging support
- Visual data modeling and point-and-click SQL editing
- HTML page and JSP design with point-and-click data binding
- Client-side Javascript support
- Deployment support
- Integrated source control

Wizard-based development, including the following features:

- Various wizards designed to guide you in creating parts of your application, allowing you to quickly prototype.

- Many of the files that iPlanet Application Builder creates for you, especially Java files, contain auto-generated code to get you started, including wizard support for JDBC RowSet objects for point-and-click data binding, Java servlets, EJBs, and JavaServer pages.

iAS Deployment Tool

An application must be deployed before it can be used, and iAS Deployment Tool is a GUI tool that makes application deployment easier. You access this tool either from iAS Administration Tool or from iPlanet Application Builder. The deployment tool can also be used stand alone.

The iPlanet Application Server Deployment Tool Features:

- Support for creation and deployment (and enhanced XML editing) of J2EE modules
- Support for J2EE assembly and automated deployment of J2EE applications and components Configuration of security roles, authentication for the application, and binding in LDAP

For detailed information about using iAS Deployment Tool, see the *Administration and Deployment Guide*

iAS Administration Tool

iAS Administration Tool (iASAT) is a GUI tool that contains several smaller tools for managing one or more iAS machines or applications.

When you deploy an application, iAS Deployment Manager installs all the application's files and registers all of its components on the destination server (a server on which iAS has been installed).

iPlanet Application Server Administrator enables the following capabilities:

- Remote management of multiple servers and distributed applications.
- Dynamic deployment and scaling of applications.
- Performance tuning and optimization of the server environment.
- Management and tuning involves tasks such as adjusting database connection threads, adjusting load-balancing parameters, configuring web servers, and managing roles
- Event Logging and Failure Analysis
- Security features including viewing and management of security roles

- Transaction Management features for local or global transactions
- Application management features for J2EE applications

For detailed information about using iAS Administration Tool, see the *Administration and Deployment Guide*.

Dynamic Application Management

iPlanet Application Server's architecture allows partitioned applications to run even if one or more servers fail. In a load-balanced server configuration, application logic can be replicated on multiple servers. If a server fails, the load balancing module dynamically directs requests to other available servers, thus preventing application-wide failure.

Because the iPlanet Application Server architecture promotes high availability of applications, server administrators can use iPlanet Application Server Administrator to perform a variety of tasks in real time, without interrupting an application's operation. These tasks include:

- monitoring, reconfiguring, or replacing servers
- swapping out or updating application components

iPlanet Unified Integration Framework

The iPlanet Unified Integration Framework is a toolkit that enables development of server extensions that integrate with new, web-based enterprise applications and systems, client-server applications, and third-party Internet solutions. These extensions provide a consistent access layer to disparate back-end systems, dramatically reducing development effort. The framework provides support for features such as object-pooling, distributed state and session management, template streaming, and multi-threading enables high-performance, fault-tolerant integration that can scale to tens-of-thousands of users. Corporate IT developers, application vendors, and system integrators can easily build server extensions to iPlanet Application Server in Java or C/C++.

The Unified Integration Framework offers:

- Faster time to market
- Enhanced manageability
- Enhanced application performance
- Enhanced Web-based application solutions
- Lower development costs

iAS Enterprise Connectors

iPlanet Application Server Enterprise Connectors include packaged solutions for CICS, Tuxedo, SAP R/3 and PeopleSoft. Companies that want to extend their assets — and decrease their time to market in the Net Economy — can quickly and easily convert their legacy data into rich, dynamic Internet application services. iPlanet Enterprise Connectors provide “out-of-the-box” integration using the iPlanet Unified Integration Framework — developers can add new services on top of native logic without needing to learn the native back-end APIs. Developers can extract both native legacy data and logic, and store it into a metadata repository. New, Internet-ready services are added to the logic and the application can then be deployed to the Internet without any modification to native code. New services are rendered as Enterprise JavaBeans, components which can be reused across multiple applications.

- Packaged Enterprise Connectors offer “out-of-the box” integration capability to common legacy and enterprise resource planning (ERP) systems
- All iPlanet integration solutions work together in any combination on a single iPlanet Application Server installation
- Provides support for Internet standards including Java 2 Enterprise Edition (J2EE), XML, WAP, JMS
- Consistent development model and API utilize a common integration framework
- Comprehensive solutions convert existing enterprise, legacy and ERP assets into dynamic Internet services

iPlanet Process Manager

A comprehensive web-based solution for designing, deploying, managing, and participating in automated business processes such as claims management, customer self service, and order fulfillment. An intuitive development environment and advanced scalability and reliability features allow an enterprise to easily extend information on Enterprise Resource Planning (ERP) systems, mainframes, and custom applications to employees, partners, suppliers, and customers. It includes iPlanet Process Builder, iPlanet Process Express, iPlanet Process Administrator, iPlanet Application Server, iPlanet Web Server, and iPlanet Directory Server.

iPlanet Process Manager helps streamline communication and business processes by providing authorized employees, partners, suppliers, and customers with immediate, real-time access to core business processes and applications using a standard web browser. This allows participants in automated processes to search for information in a variety of categories: work in progress, application, process instance, date, and user. iPlanet Process Manager also allows an enterprise to customize user interfaces without the need for custom software.

Sample Applications

iAS includes sample web-based applications, enabling you to quickly learn techniques for developing and deploying applications in a iAS environment.

These Sample Applications are fully functional and J2EE compliant. See the developers section of the iPlanet web site for these and upcoming examples.

- Java Pet Store
- Bank
- Fortune
- J2EE developers examples

One sample presents a bookstore application that simulates browsing, searching, and ordering books online. This Java application demonstrates the iAS application model that uses industry-standard components such as servlets, JavaServer Pages, Enterprise JavaBeans, and data access with JDBC. For information about installing or using the online bookstore application, see the *Installation Guide* or the *Programmers Guide (Java).*

Another sample presents a banking application that simulates a user session with an online account. This sample demonstrates techniques for migrating existing applications to comply with the industry-standard Java application model. For information about installing the bank application, see the *Installation Guide*. For details about the application code, see the *Migration Guide*.

Encina Transaction Manager

iPlanet Application Server integrates the Encina transaction monitor as a core feature of the server for optimal performance, reliability, and manageability. The Encina transaction monitor provides reliability in area of distributed transactions. Global transactions are coordinated within a Java Server with the Transaction Manager. Global Transactions are a set of related operations that must be executed as a unit, though each operation may run in a different process.

You can use global transactions to update a database that uses one or more EJBs running concurrently for the same global transaction, from within one or more KJS processes. This occurs when an EJB triggers another EJB to run and they both participate in the same transaction. You can also update multiple databases that are distributed over different geographic locations or update multiple databases of different types, such as Oracle and Sybase).

iPlanet Directory Server

iPlanet Directory Server (iDS) provides a comprehensive, enterprise-wide directory service for managing information about users, groups, and access control lists. iAS 6.0 includes iPlanet Directory Server, which supports versions 2 and 3 of the Lightweight Directory Access Protocol (LDAP). iAS uses the Directory Server for storing configuration information.

- IDS automatically monitors any updates made to iAS clusters or applications. This reduces the burden on system administrators when adding or modifying J2EE applications ensuring the most recent applications are available.
- iDS manages password policies and user groups for iAS.
- iDS Stores information on location and availability of components in iAS. iDS stores application configuration information and access controls for J2EE application components.
- iDS clusters alongside iAS clusters to ensure high availability of server configuration.

- iDS integration with iAS provides significant performance improvements over using flat file or RDBMS systems for user and application information.

For more information, see the data sheet for iPlanet Directory Server 6.0.

iPlanet Web Server Enterprise Edition

iPlanet Web Server Enterprise Edition, (iWS) is best suited for the Enterprise and Service Provider segment, particularly for e-commerce sites. Designed to handle the extremely high demands of managed sites, while minimizing the IT workload required to build, secure, and maintain the. iWS:

- Maximizes uptime through intelligent load balancing, process monitors, dynamic log rotation, and support for multiple processes on UNIX®
- Delivers a personalized user experience through its high-performance Java(TM) application platform supporting Java Servlets, JavaServer Pages(TM), and in-process, plugable Java Virtual Machines
- Performs optimally at high load because of its multi-process, multi-threaded architecture, HTTP 1.1 compliance, and support for SSL hardware accelerators
- Eases management of complex websites with millions of users through delegated administration, cluster management, SNMP monitoring and tight integration with iPlanet Directory Server

Index