

Getting Started Guide

iPlanet™ Application Server

Version 6.5 SP1

October 2002

Copyright © 2002 Sun Microsystems, Inc., 901 San Antonio Road, Palo Alto, California 94303, U.S.A. All rights reserved.

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

This product is distributed under licenses restricting its use, copying distribution, and decompilation. No part of this product may be reproduced in any form by any means without prior written authorization of Sun and its licensors, if any.

Third-party software, including font technology, is copyrighted and licensed from Sun suppliers.

Sun, Sun Microsystems, the Sun logo, Java, Solaris, iPlanet and the iPlanet logo are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and other countries.

All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. in the U.S. and other countries. Products bearing SPARC trademarks are based upon architecture developed by Sun Microsystems, Inc.

UNIX is a registered trademark in the U.S. and other countries, exclusively licensed through X/Open Company, Ltd.

This product includes software developed by Apache Software Foundation (<http://www.apache.org/>). Copyright (c) 1999 The Apache Software Foundation. All rights reserved.

Federal Acquisitions: Commercial Software - Government Users Subject to Standard License Terms and Conditions.

Copyright © 2002 Sun Microsystems, Inc., 901 San Antonio Road, Palo Alto, California 94303, Etats-Unis. Tous droits réservés.

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

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

Le logiciel détenu par des tiers, et qui comprend la technologie relative aux polices de caractères, est protégé par un copyright et licencié par des fournisseurs de Sun.

Sun, Sun Microsystems, le logo Sun, Java, Solaris, iPlanet et le logo iPlanet sont des marques de fabrique ou des marques déposées de Sun Microsystems, Inc. aux Etats-Unis et dans d'autres pays.

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

UNIX est une marque déposée aux Etats-Unis et dans d'autres pays et licenciée exclusivement par X/Open Company, Ltd.

Contents

Preface	7
About This Guide	7
What You Should Know	7
How This Guide is Organized	8
Documentation Conventions	8
Chapter 1 iPlanet Application Server Overview	11
What's an Application Server?	11
iPlanet Application Server Features	13
J2EE Platform	14
Industry Standard Components	14
High Scalability	15
Application Partitioning	15
Distributed Data Synchronization (DSync)	16
High Performance	17
Dynamic Load Balancing	18
Database Connection Pooling	18
Result Caching	19
Data Streaming	20
Multi-threaded Capabilities	20
EJB Session Pooling	21
Optimized Communication with Web Servers	21
High Availability	21
Rapid Application Development	22
Multi-tier Application Model	22
Core Services	24
Application Development Tools	24
Session and State Management	25
Security	25
User Authentication	26

Secure Access to Data Sources	26
Product Components	26
Programming APIs	27
System Services and Application Services	27
Sample Applications	27
Core Application Server Components	28
Web Connector Plug-In	28
iPlanet Application Server Administration Tool	28
iPlanet Application Server Deployment Tool	29
iPlanet Directory Server	29
Chapter 2 Deploying Applications	31
Starting PointBase Database Server	31
Stopping and Starting the iPlanet Application Server	32
To Stop and Start the Server on Windows	32
To Stop and Start the Server on Solaris	34
Registering the Data Source	34
To Register the Data Source on Solaris and Windows	35
Building the Application	35
To Build the Application	35
Viewing the Application	36
Undeploying Applications	37
Chapter 3 Administration and Deployment Tools	39
iPlanet Application Server Administration	39
Basic Administrative Tasks	40
To start the server	40
Command Line Tools	40
Using Command-Line Tools	40
To Start and Stop the Server	41
Deploying Applications	41
To use GUI based tools	41
To access the command line Deployment Tool	41
Logging Application Messages	42
How Log Messages Are Formatted	42
Determining the Logging Destination	43
iPlanet Directory Server Administration	43
Support for Third-Party Management Tools	43
Chapter 4 Troubleshooting iPlanet Application Server	45
Installation Issues	45
Core Server Issues	46

Administration Tool Issues	46
Deployment Issues	47
Database Support Issues	48
 Chapter 5 Additional References	 49
 Appendix A iPlanet Application Server Architecture	 53
Server Processes	53
Summary of Process Interactions	54
The Executive Server	55
The Administrative Server	56
The Java Server and C++ Server	56
System Components	56
Protocol Manager	57
Load Balancing System	59
Load Manager	59
Load Balancer	59
Request Management System	59
Application Components	60
Application Services	61
Services Hosted by Either KJS or KCS	61
Services Hosted by KJS Only	62
System Services	63
Transaction Management System	64
Local versus Global Transactions	65
Architectural Details	65
Security	66
Administrative Services	66
 Index	 67

Preface

This chapter describes the contents of iPlanet Application Server *Getting Started Guide*. It contains the following sections:

- About This Guide
- What You Should Know
- How This Guide is Organized
- Documentation Conventions

About This Guide

This *Getting Started Guide* discusses the various features of iPlanet Application Server and how to deploy a sample application. This guide also includes an architectural overview of iPlanet Application Server.

This manual is intended for system administrators, network administrators, evaluators, application server administrators, web developers, and software developers who want to get an understanding of the various tasks and tools available with iPlanet Application Server.

What You Should Know

Before you begin, you should already be familiar with the following topics:

- Application Servers
- Client/Server programming model
- Internet and World Wide Web

- Windows NT/2000 or Solaris™ operating systems
- Java programming and J2EE

How This Guide is Organized

This guide is organized as follows:

Chapter 1, “iPlanet Application Server Overview”, gives an overview of iPlanet Application Server features and iPlanet Application Server components.

Chapter 2, “Deploying Applications”, describes the steps to deploy a sample application to iPlanet Application Server.

Chapter 3, “Administration and Deployment Tools”, describes the various tools that are available to administer and deploy applications.

Chapter 4, “Troubleshooting iPlanet Application Server”, contains information on common errors and their workarounds.

Chapter 5, “Additional References”, contains references to other iPlanet Application Server documentation and resources.

Appendix A, “iPlanet Application Server Architecture”, describes the J2EE concepts and iPlanet Application Server architecture.

Documentation Conventions

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

This guide uses URLs of the form: `http://server.domain/path/file.html`, where:

- `server` is the name of the server where you are running the application.
- `domain` is your internet domain name.
- `path` is the directory structure on the server.
- `file` is an individual filename.

The following table shows the typographic conventions used throughout iPlanet documentation.

Table 1 Typographic Conventions

Typeface	Meaning	Examples
Monospaced	The names of files, directories, sample code, and code listings; and HTML tags	Open <code>Hello.html</code> file. <HEAD1> creates a top level heading.
<i>Italics</i>	Book titles, variables, other code placeholders, words to be emphasized, and words used in the literal sense	See Chapter 8 of the <i>Getting Started Guide</i> . Enter your <i>UserID</i> . Enter <i>Login</i> in the Name field.
Bold	First appearance of a glossary term in the text	Templates are page outlines.

iPlanet Application Server Overview

This chapter contains the following topics:

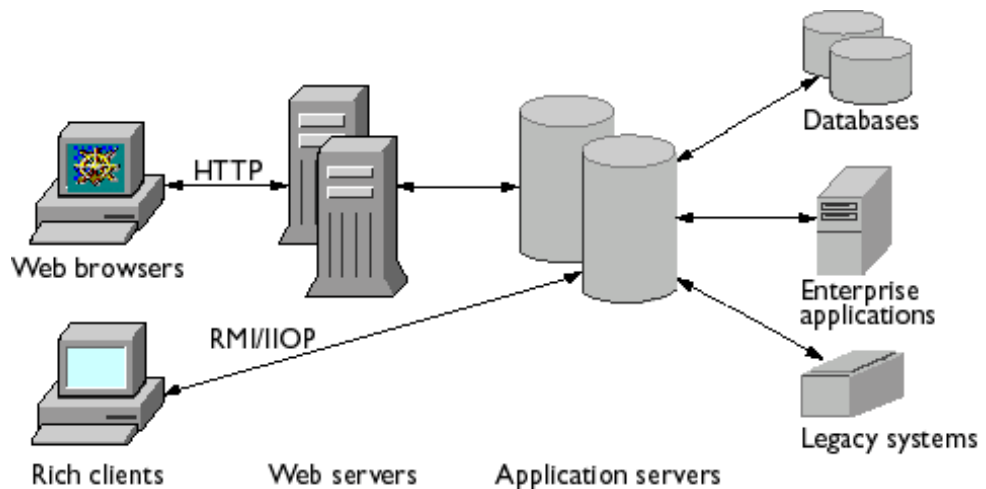
- What's an Application Server?
- iPlanet Application Server Features
- Product Components

iPlanet Application Server is designed with performance, scalability, and availability at its foundation. These qualities enable iPlanet Application Server to meet the massive increase in user size and processing that occurs in an e-commerce Web site or an internet-enabled enterprise.

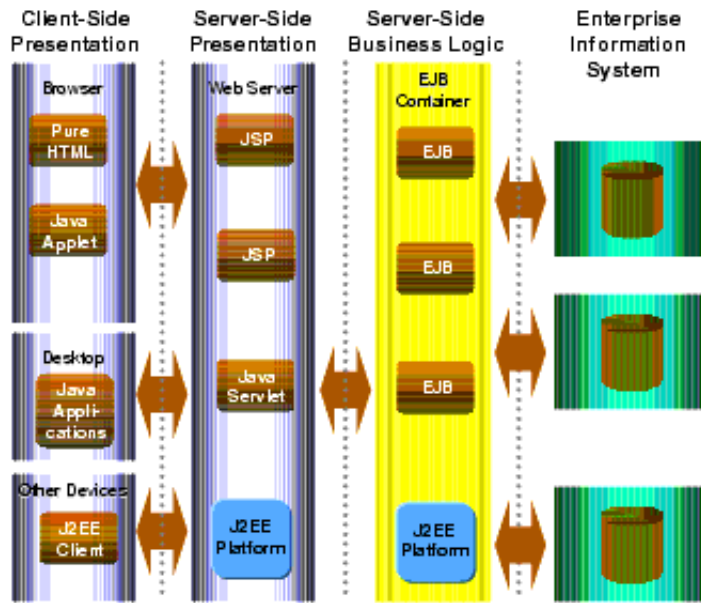
What's an Application Server?

Application servers provide the framework for a client to connect to a backend source, execute the applications logic, and return the result to the client. The application server occupies the middle-tier in the three-tier computing model.

Figure 1-1 The application server space on the Internet.



The most popular kind of application server is the Java™ application server. iPlanet Application Server is a Java application server and is fully compliant with the Java 2 Enterprise Edition (J2EE™) specifications. 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.

Figure 1-2 Application server components based on the J2EE platform

iPlanet Application Server Features

The following features make iPlanet Application Server 6.5 one of the most scalable, robust, and high-performance application servers available today:

- J2EE Platform
- High Scalability
- High Performance
- High Availability
- Session and State Management
- Rapid Application Development
- Security
- Product Components

J2EE Platform

iPlanet Application Server provides a fully compliant Java 2 Enterprise Edition (J2EE) application server platform. The J2EE application model allows developers to focus on the business logic while J2EE components handle all the low level details. Therefore, applications and services can be easily enhanced and rapidly deployed, allowing business to quickly react to competitive changes.

By providing an open standard architecture through the J2EE Platform, iPlanet Application Server solves the problem of the cost and complexity in developing multi-tiered services that are scalable, highly available, secure and reliable. For more information, see “iPlanet Application Server Architecture,” on page 53.

Industry Standard Components

Application development is based on this open industry Java standard. Following is a list of specific standards and components that iPlanet Application Server 6.5 supports:

- JDK 1.3 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
- JavaMail 1.0
- Application Assembly and Deployment (XML)
- HTML
- XML
- LDAP
- SNMP AGENTS
- XA

High Scalability

iPlanet Application Server has a scalable architecture that allows you to meet the needs of a growing business without reprogramming. Developers need not change any application logic as the user base grows. For example, an enterprise with a three server cluster can add fifty more servers dynamically while iPlanet Application Server continues to function. Adding more resources allows an increase in the number of transactions and requests leading to an improvement in performance.

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 is then deployed to the new servers. Scaling is integral to iPlanet Application Server by the nature of its distributed architecture; the business software runs over several physically separate servers. Each server usually has a copy of the applications and the administrator engine determines at run time which server in the cluster will run a module. Application tasks can be assigned to the server best able to process the request efficiently.

Because it has a scalable architecture, iPlanet Application Server can efficiently employ both of the following:

- Application partitioning.
- Distributed Data Synchronization (DSync)

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.

Application logic can also be grouped so that each group consists of related operations. Each application component can belong to one or more groups. Applications can also share application logic. Using the Dynamic “Point- and-Click” partitioning feature of the iPlanet Application Server Administrator Tool, system administrators can do the following:

- Monitor component load in real time
- Instantly change load monitoring parameters
- Manage application components
- Deploy components to any or all machines in a cluster
- Deploy based on component or machine suitability

- Disable components so they are still available, ready for instant activation

When an order is entered, the order processing component is always called from Server 1; the checkout function, however, can be called from any of the three servers. Regardless of how applications are partitioned and distributed, the application functions as a single, cohesive unit.

Application objects can also be grouped so that each group consists of related operations. For example, a group might contain all the EJBs 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 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 objects can be stored on the server that can run it most efficiently. For example, data-intensive application objects 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 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.

Distributed Data Synchronization (DSync)

Distributed Data Synchronization provides cluster management and data synchronization across processes, aiding iPlanet Application Server scalability.

A cluster is a set of iPlanet Application Server servers that share a single source of session (client's interaction information) and state (open data from databases, variables) data. A cluster maintains integrity of shared session and state data and is used when applications need to share session and state data and are not necessary to run iPlanet Application Server.

When a cluster is in use, a Sync Primary is accessed by the other servers to retrieve session and state data.

Clusters are easily managed and added through the Administration tool. DSync ensures that distributed session information (for example, user id), state data (for example, stock quotes) and stateful session information (for example, shopping cart) are available throughout the cluster. In the case of a server failure, the information is still available on a Sync Backup with no loss of data.

DSync and cluster management are features of the scalability and availability of iPlanet Application Server. DSync also plays an important role in server availability; see the section on “High Availability,” on page 21. Detailed information on DSync can be found in *iPlanet Application Server Administration Guide*.

High Performance

iPlanet Application Server is a high performance, multi-threaded, and multi-processing 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.

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

- Dynamic Load Balancing
- Database Connection Pooling
- Result Caching
- Data Streaming
- Multi-threaded Capabilities
- EJB Session Pooling
- 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. These topics are beyond the scope of the Getting Started Guide and therefore not described in this guide.

Dynamic Load Balancing

iPlanet Application Server supports dynamic load balancing to provide optimal performance levels under heavy loads. Load balancing ensures that each application component is processed by the system best able to handle the component's requirements at that moment, thereby assuring optimum performance.

The Load Balancer uses the load and performance statistics retrieved by the Load Monitor to determine the server with the most available resources to handle an incoming request. Each instance of iPlanet Application Server has its own load-balancing module which makes routing decisions based on load balancing properties.

Load balancing can be server driven or controlled via a plug-in. With load balancing enabled, the iPlanet Application Server can direct certain requests to be run on an available server instead of waiting for a busy server to become available. If one server is overburdened, another can take its place to process the request. The Load Monitor tabulates information on server resource availability. Simpler methods of load balancing such as weighted round robin load balancing can be employed to reduce the amount of server statistics collected.

To use load balancing, application logic must be partitioned across all iPlanet Application Servers that might process the application logic at run time. System administrators must determine if an entire application or specific parts of an application should be load balanced.

For optimal performance and resource utilization, system administrators can deploy application logic on servers that are best configured to handle execution of that logic. There are many options an administrator can use to load balance requests. One is 'sticky' load balancing, which routes all requests in a given client session to the same server that handled the first request. This is useful for caching, and for objects that cannot distribute session or state information across servers.

Partitioning characteristics are dynamically known to all servers in the cluster. iPlanet Application Servers regularly update their load statistics and broadcast them to other iPlanet Application Servers in the cluster. Based on load balancing factors, requests are dynamically routed to servers. Load balancing factors are configured using iPlanet Application Server Administration.

Database Connection Pooling

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

At run time, when an application creates a new connection to a database, the application is actually creating a virtual connection: iPlanet Application Server creates and manages the “real” connection and links it with the application’s virtual connection. When the application is not using the connection, the connection is marked as free in the pool. If a new request is made to the same database with the same permissions (perhaps in a different session or even a different application), iPlanet Application Server can use an existing free connection rather than creating a new one. This is called connection sharing. If the freed connection remains unused after a specified time-out period, it’s released by the application server.

Typically, Web applications grant similar access permissions to users by group, also known as roles. For security purpose, each role grants its users only the connection permissions they need. For example, administration users log in and use a database connection that has been configured for an Administrator. Users who are not members of the Administrator group cannot reuse this connection resource even if it is free in the pool. However, if the restricted connection is not currently in use, iPlanet Application Server can close it and create a new unrestricted connection, thus optimizing use of the limited connection resources. This is called connection stealing.

System administrators can use *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 Tool, 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.

Result Caching

iPlanet Application Server improves application performance by caching the results of application logic execution. By saving the results of a request, the next time the same request is received, the results are immediately on hand. Developers have the option to 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. Caching for Java Server Pages is also available and occurs in the iPlanet Application Server java engine instead of the executive (main) iPlanet Application Server engine.

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 (or 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 monitor server performance and tune the number of available threads to achieve an optimal ratio of threads to system resources.

EJB Session Pooling

Stateless session bean instances can be reused and re-pooled to reduce overhead for each call. As the workhorses of business applications, EJBs require stateless session beans to track the conversation state between the client and iPlanet Application Server. In cases where there are many clients accessing in short durations, each lost session can be reused by the system.

Optimized Communication with Web Servers

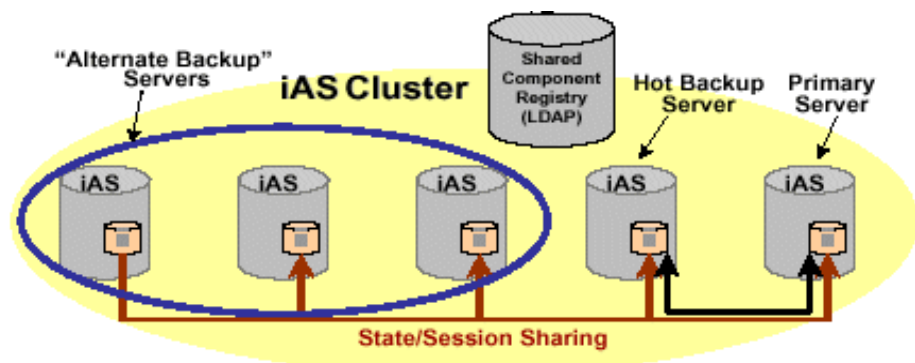
iPlanet Application Server optimizes application performance through tighter integration with web servers. This integration occurs using Web Connector Plug-ins and corresponding Listeners. iPlanet Application Server supports NSAPI, ISAPI, APACHEAPI and optimized CGI for iPlanet, Microsoft, and CGI-compatible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

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 Dynamic Failover (also called failure recovery) and through Dynamic Load Balancing.

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.

Figure 1-3 Dynamic Failover



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. For more information, see *iPlanet Application Server Administrator's Guide*.

Rapid Application Development

iPlanet Application Server enables rapid development of enterprise applications through the following features, which are described in this section:

- Multi-tier Application Model
- Core Services
- Application Development Tools

Multi-tier 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 and based on Java technologies.

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

Table 1-1 Application components and their functions

Functions	Application Components
Functionality in Application Model	Java Applications
Presentation logic	Servlets

Table 1-1 Application components and their functions

Functions	Application Components
Page layout	JavaServer Pages
Business logic	Enterprise JavaBeans
Database access	Enterprise JavaBeans using JDBC; Object to Relational Mapping
Access to other data sources	Enterprise Connectors

These application components fall into two categories:

- Industry-Standard Components
- Other Components

Industry-Standard 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):

- 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.

For more information about using these standard components, see the *iPlanet Application Server Developer's Guide*.

Other Components

Other application components include AppLogics, HTML templates, query files, and extensions. An application must use these components if it is written in C++ or if it will run in the iPlanet Application Server 2.x environment. And if your application must access propriety enterprise data sources, you may need to use prebuilt extensions or create your own.

- An AppLogic is a set of programming instructions, written in C++ or Java, that perform a well-defined, modular task within an application.

- HTML templates are text files that merge HTML with dynamic data to produce formatted output. Templates use special markup tags called GX tags.
- Query files are text files that contain SQL commands, such as for querying or updating a database.
- Extensions enable business logic components to connect to custom or proprietary enterprise resources. Extensions are persistent modules, written in C++ or Java, that are dynamically loaded into iPlanet Application Server and are accessed by multiple AppLogics or EJBs over the life of the extension.
 - Ready-to-use extensions are provided as part of iPlanet Application Server Integration Solutions. iPlanet Application Server Integration Solutions are available for MQSeries, Tuxedo, CICS, IMS, and R/3 applications. These applications can be easily integrated with applications deployed on iPlanet Application Server.
 - Developers can create their own custom extensions using iPlanet Extension Builder, a separately packaged, GUI-based tool that integrates with iPlanet Application Server.

For more information about AppLogics, HTML templates, and query files, see *iPlanet Application Server Migration Guide*. For more information about extensions, consult your iPlanet Extension Builder documentation.

Core Services

iPlanet Application Server provides a set of application services and system services for building, deploying, and managing high-performance, scalable, transactional applications. These services include built-in state and session management, request and transaction management, results caching, connection caching, and so on.

Application Development Tools

The J2EE environment of JSPs, Servlets, and EJBs are fully supported in iPlanet Application Server. iPlanet Application Server supports tight integration with leading integrated development environments.

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).

The Forte for Java product family is the premier development tool set for creating entry-level to enterprise applications in the Java language on any platform. It is a completely open solution, based on the Netbeans™ open source tools platform.

iPlanet also provides a plug-in for the Forte for Java, Internet Edition toolkit that provides developers with an integrated development and deployment environment to take full advantage of the Forte for Java development features for building applications that can be transparently deployed, tested and debugged in the iPlanet Application Server runtime environment both locally and remotely.

Additionally, the plug-in allows WAR packaging of Servlets, JSPs and customizers for iPlanet Application Server descriptors. An integrated Enterprise Java Beans (EJB) module wizard allows developers to easily develop and deploy sophisticated web applications that contain business logic based on EJB technology.

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 Tool to optionally designate a single server to act as the central repository for all state information.

For more information about session and state management, see the *iPlanet Application Server Developer's Guide*.

Security

iPlanet Directory Server provides iPlanet Application Server applications with data security by using access control lists, Secure Sockets Layer (SSL), and password policies.

In addition, iPlanet Application Server provides secure web server communication and supports SSL, HTTPS, and HTTP challenge-response authentication. 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.

User Authentication

iPlanet Application Server Administrator and iPlanet Directory Server provide facilities to enable user authentication to ensure that only authorized users can access applications, databases, and directories.

Server administrators can use the security tool of the Administrator to create users, groups, and roles to manage access to specified resources.

Secure Access to Data Sources

iPlanet Application Server works within the framework of existing role access for relational database management systems. A user or application must log into the database before gaining access to the data.

Developers can write applications so that users enter login information only once and the application saves the information in a session object. Thereafter, the application uses the initial login information to log into different databases, as needed, in the background without requiring additional user input.

iPlanet Application Server shields back-end data by acting as a secure gatekeeper between the web server and the relational database system.

Product Components

This section describes the software and product components of iPlanet Application Server. This section consists of the following topics:

- Programming APIs
- System Services and Application Services
- Sample Applications
- Core Application Server Components
- iPlanet Directory Server

Programming APIs

iPlanet Application Server is compliant with the standards set by the Java 2 Platform Enterprise Edition. iPlanet Application Server supports these industry-standard Java APIs. In particular, iPlanet Application Server supports these APIs and technologies.

- JDK 1.2 Specification
- Java Servlet 2.2 API Specification
- Enterprise JavaBeans 1.1 Specification
- JavaServer Pages 1.0 Specification
- JDBC 2.0 Core API Specification
- JDBC 2.0 Standard Extensions Specification

Note that iPlanet Application Server 6.5 provides full support for JDBC 2.0 as described in *iPlanet Application Server Developer's Guide*.

For more information about the industry-standard Java APIs that are supported in iPlanet Application Server 6.5, see the appropriate API specification. All specifications are accessible from `install_dir/ias/docs/index.htm`, where `install_dir` is the location in which you installed iPlanet Application Server.

System Services and Application Services

System and application services provide a variety of application-level capabilities and system-level capabilities. These services enable the development, deployment, and management of complex business-logic and transaction-based applications.

Sample Applications

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

One sample presents a bookstore application that simulates browsing, searching, and ordering books online. This Java application demonstrates the iPlanet iPlanet Application Server 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 *iPlanet Application Server Developer's Guide*.

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 *iPlanet Application Server Developer's Guide*. For details about the application code, see *iPlanet Application Server Migration Guide*.

Core Application Server Components

The following core components are installed with the iPlanet Application Server:

- Web Connector Plug-In
- iPlanet Application Server Administration Tool
- iPlanet Application Server Deployment Tool

Web Connector Plug-In

The Web Connector plug-in enables communication between iPlanet Application Server and a Web server. When you install iPlanet Application Server, your Web server is automatically configured for the Web Connector plug-in. This means that all necessary directories and settings on the Web server are updated.

If you have problems with the connection between iPlanet Application Server and the Web Connector plug-in, see *iPlanet Application Server Administrator's Guide* for more information.

iPlanet Application Server Administration Tool

The iPlanet Application Server Administration Tool is a stand-alone Java application with a graphical user interface that allows you to administer one or more instances of iPlanet Application Server.

iPlanet Application Server Deployment Tool

The iPlanet Application Server Deployment Tool allows you to package and deploy your J2EE applications. Like the Administration Tool, the Deployment Tool is also a stand-alone Java application with a graphical user interface.

iPlanet Directory Server

Your iPlanet Application Server and other directory-enabled applications use the iPlanet Directory Server as a common, network-accessible location for storing shared data such as user and group identification, server identification, and access control information. The most well known of the Directory Server's services is the Distinguished Name Service (DNS).

The iPlanet Directory Server provides global directory services: it provides information to a wide variety of applications. A global directory service is a single, centralized repository of directory information that any application can access through network-based communication between the applications and the directory. iPlanet Directory Server uses LDAP (Lightweight Directory Access Protocol) to give applications access to its global directory service. The LDAP protocol enables iPlanet Directory Server to scale to millions of entries for a modest investment in hardware and network infrastructure.

NOTE iPlanet Directory Server runs as the `slapd` service on Windows, and `ns-slapd` on Solaris.

iPlanet Directory Server 4.13, installed along with iPlanet Application Server, is configured to store two types of information: configuration information and authentication information. As you install iPlanet Application Server, you set up the Directory Server Data Information Tree (DIT), which has branches for this information. For more information, see the *iPlanet Directory Server Installation Guide* at:

<http://docs.iplanet.com>

The configuration directory is the part of Directory Server used to store the Application Server's configuration information. It contains the data tree, `o=NetscapeRoot`, used by the iPlanet Application Server to store the configuration settings under the suffix that you set up to identify your organization. Multiple server installations can store their configuration settings in this configuration directory.

If you install the Directory Server component with iPlanet Application Server, you must designate this installation of Directory Server as the configuration directory, even if another installation of directory server already exists at your site.

For an overview of the various functions of Directory Server, see *iPlanet Directory Server Installation Guide*.

Deploying Applications

This chapter describes the steps necessary to deploy the Hello DB sample application provided with iPlanet Application Server 6.5.

Topics covered in this chapter:

- Starting PointBase Database Server
- Stopping and Starting the iPlanet Application Server
- Registering the Data Source
- Building the Application
- Viewing the Application
- Undeploying Applications

For more detailed information on building and packaging applications, see the *iPlanet Application Server Developer's Guide*.

Starting PointBase Database Server

PointBase Network 3,5 is a database server provided with iPlanet Application Server to test your applications. If you have not stopped the database server or restarted your system since you installed iPlanet Application Server, you do not need to start the PointBase Database Server. The examples in this guide assume that you are running the PointBase Database Server.

If the PointBase Database Server is not running, you should follow these steps before continuing with the guide:

- On Solaris, go to the *installdir*/pointbase/network/bin and type:

```
pointbaseServer Start
```

- On Windows, go Start Menu > Programs > PointBase Network 3.5 > PointBase Server.

Stopping and Starting the iPlanet Application Server

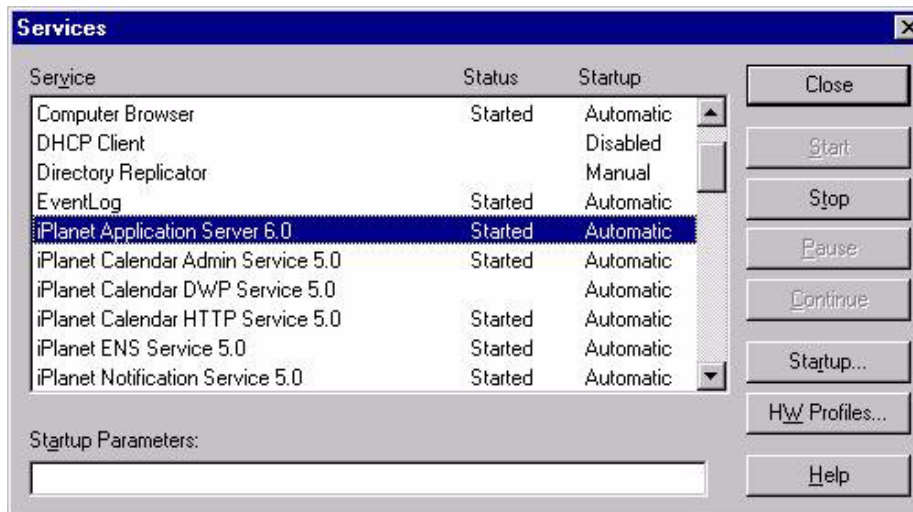
Because the server is automatically started during installation, you must first stop it before you can start it. Directions are provided for both Windows NT and for Solaris.

To Stop and Start the Server on Windows

The iPlanet Application Server is automatically installed as a Windows service. To stop, and start the sever, and to set service properties, follow these steps:

1. From the Start menu, follow this path to open the Control Panel:
Settings > Control Panel.
2. Double click Services. You will see the screen shown in Figure 2- 1.

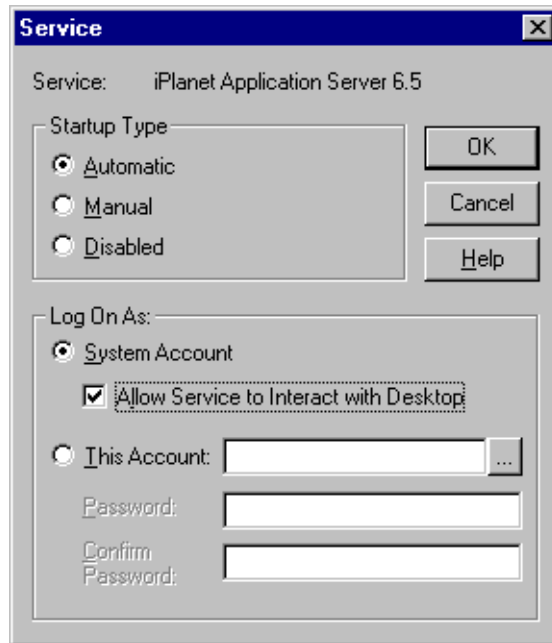
Figure 2-1 The Services Screen



3. Highlight iPlanet Application Server 6.5 and click Startup.

You will see the screen shown in Figure 2-2.

Figure 2-2 The Service Parameters Screen



4. Turn on the Allow Service to Interact with Desktop flag (see arrow 1), and click OK. This returns you to the Services screen (Figure 2 -1).

(Though this step is unrelated to starting and stopping the server, it will be useful later to view the logs while the application is running.)

TIP If you prefer to manually start the server instead of automatically starting it at system startup, you can also select the Manual option in the Startup Type box (see arrow 2 in Figure 2-2).

5. Click on the Stop button.
6. Wait thirty seconds for all of the server processes to stop.

7. Click on the Start button.

Since you are now allowing the service to interact with the desktop, you now have five additional pop-up windows on your desktop. These show the logs of the different components.

TIP	The server process logs can become voluminous. You might find it useful to increase the buffer size and the window size of the log windows. To find these settings, click the icon on the top left of the log window to enable the window menu. From the Properties menu item, select the Layout tab. The ensuing screen allows you to change the buffer and window sizes.
------------	--

To Stop and Start the Server on Solaris

To stop and then restart the iPlanet Application Server on Solaris, follow these steps:

1. Make sure that you are logged in as the root user.
2. Change the current directory to: *ias install directory/ias/bin/*
3. To stop the server, run this command:

```
./iascontrol stop
```

4. Wait thirty seconds for all of the server processes to stop.
5. To restart the server, run this command:

```
./iascontrol start
```

Registering the Data Source

You will now register the database information with the iPlanet Application Server as a data source named `jdbc/hellodb/HelloDB`.

This registration will define the connection parameters to the database your application needs.

To Register the Data Source on Solaris and Windows

1. On Solaris, add *installdir*/ias/bin directory to your PATH environment variable: (the *installdir* directory is where you installed iPlanet Application Server.)

On Windows, open a DOS command prompt by selecting Programs -> Command Prompt from the Start menu.

2. Change the current directory to the hellodb source directory using this command:

Solaris:

```
cd installdir/ias/ias-samples/database/hellodb/src/schema
```

Win NT

```
cd installdir:\ias\ias-samples\database\hellodb\src\schema
```

3. Register the data source with this command:

```
iasdeploy regdatasource hellodb-pointbase-type4.xml
```

The file hellodb-pointbase-type4.xml contains the deployment descriptor for the data source. You can view the file to see that it contains all the information necessary to create a JDBC connection.

Building the Application

Now you are ready to build the application. iPlanet Application Server includes several samples installed in *installdir*/ias/ias-samples. iPlanet Application Server 6.5, includes the Java-based build tool, Ant, which uses an XML-based configuration file analogous to traditional make files. Each sample includes a build.xml with build instructions for its components.

To Build the Application

To build the application, follow these steps:

1. If you are using Solaris, make sure the following directory is part of your PATH environment variable:

```
installdir/ias/bin
```

If you are using Windows NT, this step is automatically completed during the installation.

2. Change the current directory to the hellodb source directory using this command:

Solaris

```
cd installdir/ias/ias-samples/database/hellodb/src/
```

Windows

```
cd installdir\ias\ias-samples\database\hellodb\src
```

3. Build the sample components using this command: build all

This command will compile all the Java source files in the sample, build the WAR and JAR components, and assemble them into an EAR file.

TIP	The WAR, JAR, and EAR files are the standard J2EE packaging components. For detailed information about packaging, see Packaging J2EE Applications for the iPlanet Application Server.
------------	---

4. Deploy the application to your application server using this command:

```
build install
```

This command will register the hellodb application with the iPlanet Application Server using the iasdeploy utility included with the product. This utility uses the application components and deployment descriptors stored in the EAR file to determine how to install hellodb on the application server.

Viewing the Application

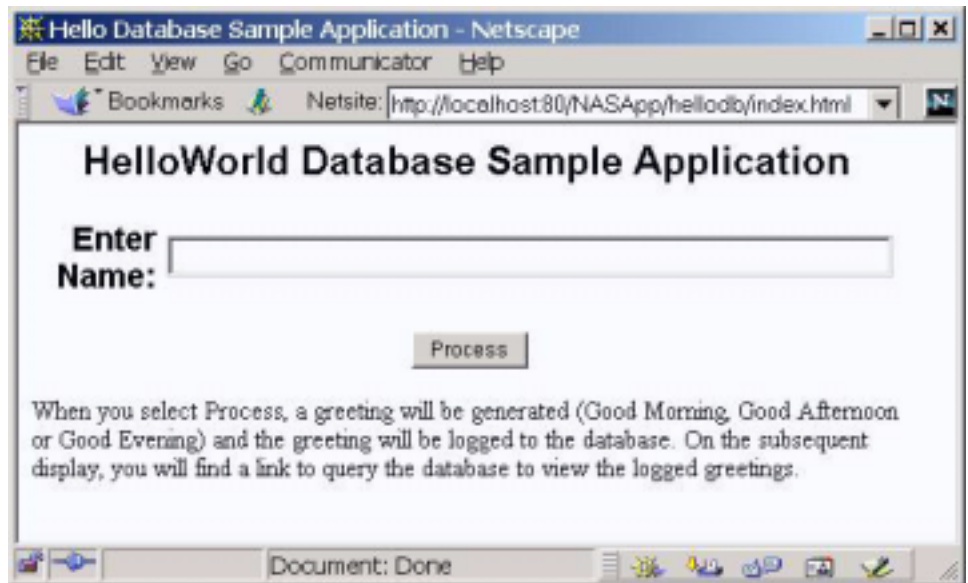
Your hellodb application is now installed and ready to be executed.

Enter the number of the network port used by your web server in place of the [PORT_NUMBER], and enter this URL:

```
http://127.0.0.1:[PORT_NUMBER]/NASApp/hellodb/index.html
```

You will see the screen shown in Figure 2 - 3.

Figure 2-3 HelloDB sample application



Undeploying Applications

Use `iasdeploy` to cleanly undeploy applications.

For example: `iasdeploy removeapp application_ear_file`.

You can also use the J2EE application name as the argument, such as:

```
iasdeploy removeapp j2ee_app_name
```

Some of the other options, include:

1. `removeweb webapp_name`, to remove the web application.
2. `removeejb ejbjar_name`, to remove EJBs.

Use `iasdeploy -help` command to see all available options.

Administration and Deployment Tools

This chapter describes the tools you can use to deploy applications and administer iPlanet Application Server.

iPlanet Application Server eases application management by providing integrated management facilities. These facilities include the following:

- iPlanet Application Server Administration
- Logging Application Messages
- iPlanet Directory Server Administration
- Support for Third-Party Management Tools

iPlanet Application Server Administration

iPlanet Application Server (iPlanet Application Server) Administration Tool is a Java application with a graphical user interface. 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 role based security.

For more information on the iPlanet Application Server Tool, see the *iPlanet Application Server Administration Guide*.

Basic Administrative Tasks

This section describes how to access and use the iASAT tool to start and stop the iPlanet Application Server.

- On Windows go to Start>Programs>iPlanet Application Server 6.5 > iAS Administration Tool
- On Solaris go to, *iasInstallDir/ias/bin* and type: `ksvradmin`

To start the server

- Select the server from the list of registered servers.
- Click Start Server.

Similarly, to stop the server, select the particular server and click Stop Server.

Command Line Tools

iPlanet Application Server comes with various command-line tools and executables, that can be run from the command-line prompt (Windows) and the Shell prompt (Solaris).

Using command-line tools, you can perform a variety of tasks, right from basic configuration to deploying an application.

For a complete list of all the command-line tools that you can use to administer iPlanet Application Server, see *iPlanet Application Server Administrator's Guide*.

To get a complete description of any command-line tool, type the command at the prompt, insert a space and type `-help`. For example, to get a complete list of all the options that you can try with the `iascontrol` command, type `iascontrol-help` at the command line prompt.

Using Command-Line Tools

The usage of iPlanet Application Server's command-line tools is different for Windows and Solaris platforms. Most of the command-line tools have been integrated with the GUI-based iPlanet Application Server Administration Tool and iPlanet Application Server Deployment Tool.

On Solaris, even though the iPlanet Application Server Administration Tool and the iPlanet Application Server Deployment Tools are available, the usage of command-line tools is quite extensive. As you can have multiple instances of iPlanet Application Server on Solaris, it becomes necessary at times to execute command-line tools from the installation directory of a specific instance of iPlanet Application Server, to modify the attributes of only that instance.

On Windows, command-line tools are in the form of executable (.exe) files.

Command-line tools are located in the <iASInstallDir>/ias/bin path, on both Solaris and Windows systems.

To Start and Stop the Server

- On Windows, go to Start>Run and type `iascontrol`
- On Solaris, go to *iASInstallDir*/ias/bin and type: `./iascontrol`

The command to start a server is: `iascontrol start`

The command to stop a server is: `iascontrol stop`

For more information on the various options you can use with `iascontrol`, see the online help (`iascontrol -help`), and the *iPlanet Application Server Administration Guide*.

Deploying Applications

Use the following tools to deploy applications on iPlanet Application Server.

To use GUI based tools

- On Windows, go to Start>Programs> iPlanet Application Server 6.5 >iAS Deployment Tool.
- On Solaris, go to *iASInstallDir*/ias/bin and type: `./deploytool`

To access the command line Deployment Tool

- On Windows, go to Start>Run and type-in `iasdeploy`.
- On Solaris, go to *iASInstallDir*/ias/bin and type: `./iasdeploy`

Logging Application Messages

Message logging is useful for tracking and debugging application errors. By using the `log()` method, you can send messages to the same log destination that the server administrator configures for iPlanet Application Server services.

Logs pertaining to administration and deployment are stored in the Administration Server, known as KAS. The other servers that store message and event logs are KJS (Java Server), KCS (C++ Server) KXS (Executive Server). KAS starts the other three servers and monitors their activities.

For example, if an application encounters a problem in a segment of code, you can log the associated error message. Informational messages about the application's status, rather than error messages, are also useful.

How Log Messages Are Formatted

Every log message has the following four components:

- date and time the message was created
- message type, such as information, warning, or error
- service or application component ID
- message text

When a log message is sent to the text-based destination logs, it is formatted as follows:

[Date and time of message] Message type: Service ID: Message text

For example, the following messages sent to an ASCII text file illustrate message format:

```
[01/18/00 11:11:12:0] info (1): GMS-017: server shutdown (host  
0xc0a801ae, port 10818, group 'iAS') - updated host database
```

```
[01/18/00 11:11:18:2] warning (1): GMS-019: duplicate server (host  
0xc0a8017f, port 10818) recognized, please contact iPlanet  
Communications for additional licenses
```

Determining the Logging Destination

You can configure the logging service to record server and application messages in any or all of the destinations:

- Process consoles
- Application log
- ASCII text file
- Database table

When you enable logging, the logging service automatically sends messages to the process consoles on Windows and Solaris platforms, as long as those consoles are open and console logging is enabled. On Windows, the logging service also sends messages to the application log. Logging to a process console does not record the messages. You cannot retrieve the messages once they scroll off the screen.

For more information on how to enable logging to a particular destination, see the *iPlanet Application Server Administrator's Guide*.

iPlanet Directory Server Administration

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—up to five clusters, in fact. This means that administrative data for all iPlanet Application Server installations can be centralized in one place.

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.

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 network 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.

For more information, see the *iPlanet Application Server Administrator's Guide*.

Troubleshooting iPlanet Application Server

Installation Issues

This section describes the known iPlanet Application Server, Enterprise Edition 6.5 installation issues, and the associated workarounds.

Problem (549653): Installation/upgradation fails on Windows if iPlanet Application Server installables are placed in a directory containing spaces or special characters.

Workaround

The iPlanet Application Server installables should be placed in a directory, which has no spaces or special characters.

Problem (546941): iPlanet Application Server fails to install Directory Server on Windows 2000, after IDS 5.0 has been uninstalled from the machine.

This problem occurs if you had previously uninstalled iPlanet Directory Server 5.0 from the Windows 2000 machine on which you are installing iPlanet Application Server, along with the default directory server.

Workaround

Delete `ns-dshttpd40.dll` from the `winnt\System32` directory before installing iPlanet Application Server.

Problem: Installing iPlanet Application Servers on machines that do not have a static IP address.

Workaround

Install a loopback adapter on the Windows 2000 or NT machine on which you are installing iPlanet Application Server and assign it a static IP address.

Problem: Installation crashes when trying to install on an NFS mounted file system. iPlanet Application Server cannot be installed on a NFS mounted file system.

Core Server Issues

This section describes the known iPlanet Application Server, Enterprise Edition 6.5 core server issues, and the associated workarounds.

Problem (546093): Automatic restart of iPlanet Application Server on the Windows platform may not occur on slower machines.

This is due to a problem with iPlanet Directory Service startup.

Workaround

Add the environment variable, `IAS_KASWAITTIME`, and set it to between 5-15 (seconds) for automatic startup to work in such cases. This variable need not be set for manual startup.

Administration Tool Issues

This section describes the known iPlanet Application Server, Enterprise Edition 6.5 Administration Tool issues, and the associated workarounds.

Problem (544526): Deleting Datasource from Administration Tool does not work

Workaround

Deleting an external or iPlanet Type 2 Datasource cannot be done from the Administration Tool. Delete the datasource entry from the registry using `kregedit`.

For more information, see Chapter 8, 'Administering Database Connectivity' in *iPlanet Application Server Administrator's Guide*.

Problem (540597): Shutdown method in the StartupClass is not invoked when `KIVAes.sh stop` command is used

Workaround

Use the command `iascontrol stop`. We recommend that you do not use `KIVAes.sh` as it has been deprecated.

For more information, see Chapter 1, 'Performing Basic Administrative Tasks' in *iPlanet Application Server Administrator's Guide*.

NOTE	To use the <code>iascontrol</code> command, you must have registered the server with the Administration Tool.
-------------	---

Deployment Issues

This section describes the known iPlanet Application Server, Enterprise Edition 6.5 Deployment issues, and the associated workarounds

Problem: While trying to deploy an EAR file, `iasdeploy` displays 'no servers registered in the registry' error.

Workaround

This message will be displayed if you haven't first registered the iPlanet Application Server using the Administration Tool.

This error could also be caused if there are multiple instances of iPlanet Application Servers or the server is deployed in a remote machine. In such cases you have to mention the instance of iPlanet Application Server you are deploying the application on and its host and port number information.

For more information, see *iPlanet Application Server Administrator's Guide*.

Problem: Either home or remote of the bean Document.class not found".

This error message usually occurs if you have packaged Servlets, JSPs, XML files and Beans in a single EAR file.

Workaround

1. Create a new EJB module and add the EJBs and any classes they depend on to it.
2. Create a Web module and add the Servlets, JSPs, XML files and any classes they depend on to it.
3. Add both these modules to a J2EE application and fix any unresolved links to the EJBs.

Problem (548426): Redeployment of an application/module does not remove the contents of the previously deployed application's/modules's contents. It just overwrites the currently deployed application/modules.

Some of the registry settings are also not overwritten when an application is redeployed.

Workaround

To perform a clean redeployment, first perform a `remove` followed by a `deploy`.

Database Support Issues

This section describes the known iPlanet Application Server, Enterprise Edition 6.5 database support issues, and the associated workarounds.

Problem (481221): RowSet lookups from JNDI is not implemented for Third Party drivers.

Workaround

To use RowSet, construct the RowSet object directly and set the URL to make the underlying connection.

Problem (427149): Text/blob datatypes in prepared statements is not supported for Informix and Sybase native drivers.

Workaround

Use Third Party JDBC drivers.

Additional References

iPlanet Application Server documentation is available online in PDF and HTML format, at: <http://docs.iplanet.com/docs/manuals/ias.html>. Additional resources for the developer community is available online at <http://developer.iplanet.com/>

The following table Where to find more information, lists the tasks and concepts that are described in the iPlanet Application Server manuals and *Release Notes*. If you are trying to accomplish a specific task or learn more about a specific concept, refer to the appropriate manual.

Table 5-1 Where to find more information

For information about	See the following	Shipped with
Late-breaking information about the software and the documentation	<i>Release Notes</i>	The latest version is available online at http://docs.iplanet.com/docs/manuals/ias.html
Installing iPlanet Application Server and its various components (Web Connector plug-in, iPlanet Application Server Administrator), and configuring the sample applications	<i>Installation Guide</i>	iPlanet Application Server 6.5

Table 5-1 Where to find more information

For information about	See the following	Shipped with
<p>Creating iPlanet Application Server 6.5 applications that follow the open Java standards model (Servlets, EJBs, JSPs, and JDBC), by performing the following tasks:</p> <ul style="list-style-type: none">• Creating the presentation and execution layers of an application• Placing discrete pieces of business logic and entities into Enterprise Java Bean (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	<i>Developer's Guide</i>	iPlanet Application Server 6.5
<p>Administering one or more application servers using iPlanet Application Server Administrator Tool to perform the following tasks:</p> <ul style="list-style-type: none">• Monitoring and logging server activity• Implementing security for iPlanet Application Server• Enabling high availability of server resources• Configuring web-connector plugin• Administering database connectivity• Administering transactions• Configuring multiple servers• Administering multiple-server applications• Load balancing servers• Managing distributed data synchronization• Setting up iPlanet Application Server for development	<i>Administrator's Guide</i>	iPlanet Application Server 6.5

Table 5-1 Where to find more information

For information about	See the following	Shipped with
Migrating your applications to the new iPlanet Application Server 6.5 programming model from the Netscape Application Server version 2.1, including a sample migration of an Online Bank application provided with iPlanet Application Server	<i>Migration Guide</i>	iPlanet Application Server 6.5
Using the public classes and interfaces, and their methods in the iPlanet Application Server class library to write Java applications	<i>Server Foundation Class Reference (Java)</i>	iPlanet Application Server 6.5
Using the public classes and interfaces, and their methods in the iPlanet Application Server class library to write C++ applications	<i>Server Foundation Class Reference (C++)</i>	Order separately

iPlanet Application Server Architecture

This appendix describes the following topics:

- Server Processes
- System Components

Server Processes

The architecture of iPlanet Application Server includes four internal servers, which are often called engines or processes. They are responsible for all processing within iPlanet Application Server. The table, iPlanet Application Server servers and their descriptions, summarizes the internal servers:

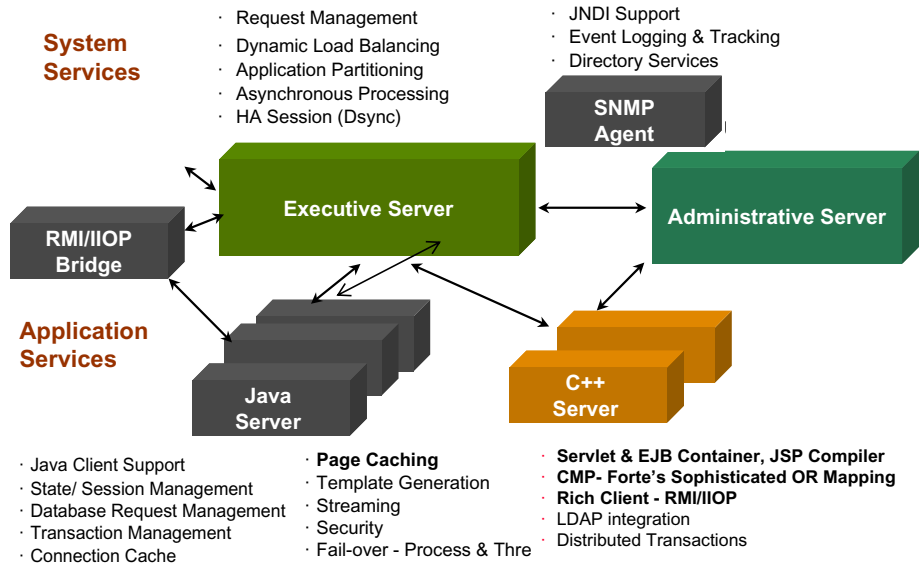
Table A-1 iPlanet Application Server servers and their descriptions

Internal Server	Process Name	Description
Executive Server	KXS	Provides most system services (some services are managed by the Administrative Server).
Administrative Server	KAS	Provides system services for iPlanet Application Server administration and failure recovery.
Java Server	KJS	Provides services to Java applications.

Summary of Process Interactions

The iPlanet Application Server processes figure shows how the four iPlanet Application Server processes interact to handle requests from clients:

Figure A-1 iPlanet Application Server processes



When a web server forwards requests to iPlanet Application Server, the requests are first received by the Executive Server process (KXS). The KXS process forwards the request either to a Java Server process (KJS) or to a C++ Server process (KCS). A KJS process runs Java programming logic, whereas a KCS process runs C++ programming logic.

Each KJS and KCS 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 iPlanet Application Server 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 iPlanet Application Server environment for best performance by:

- Adding any number of iPlanet Application Server machines.
- Specifying any number of KJS and KCS processes.

- Maintaining any number of threads on each process.

In addition to providing high performance, the internal processes make it possible for iPlanet Application Server to remain available 24 hours a day, 7 days a week. If a KJS or KCS process goes down, the Administrative Server process (KAS) will restart it. Additional monitoring in iPlanet Application Server makes sure that the KAS process is always running.

If all iPlanet Application Server processes go down, then other iPlanet Application Server machines in the cluster will take over. (This assumes a multiserver environment.) In addition, iPlanet Application Server can send notifications by email and FAX to alert the system administrator and redirect the system to a different site.

The following sections describe the internal servers in more detail:

- The Executive Server
- The Administrative Server
- The Java Server and C++ Server

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 iPlanet Application Server.

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

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

The Executive Server invokes the request manager, a system-level service.

The request manager assigns a thread to the request and forwards the request to the appropriate application process, which is either the Java Server or the C++ Server.

When the request manager is finished, it is dismissed from the Executive Server, providing a dynamic-usage model for increased server performance.

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 iPlanet Application Server Administrator Tool (iPlanet Application ServerA), 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.

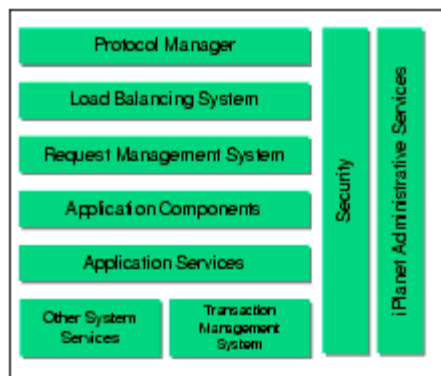
The Java Server and C++ Server

The Java Server and C++ Server processes are the application servers. Business logic components written in Java are hosted in the Java Server, whereas components written in C++ are hosted in the C++ Server. Business logic components are the core of the application, holding the compiled code instructions written by the developer.

The Java Server and C++ Server processes (KJS and KCS) also host 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, iPlanet Application Server 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 iPlanet Application Server architecture. These systems are shown in the iPlanet Application Server architecture figure:

Figure A-2 iPlanet Application Server

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

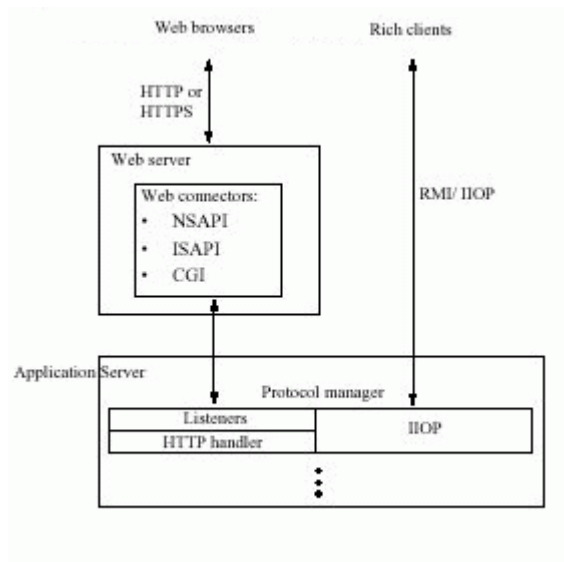
The following is a list of connectors, the protocols used and their functions that are managed by the Protocol Manager to enable communication between iPlanet Application Server and a client:

- Communication between a web server and iPlanet Application Server occurs through NSAPI, ISAPI, and optimized CGI. Optimization and superior performance are achieved through listeners, web connectors, the streaming service, and the Protocol Manager, as described here:

- Web connectors and listeners manage the passing of requests from the web server to the iPlanet Application Server. Listeners distribute and handle requests from the web connectors. New listeners can be added with the HTTP handler.
- The HTTP streaming service handles data streams from the iPlanet Application Server to the web server and to the web browser.
- The iPlanet Application Server Protocol Manager manages and provides services for all active, loaded listeners. The Protocol Manager supports HTTP, HTTPS (HTTP over SSL), and IIOP.

The Protocol Manager figure shows how the Protocol Manager enables communication between iPlanet Application Server and a client (whether web-based or not):

Figure A-3 Protocol manager



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.

The request is then forwarded to the corresponding listener on iPlanet Application Server. Listeners on iPlanet Application Server include NSAPI listeners, ISAPI listeners, and optimized CGI listeners that communicate with iPlanet, Microsoft, and CGI-compatible web connectors, respectively.

Load Balancing System

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. The Load Balancing System includes a Load Monitor and a Load Balancer.

Load Manager

The Load Monitor takes a snapshot of server load based on administrator-configured interval settings. The Load Monitor then tabulates resource availability and system performance.

Load Balancer

The Load Balancer uses the load and performance statistics retrieved by the Load Monitor to determine the server with the most available resources to handle an incoming request. Each instance of iPlanet Application Server has its own load balancing module which makes routing decisions based on load balancing parameters. Administrators can tune these parameters using iPlanet Application Server Administrator.

After a request is routed to the appropriate iPlanet Application Server machine, the request is passed to the Request Management System.

Request Management System

Incoming requests are handled by the Request Management System. iPlanet Application Server 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:

- Thread Manager

The Thread Manager provides a dynamic pool of threads. From this pool, a thread is assigned to process the request.

- Queue Manager

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

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 iPlanet Application Server's Global Directory Service (GDS) and iPlanet Directory Server, an LDAP server.

GDS determines the language of the application component and then loads the appropriate language-specific handler for the request. For example, on Windows NT, a `.class` file is loaded for Java application logic, and a `.dll` file is loaded for C++ application logic. On UNIX, a `.class` file is loaded for Java application logic, and a shared library (a `.so` file) is loaded for C++ application logic.

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.

Typically, request processing involves calling functions from among the application services, transaction management system, or other system services. For example, the application may make use of state and session management or may connect to a database to update a user's account.

For more information about application components, see *iPlanet Application Server Developer's Guide (Java)*.

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 either a KJS process, a KCS process, or both. The following sections summarize the services available to Java and C++ applications:

Services Hosted by Either KJS or KCS

The table, Services available to Java or C++ clients, lists the services that are available to applications written in Java or C++:

Table A-2 Services available to Java or C++ clients

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 iPlanet Application Server 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.
Template management	Manages the merging of result-set data with templates before data is returned to the user.
Database connection caching	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.

Table A-2 Services available to Java or C++ clients

Application Service	Description
Application events	Based on time criteria or other event criteria, allows applications to send and receive emails, to invoke an AppLogic, or (for 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 iPlanet Application Server. Connectors are persistent modules that are dynamically loaded into iPlanet Application Server and are accessed by multiple AppLogics or EJBs over the life of the extension. Although connectors act as application services, connectors can also be considered as application components.

Services Hosted by KJS Only

The table, Service available only to Java applications, lists services that are available only to applications written in Java:

Table A-3 Service available only to Java applications

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 0.92 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.

Table A-3 Service available only to Java applications

Application Service	Description
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 iPlanet Application Server.
LDAP support	Eases management and security by providing a central repository for information about users, groups, and access control lists. LDAP clients that ship with iPlanet Application Server include iPlanet Application Server Administration Tool and iPlanet Directory Server.

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 table, System services. (Note that some of the following services are described elsewhere in this chapter.)

Table A-4 System services

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 (either the Java Server or the C++ Server) and managing request thread allocations.
Global Directory Service	Determines the programming language of an application component so it can be loaded into the appropriate server process, KJS or KCS.
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.

Table A-4 System services

System Service	Description
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 request caching	Caches application requests so that future requests for the same application components by the same user are handled immediately.
Asynchronous processing	Provides processing of multiple requests at the same time when they arrive at varying times.
Failure recovery	Restarts the Executive Server, Java Server, or C++ Server processes if they ever become unavailable.
Distributed data synchronization	Supports failure recovery by synchronizing data. Data is synchronized not only across all KJS or KCS processes running in iPlanet Application Server, but also across all iPlanet Application Server installations within a cluster.
SNMP support	Provides access to iPlanet Application Server via SNMP agents, thereby allowing remote management from third-party administration tools.
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 supports access to back-end databases.

- C++ applications access databases using classes and interfaces provided in the iPlanet Application Server API, also called the Foundation Class Library.
- Java applications access databases using the standard JDBC API.

Local versus Global Transactions

For Java applications, iPlanet Application Server supports both local transactions and global (also called distributed) transactions.

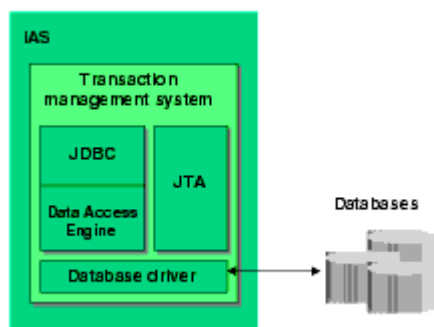
Global transactions span multiple databases (optionally of different types). Global transactions occur using a two-phase commit from Encina, a transaction manager built into iPlanet Application Server. Local transactions involve access to a single database. They provide better application performance because they are less complex.

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

Architectural Details

The Transaction Management System shows the architectural details of the transaction management system:

Figure A-4 Transaction Management System



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.

The transaction management system also includes the Java Transaction API (JTA). JTA is used for 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 by the Encina transaction manager.

For more information about accessing databases, see the *iPlanet Application Server Developer's Guide* and *iPlanet Application Server Administrator's Guide*.

Security

In iPlanet Application Server, security is supported across all subsystems through the use of LDAP and other secure protocols. For more information, see the *iPlanet Application Server Developer's Guide* and *iPlanet Application Server Administrator'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 iPlanet Application Server Administration Tool, iPlanet Directory Server, and third-party SNMP agents.

Index

A

- Administrative Server (KAS) 55, 56
- APIs 27
- application events 62
- application model 22
- Application Partitioning 15
- application partitioning 15
- AppLogics 23

C

- C++ Server (KCS) 54, 56
- caching
 - database connections 61
 - results 19, 61
- configuration directory 30
- cookie management 61

D

- data streaming 20
- Directory Server
 - configuration directory 30
- distributed transactions 63

E

- EJB container 62
- Enterprise JavaBeans 23
- event logging 64
- Executive Server (KXS) 54, 55
- extensions 24

F

- failure recovery service 56
- format
 - URLs, in manual 8

G

- Global Directory Service (GDS) 63

J

- Java Server (KJS) 54, 56
- JavaServer Pages 23
- JSP compiler 62

L

LDAP 63
listeners 58
load balancing 18, 59, 64

M

multi-threading support 20

N

Netscape Application Server Administrator 39
Netscape Extension Builder 24

P

protocol manager 57

Q

query files 24

R

request management 59

S

sample applications 27
security 25
server processes 54
services

 application services 61
 system services 63
servlet container 62
servlets 23
session management 25, 61
state management 25, 61
streaming 20

T

templates 24
transaction management 64

U

URLs
 format, in manual 8

W

web connectors 58