IBM

e-Marketplace Pattern Using WebSphere Commerce Suite, Marketplace Edition

Patterns for e-business Series

Understanding the e-Marketplace Pattern

Guidelines for building an e-Marketplace solution

Implementation examples

Bill Moore
Mitch Fielding
Charles Wilson
Ara Avanesian
Pieter Viljoen

Redbooks

ibm.com/redbooks

SG24-6158-00

International Technical Support Organization

e-Marketplace Pattern Using WebSphere Commerce Suite, Marketplace Edition Patterns for e-business Series

November 2000

Take Note!

Before using this information and the product it supports, be sure to read the general information in Appendix B, “Special notices” on page 467.

First Edition (November 2000)

This edition applies to Version 4.1 of WebSphere Commerce Suite, Marketplace Edition for AIX for use with the AIX operating system.

Note

This book is based on a pre-GA version of a product and may not apply when the product becomes generally available. We recommend that you consult the product documentation or follow-on versions of this redbook for more current information.

Comments may be addressed to:
IBM Corporation, International Technical Support Organization
Dept. HZ8 Building 678
P.O. Box 12195
Research Triangle Park, NC 27709-2195

When you send information to IBM, you grant IBM a non-exclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

© Copyright International Business Machines Corporation 2000. All rights reserved.
Note to U.S Government Users – Documentation related to restricted rights – Use, duplication or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract with IBM Corp.

Contents

© Copyright IBM Corp. 2000

Preface

The Patterns for e-business are a group of proven, reusable assets that can help speed the process of developing applications. The pattern discussed in this book, Business-to-Business e-Marketplace Pattern, is an emerging pattern that allows the development of e-Marketplace hub applications that bring multiple buyers and sellers together in a way that provides efficient electronic trading of goods and services. This pattern is a composition of existing patterns, including the User-to-Online Buying pattern, the User-to-Business pattern and the User-to-User pattern.

This redbook discusses subsets of the application topologies for the Business-to-Business e-Marketplace Pattern. These subsets are used to describe different parts of the full marketplace topology, and they represent increasing levels of complexity, functionality, and integration in the topology, ranging from a simple e-Marketplace to a fully integrated e-Marketplace.

Part 1 of the redbook describes the nature of e-Marketplaces and guides you through the process of choosing an application and runtime topology to deliver the desired market functionality. It then provides you with possible product mappings for implementation of the chosen runtime.

Part 2 of the redbook provides a set of guidelines for building your e-Marketplace application. These guidelines include discussion of performance, technology options, application design, application development, systems management, and security.

Part 3 of the redbook describes, using the standard sample application, the functions available in WebSphere Commerce Suite, Marketplace Edition for AIX. At the time this redbook was written we were using a pre-release version of Marketplace Edition so our working example is not complete. We have taken the approach of first describing our understanding of the facilities that will be available in the released version of WebSphere Commerce Suite, Marketplace Edition for AIX, and then giving some detail of the features we were able to implement and test.

The team that wrote this redbook

This redbook was produced by a team of specialists from around the world working at the International Technical Support Organization, Raleigh Center.

Bill Moore is a WebSphere Specialist at the International Technical Support Organization, Raleigh Center. He writes extensively and teaches IBM classes

© Copyright IBM Corp. 2000

on WebSphere and related topics. Before joining the ITSO, Bill was a Senior Aim Consultant at the IBM Transarc lab in Sydney, Australia. He has 16 years of application development experience on a wide range of computing platforms and using many different coding languages. He holds a Master of Arts degree in English from the University of Waikato, in Hamilton, New Zealand. His current areas of expertise include the VisualAge family of application development tools, object-oriented programming and design, and e-business application development.

Mitch Fielding is an e-Business Specialist at Solution 6, an IBM Business Partner based in Sydney, Australia. He has 10 years' experience in software development and consulting in private and government industry sectors - two years of which have been in the e-Business arena using WebSphere-based technologies. He has previously written on servlet and JSP programming using VisualAge and WebSphere Studio.

Charles Wilson is an e-Business IT/Architect in IBM USA. He has 15 years of experience in software development and consulting in the private sector and with the US government. He has been designing and building Internet and Intranet sites for five years and specializes in e-Business, Java, Linux, and high-performance computing. He holds a degree in Computer Science from Texas Central College.

Ara Avanesian is an Advisory I/T Professional in IBM Canada. He has five years of experience in application development and system design in finance, retail and government industry sectors. As part of IBM Canada e-Business Services and Consulting he specializes in WebSphere-based technologies. He holds a degree in Computer Science from York University.

Pieter Viljoen is a Technical Sales Specialist for the western region in the United States. He has 11 years of experience in technical support, sales, marketing and solutions development in the field. His areas of expertise include technical solutions development, programming and system analysis and design.

Figure 1. The authors

Thanks to the following people for their invaluable contributions to this project:

Margaret Ticknor
John Ganci
Linda Robinson
International Technical Support Organization, Raleigh Center

Vibby Gottemukkala
Richard Goodwin
Scott Smith
Charles Shan

Peter Becker
IBM Dynasty Project

Comments welcome

Your comments are important to us!

We want our Redbooks to be as helpful as possible. Please send us your comments about this or other Redbooks in one of the following ways:

- Fax the evaluation form found in “IBM Redbooks review” on page 487 to the fax number shown on the form.
- Use the online evaluation form found at **ibm.com**/redbooks
- Send your comments in an Internet note to redbook@us.ibm.com

Part 1. Business-to-Business patterns: e-Marketplace topology

© Copyright IBM Corp. 2000

Chapter 1. Introduction to the e-Marketplace Pattern

In this chapter we provide some introductory information about patterns and e-Marketplaces. We first describe patterns and their origins, then we provide a definition for an e-Marketplace as well as some general information about the challenges, success factors, and the key areas that are important when planning an e-Marketplace.

Later on in this chapter we provide a high-level definition of the Patterns for e-business and our proposed Business-to-Business e-Marketplace Pattern. We provide some guidelines on how to use Patterns for e-business and where to get more information.

At the end of this chapter we will briefly cover IBM's Application Framework for e-business and state the outline of this redbook.

1.1 Introduction to Patterns

A pattern is a structured and formal approach for describing a reusable solution to a reoccurring problem. Patterns should be concise and specific but not inflexible. A successful pattern not only is reused in addressing the same reoccurring problem, but is flexible enough to be customized to meet a slightly different but relevant challenge. Patterns are typically a collection of documented best practices and lessons learnt from having to tackle similar problems.

The concept of using well-defined patterns has originated in the building architecture and construction industry. *A Pattern Language: Towns, Buildings, Construction* by Christopher Alexander et al, published in 1977, has been largely credited with introduction of the concept of patterns.

Deployment of pattern type standards has contributed to the rapid advancements in the computer hardware industry. This success and the need for reusability and rapid software development gave rise to object-oriented software, design pattern and component-based development.

The idea of design patterns has gained acceptance by software designers and developers because it enables an efficiency in both the communication and implementation of software design, based upon a common vocabulary and reference.

Buschman et al, the author of *Pattern-Oriented Software Architecture - A System of Patterns,* identified patterns for system architecture at a higher level than the original design patterns. Their patterns are related to the macro-design of system components such as operating systems or network stacks.

Information technology architects, encouraged by the success of design patterns, and facing challenges in systematic and repeatable description of systems, have also explored the idea of architectural patterns.

The Enterprise Solution Structure (ESS) work (see “Enterprise Solutions Structure” in *IBM Systems Journal, Volume 38, No. 1, 1999* at `http://www.research.ibm.com/journal/sj38-1.html )` looked at patterns for complete end-to-end system architectures. ESS is now part of the IBM Global Services methodology.

1.2 Introduction to e-Marketplaces

In the past few years the computer industry has been going through a revolution, one that started slowly but since has accelerated and has forever changed the industry. In its early years the e-business revolution introduced the concept of sharing of information through open media, so that anyone with access to Internet had an enormous amount of information available to them instantaneously. Even though this stage had very little to do with “business”, nonetheless it paved the way for what was to come. The next phase of the e-business revolution was led by companies who seized the power of the Internet to conduct business-to-consumer (B2C) transactions and enabled themselves to drive up their market share through using this alternative channel. The current phase of the e-business revolution promises to surpass the B2C phase in all aspects. This phase is driven by the enterprises who have the vision and the desire to conduct business-to-business (B2B) transactions over the Internet. Among many other benefits, these enterprises look to B2B solutions to improve communications and provide a fast and error-free method of transacting with one another to address their procurement and supply chain processes. e-Marketplaces are the vehicles that provide the desired B2B functionality.

An e-Marketplace is an electronic gathering place that brings multiple buyers and sellers together. An e-Marketplace provides to its members a unified view of sets of goods and services and enables its members to transact using many different mechanisms available in the e-Marketplace.

The e-Marketplace is created and maintained by a “market maker” who brings the suppliers and vendors together. The market maker assumes the responsibility of e-Marketplace administration and performs maintenance tasks to ensure the e-Marketplace is open for business.

Typically there are two distinctive sides to any e-Marketplace: the buy side and the sell side.

- The buy side represents businesses that use the e-Marketplace for their buying needs, such as spot purchasing and/or addressing their enterprise-wide procurement needs.
- The sell side, as the name suggests includes businesses who leverage the e-Marketplace to sell their products via the transaction mechanisms offered in the e-Marketplace.

At the highest level e-Marketplaces can be categorized in two groups:

- *Horizontal e-Marketplaces:* This variation of an e-Marketplace is typically an offering of goods and services at reduced prices. This variation mainly addresses the spot purchasing needs of different types of businesses, and its main value proposition is the delivery of goods and services at reduced prices.
- *Vertical e-Marketplaces*: This variation of an e-Marketplace provides value by efficiently managing interactions between buyers and sellers in a specific industry. The vertical e-Marketplaces are typically very industry specific and deal with a set of specialized goods or services. This variation mainly addresses the supply chain processes of businesses.

In any e-Marketplace there are four main transaction mechanisms. They are:

- *Standard price offerings:* Standard price offerings are predetermined prices for a given set of goods or services. This transaction model is in some ways similar to the typical B2C catalog pricing system. In e-Marketplaces Standard price offerings are further broken down to two sub-models. They are:
  - *Fixed price:* Fixed price offerings as the name suggests are offerings that have a fixed price associated for a particular set of goods or services.
  - *Contract:* Contracts are results of offline negotiations between buying and selling organizations. Contracts typically are entered into the system with special pricing, expiration rules, and termination rules. Contracts can also be generated as result of a Requess For Quote (RFQ) process.
- *Auctions:* Auctions are used in e-Marketplaces to allow rapid inventory turnover, while providing goods and services to the buyers at a reduced

price. Several different styles of auctions, such as *Open cry, Sealed bid* and *Dutch auctions* are available

- *Requests For Quote (RFQs):* Buyers can create RFQs in the e-Marketplace if the product they are interested in does not exist in the e-Marketplace, or if they would like to solicit the selling organizations for a better price for an existing product. Selling organizations view the RFQs in the e-Marketplace and provide their response for the ones they are interested in. The buying organization that initiated the RFQ reviews the responses and selects a possible winner. The winning RFQs can be used to create a fixed price order or to establish a contract. Due to the opposite behavior of RFQs compared to auctions, RFQs are sometimes referred to as reverse auctions.
- *Exchanges:* Exchanges utilize sets of predefined rules to fulfill the buying and selling needs of the e-Marketplace members. Exchanges match *bid offers* with *ask offers* based on the e-Marketplace rules and inform the parties involved of the potential match. The general idea is very similar to the procedures in stock market exchanges. Exchanges are the most complicated mechanism of conducting transactions in an e-Marketplace.

The success of an e-Marketplace depends on many factors, one of which is the ease of performing a transaction. In order to provide an easy and quick method of accessing products and offerings available, the e-Marketplace uses an aggregated catalog. Buyers interact with an aggregated catalog to view the products with real-time pricing, descriptions, and comparisons between different vendors. The general idea is to consolidate products from multiple vendors with all possible existing transaction mechanisms in a single catalog and allow the buyers to be more efficient in purchasing goods and services.

1.2.1 Why e-Marketplaces?

A market maker’s primary purpose is to bring together a highly targeted audience of corporate buyers and sellers to solve specific industry problems imbedded in the trading process. Market makers provide solutions that go beyond the first wave of B2B e-commerce to provide dynamic, open e-Marketplaces that can enter the supply chain of vertical and horizontal industries, introducing new efficient ways of buying and selling.

By simply resolving inefficiencies, market makers act as catalysts to compress process time, decrease costs, and improve business processes, in ways previously unimaginable. By adding value beyond efficiency, market makers can truly revolutionize the way trading partners do business.

Providing a comprehensive range of services surrounding trade, market makers offer a strong value proposition, improve customer retention and expand their own potential revenue. Some of the value market makers may provide to an e-Marketplace include:

- Industry expertise and content
- Catalog aggregation
- Transaction negotiation and facilitation
- Logistics
- Community services (chat, discussion)
- Procurement workflow
- Financial settlement or financing
- Quality assurance and rating services
- Business intelligence
- Customer service

1.2.2 Challenges and inhibitors

The competition in e-Marketplaces is increasingly fierce as market makers attempt to gain dominance and critical mass. The biggest obstacle for new e-Marketplaces is the ability to bring in enough buyers and sellers to achieve liquidity and create a truly effective dynamic e-Marketplace. The top challenges facing market makers today include:

- Attracting critical mass
- Creating brand recognition
- Building a sound infrastructure
- Getting to market quickly
- Keeping costs down
- Integrating business processes
- Keeping up with rapidly changing services
- Surviving competition and the network effect
- Achieving long-term viability

1.2.3 Critical success factors

Market makers are facing some daunting challenges. These challenges translate to a set of critical objectives they must meet in order to succeed.

Some of the primary keys to gaining critical mass and generating revenue include:

- Time to market and liquidity

  The market will reward first movers in the race for critical mass of buyers and sellers. Because there are only so many buyers and sellers in a particular industry, markets that already have members tend to attract more members. A key measure of success of an e-Marketplace is the value of the transactions flowing through it.

- Low cost of ownership and operations

  To be profitable yet still offer low-cost, full-featured solutions, the e-Marketplace should be easy to operate and maintain. Ideally, the underlying infrastructure will help accomplish this through tools, automation and built-in functionality.

- Cutting-edge functionality

  A successful e-Marketplace will have to go beyond simply aggregating content and facilitating transactions. These functions are necessary but not sufficient to realize the true potential of e-Marketplaces. Market makers need innovative capabilities to become and remain competitive.

- Flexibility

  Market makers will have to tune e-Marketplaces to deliver the value proposition that is appropriate for the unique dynamics of the industry they serve. As requirements evolve and opportunities arise, market makers should be able to adapt the e-Marketplace accordingly.

- Ability to add value

  The market is moving quickly, and e-Marketplace services are rapidly becoming commodities. What is considered valuable today may be merely meeting the minimum tomorrow. An effective e-Marketplace platform will enable market makers to quickly design and develop new value-added services.

- Secure, scalable, reliable solutions

  All businesses engaging in e-commerce need secure, scalable, reliable solutions, but these traits are even more crucial for an e-Marketplace facilitating trade for entire industries. An e-Marketplace platform should possess a mission-critical, bulletproof infrastructure.

1.2.4 e-Marketplace solutions: designing for success

First-moving market makers who focus on market capitalization over long-term viability and the creation of value will risk failure. Success depends on the market-maker's ability to attract and retain multiple participants—buyers, sellers and supply chain partners. The requirements and critical success factors of these participants should drive the design and implementation of an e-Marketplace. Poorly designed or executed e-Marketplaces will attract few participants. In a vicious cycle, e-Marketplaces with fewer members provide less value and therefore fail to attract and retain new members. A sound, forward-thinking market design will gain critical mass and motivate others to participate.

1.2.5 Critical success factors of e-Marketplace participants

Besides gaining access to a critical mass of buyers and a multitude of sellers, some of the key factors for attracting e-Marketplace participants and maintaining their participation include:

- Low cost of operations and ease of participant setup and enablement
- Role-based access control and approval flows
- Catalog content management
- Aggregated catalog, with organization-unique views
- Searchable, filterable content
- Variety of trading mechanisms (for example, auction, reverse auction, contracts)
- Support for buying and selling at desired terms and conditions
- Tracking and reporting of buying and selling activity

1.2.6 Characteristics of an effective e-Marketplace platform

Most early-adopting market makers built their own e-Marketplace infrastructures. They found that the e-Marketplace solutions of most third-party vendors lacked functionality or could not be configured to meet industry needs. Now, time-to-market pressures, along with the breadth and depth of function required for an e-Marketplace to be successful, make proprietary solutions ineffective and unwieldy. Market makers should choose an e-Marketplace platform carefully to avoid getting locked into limited and limiting technology, which will be costly to replace later.

Most packaged e-Marketplace software focuses only on one or two key aspects of B2B trade, such as community, procurement, or auctions. e-Marketplaces built on software platforms that support the capabilities ofonly one or two of these types of solutions cannot deliver the total value proposition of a trading hub. A true e-Marketplace must provide comprehensive trading capability comparable to that of all of the above examples combined. The strength of the design of an e-Marketplace, as well as the platform underlying it, will be a significant factor of its success. e-Marketplaces built on the most innovative and robust platforms will be the ultimate winners. Some critical characteristics of an effective e-Marketplace platform include:

- Quick to market

  One of the market-maker's most critical success factors is speed to market. An e-Marketplace solution should be quick and easy to implement, providing robust out-of-the-box functionality and pre configured options.

- Highly flexible

  In addition to being "pre baked" enough to get to market quickly, a platform must also be highly customizable and configurable. Different industries have unique market requirements; market makers have preferred business models; and e-Marketplaces will be adapted and extended over time to create additional value.

- Unified content

  e-Marketplaces should provide a single point of access to heterogeneous data sources, transparently aggregating catalog content into a single, unified view. They should also provide mechanisms for effectively searching, filtering and mining content.

- Dynamic trade mechanisms

  Businesses have a variety of wants and needs for negotiating trade. They look for a single e-Marketplace that can fulfill all of these requirements.

- First-wave B-to-B solutions supported only fixed pricing

  True e-Marketplace solutions must support dynamic pricing and negotiation through various real-time mechanisms, such as contracts, auction, requests for quote and exchange.

- Open, standards-based architecture

  To avoid locking customers into limiting, proprietary solutions, a platform should be based on industry-accepted standards. This will better enable market makers to extend the system, and achieve integration and interoperation with other systems.

- Ease of integration

  To provide a truly end-to-end solution, market makers must be prepared to enable the integration of the diverse data, applications and processes of their members. Market makers unable to manage the back-end integration among buyers and sellers will fall short of their promises of huge efficiencies.

- Built for success

  A market-making solution should presuppose the success of the e-Marketplace. Gaining critical mass will require scalability; achieving high liquidity will require robust, high-performance transaction capability; rapidly increasing scope will require advanced functionality and adaptability.

1.2.7 The future of e-Marketplaces

Industry analysts and market makers expect explosive growth, migration to next generation solutions, and a major shift from transaction fees to value-added services.

The focus of e-Marketplaces will move from gaining critical mass to deploying increasingly complex functionality to attract members and drive revenue.

The winners will be second-generation e-Marketplace industry leaders whose sites will integrate multiple transaction mechanisms and a range of value-added services.

New challenges facing market makers include:

- Increased competition
- Service offerings are rapidly beoming commodities
- Lack of virgin markets
- Technical sophistication of competitive e-Marketplaces
- Market consolidation: finite trading means only two-to-three e-Marketplaces in each industry

To address the trends and meet these challenges, an e-Marketplace platform must embody the following characteristics:

- More robust out-of-the-box functionality

  Solutions should support such features as: dynamic and flexible contract capability, collaborative procurement to refine projections, automated replenishment for better inventory management and real-time progress tracking along the entire supply chain.

- Pervasiveness

An e-Marketplace should provide access to data, applications and people—anytime, anywhere—through a variety of devices, such as phone, pager, fax and e-mail.

- Decision support

  As trading transcends price, decisions are based on real-time analysis of criteria, such as availability, delivery time, quality, financing, and insurance. Decisions are also driven by data, forecasts, and analyses housed in ERP and supply chain systems, as well as in the e-Marketplace itself. That intelligence is the true value of aggregation.

- Ease of integration

  e-Marketplaces will continue to find new ways to help buyers and sellers cut costs, tying them even more closely to the business processes they support. Businesses and business systems will need to communicate and collaborate with each other more directly and more efficiently for an e-Marketplace to be truly dynamic and efficient. e-Marketplace platforms that cannot facilitate this cost effectively will not meet the needs of market makers.

- More flexibility

  The market model and design must be easily changeable to react to new opportunities and requirements before competitors do.

- Interoperability

  Customer requirements and expectations may outpace the built-in function of a single e-Marketplace; and horizontal e-Marketplaces may be able to provide value to vertical ones. For these reasons, e-Marketplaces should be able to connect and communicate with one another, creating true networked markets.

1.2.8 WebSphere Commerce Suite, Marketplace Edition for AIX

In Part 3, “Business-to-Business Patterns: e-Marketplace example” on page 193 of this redbook we will build and analyze examples of e-Marketplaces using WebSphere Commerce Suite, Marketplace Edition for AIX. The goal of this product is to meet the needs and critical success factors of Net market makers. WebSphere Commerce Suite, Marketplace Edition for AIX is a comprehensive e-Marketplace platform based on distributed architecture and patented, best-of-breed IBM technologies and components. It provides the infrastructure and tools that market makers need to establish high-performance, full function, scalable e-Marketplaces on the Internet, efficiently and effectively.

Unlike most e-commerce solutions that focus on either buy-side or sell-side requirements, and deliver only pieces and parts of an e-Marketplace strategy, WebSphere Commerce Suite, Marketplace Edition for AIX is a complete, end-to-end solution that addresses the entire value chain. And meets the diverse needs of buyers and sellers as well as the needs of the market makers who facilitate them.

The comprehensive functional scope and out-of-the-box capability of this product greatly reduces the time and technical effort required to build and launch a custom e-Marketplace. Adherence to open standards, a modular architecture and a high degree of configurability and customizability also ensure fast time to market for new features and value-added services to meet the dynamic needs of growing e-Marketplaces. The solution combines ease and speed of development with adaptability and versatility.

- The five Cs

  WebSphere Commerce Suite, Marketplace Edition for AIX was designed to support the five critical aspects of B2B e-Marketplaces: the content the e-Marketplace provides, the commerce engine that powers the e-Marketplace, the coordination of trade activity, the community it fosters and the connectivity that facilitates a seamless web of information and transactions. A market-maker should consider including the right mix of the five Cs, which may vary according to the industry the e-Marketplace serves. WebSphere Commerce Suite, Marketplace Edition for AIX provides key capabilities in each of these five areas:

  - Content.

    Aggregated, normalized and standardized catalog information; search and filter of content; member profiles; reports of trade activity.

  - Commerce.

    Dynamic pricing, transaction, payment, global trade.

  - Coordination.

    Approval flow, negotiation: exchange, auction, reverse auction (RFP/RFQ), dynamic contracts, order tracking.

  - Community.

    Chat, discussion, shared workspace, e-mail.

  - Connectivity.

    Integration with back-end systems, trading partner systems, other e-Marketplaces.

1.3 Introduction to Business-to-Business e-Marketplace Pattern

The Business-to-Business e-Marketplace Pattern is an emerging pattern that is a subset of Patterns for e-business, so it is appropriate to first look at the Patterns for e-business before discussing the Business-to-Business e-Marketplace Pattern.

The Patterns for e-business aim to communicate in a highly accessible fashion the business pattern, systems architecture (application and runtime topologies), product mappings, and guidelines required for different classes of applications. The following figure depicts these aspects.

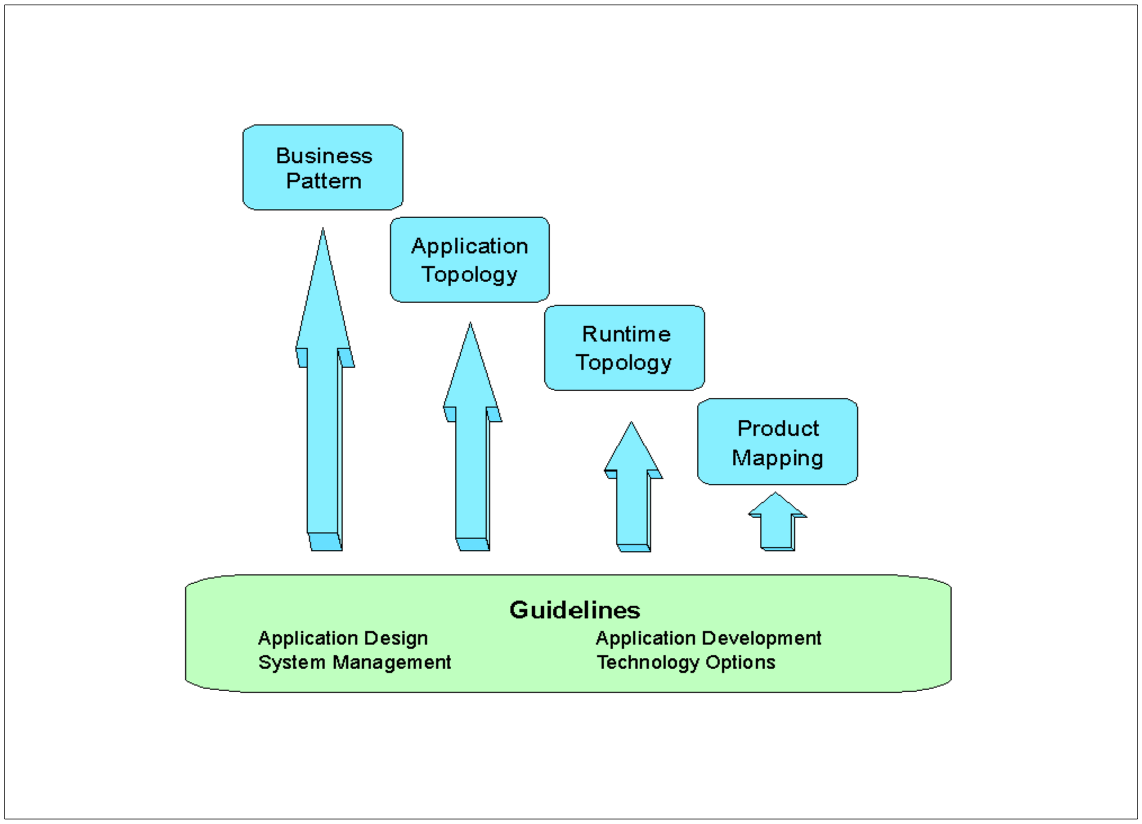


Figure 2. Patterns for e-business

The patterns are cataloged according to the following business context scheme:

- *User-to-Business pattern* describes user interactions, internal or external to the enterprise, with the enterprise transactions or data. This pattern covers all non-commerce interactions between users and businesses.
- *User-to-Online Buying pattern* addresses the commerce transactions of goods and services between the user and a business through tools such as catalogs shopping carts, electronic wallet etc.
- *Business-to-Business pattern* addresses the interaction of parties who do not belong to the same company. This pattern is sub-categorized into:
    - *Business-to-Business Integration pattern* addresses the interaction of business processes between organizations.
    - *Business-to-Business e-Marketplace Pattern* describes the interaction between multiple buyers and sellers.
- *User-to-Data pattern* describes the approach to using tools to extract useful information from large volumes of data.
- *User-to-User* addresses the interaction between users via use of email, shared documents etc.
- *Application Integration* describes the linking of applications within a business.

Based on the definition of an e-Marketplace, provided in the previous section, and the definitions of the Patterns for e-business, it becomes apparent that the Business-to-Business e-Marketplace Pattern is a pattern composed of building blocks from several Patterns for e-business. The following diagram depicts the Business-to-Business e-Marketplace Pattern using Patterns for e-business as building blocks.

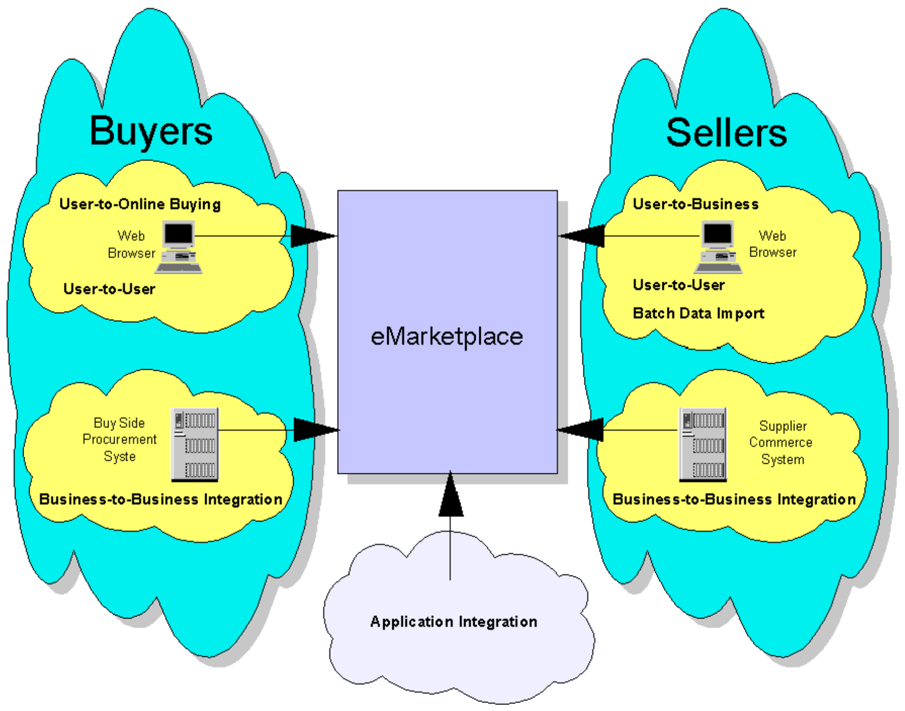


Figure 3. Business-to-Business e-Marketplace Pattern

The User-to-Online Buying pattern facilitates the interaction between the buyer and the e-Marketplace. Activities such as purchasing from an aggregated catalog, participating in auctions or exchanges are performed using this pattern. This pattern is also used for creating Requests For Quotation (RFQs) by the buyers who are looking for either one-time purchases or trying to establish long-erm contracts with suppliers.

The User-to-Business pattern facilitates the non-commerce sell side functions such as updating the catalog, checking orders, checking RFQs, and accessing orders.

The User-to-User pattern facilitates the workflow approval process of an e-Marketplace.

Batch-Data-Import does not correspond to one of the high-level business patterns. It is a batch data exchange that facilitates bulk transfer of information, as opposed to an automated data exchange.

The Business-to-Business Integration pattern is used on both the buy and sell side of the e-Marketplace pattern. On the buy side, the pattern defines the interaction between the buyer's procurement system and commerce functions of the e-Marketplace. On the sell side, this pattern defines the interaction between the supplier's commerce system and the commerce functions of the e-Marketplace.

The Application Integration pattern facilitates the integration of supporting systems and the business processes of the e-Marketplace owner and its internal network.

1.4 How to use patterns

The Patterns for e-business are particularly focused upon addressing common business application problems and providing answers to frequent architecture, design, and implementation questions.

You can use the Patterns for e-business in a number of ways according to your needs:

- As a starting point for an end-to-end system architecture.
- As a detailed example and prescriptive approach, following the product mappings and guidance provided.
- As a way to design more complex, multi-channel systems, when several patterns are used together.

As with the design patterns and ESS work, we anticipate that architects and designers will want to combine these patterns to compose solutions to more complex system architectures. As the other Patterns for e-business are published, we will identify the appropriate integration points for such composition.

We recommend that you use the Patterns for e-business together with an appropriate development methodology that considers the full set of requirements that are to be understood and implemented, whether these requirements concern the function of the solution or its operational characteristics such as availability, scalability, or performance. For more information on application design see Chapter 7, “Application design guidelines” on page 89.

1.5 Patterns Web site

The Patterns for e-business are published on the IBM Developer Works Web site, and can be located at `http://www.ibm.com/software/developer/web/patterns`.. This interactive patterns site acts as a guide to aid you in the selection of the pattern and topologies most relevant to your needs. While you can navigate via shortcuts to the information you most need, the site is structured to enable you to “drill down” into the material as you:

1. Select a business pattern.
2. Select an application topology.
3. Review runtime topologies.
4. Review product mappings.
5. Review guidelines.

At the time of writing this redbook, the Web site has material for the User-to-Business, User-to-Online Buying and Business-to-Business Integration patterns, with material for the other Patterns for e-business in the process of being developed.

1.6 Patterns and Application Framework for e-business

The advent of e-business, with the requirement for interoperability that it brings, has been a major catalyst for the more rapid adoption of standards by the industry.

IBM’s Application Framework for e-business establishes:

- A recommended approach for building systems, embodied in the Patterns for e-business.
- Innovative technology delivered in a rich product portfolio.
- Cross-platform standards, including Java and XML.

The Framework, with the standards it proscribes for e-business systems and their components, can be applied to:

- Custom application code
- Application packages
- Software products

The Patterns for e-business are an integral part of the IBM Application Framework for e-business. The Patterns make it easy to apply the technologies, standards, and products of the Application Framework to provide an e-business solution.

Figure 4 shows a pictorial summary of the major technology standards included in the Application Framework, with an indication of the areas where they are important.

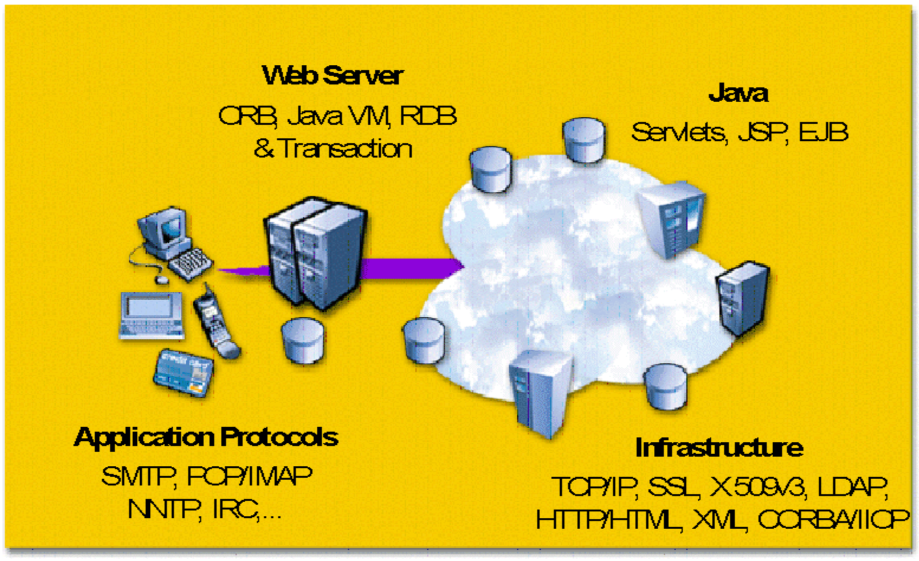


Figure 4. Technology standards and the Application Framework for e-business

Framework white papers are an important source of information for the guidance material included in the Patterns for e-business. The Application Framework site at: `http://www.ibm.com/software/ebusiness` includes a library section with this series of white papers.

1.7 Structure of this book

Chapter 2, “e-Marketplace application topology” on page 21, introduces application topology for the Business-to-Business e-Marketplace Pattern. With very accessible notation, the application topology captures the essential “shape” of the application solution.

Chapter 3, “e-Marketplace runtime topology” on page 31, discusses the runtime topologies for each application topology.

Chapter 4, “e-Marketplace product mapping” on page 45, provides sample product mappings to populate the logical runtime topology. The focus remains on the operational aspects of the solution.

Chapter 5, “Performance guidelines” on page 51, introduces performance guidelines by considering the components of a Business-to-Business e-Marketplace Pattern solution that are particularly relevant to performance.

Chapter 6, “Technology options” on page 69, discusses the technology options available to implement a Web application and provides advice on the appropriate usage.

Chapter 7, “Application design guidelines” on page 89, introduces consideration of the functional components of the application within the context of the runtime topologies.

Chapter 8, “Application development guidelines” on page 123, provides guidelines for application development, considering the roles, processes and tools that are required.

Chapter 9, “System management guidelines” on page 133, looks at the asset management, security, and availability aspects of an e-business application.

Part 3, “Business-to-Business Patterns: e-Marketplace example” on page 193 provides details of an example e-Marketplace application implemented using WebSphere Commerce Suite, Marketplace Edition.

Chapter 2. e-Marketplace application topology

An e-Marketplace is a hub that brings multiple buyers together with multiple sellers. Marketplaces provide value to their members by providing a unified view of the set of goods and services traded in the market and by providing a variety of mechanisms to facilitate trade in such products.

Implementing a successful e-Marketplace requires that you identify an appropriate application topology so that all the requirements of buyers and sellers can be properly met.

An application topology is made up of a set of logical nodes that describe how users, applications and data work together. An application topology is a high-level view of the principal layout of the application; it does not show middleware, files and databases, nor does it describe application design.

An emerging e-business pattern such as the Business-to-Business e-Marketplace Pattern, can be implemented using different application topologies. In this chapter we describe some possible topologies and the considerations for using them. The aim is to help you choose the topology that best fits the requirements for users, applications, and data.

2.1 The Business-to-Business e-Marketplace Pattern

Just as an e-Marketplace is built up from e-commerce, user service and application integration building blocks, the Business-to-Business e-Marketplace Pattern is composed of a combination of business patterns corresponding to these building blocks. Figure 5 on page 22 relates the principal functional areas of an e-Marketplace with the patterns that will be combined in this mixed pattern.

© Copyright IBM Corp. 2000

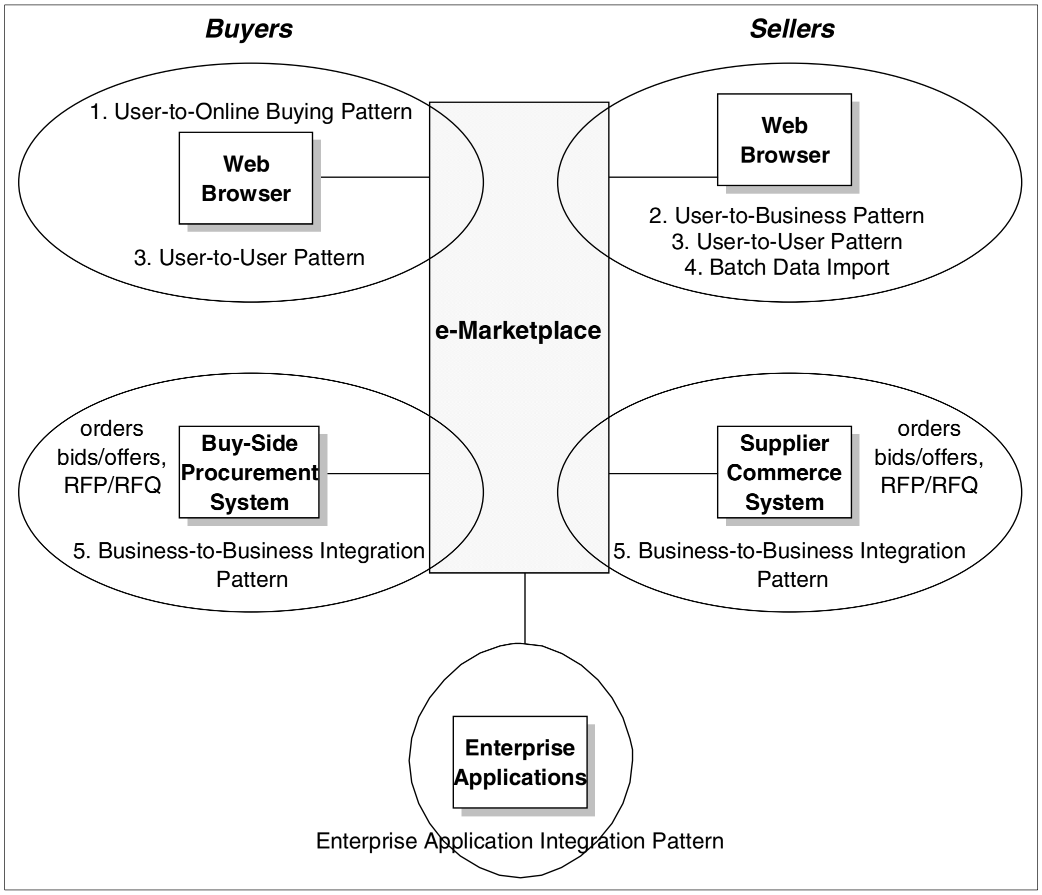

Buyers
Sellers
1. User-to-Online Buying Pattern
Web Browser
3. User-to-User Pattern
Web Browser
2. User-to-Business Pattern
3. User-to-User Pattern
4. Batch Data Import
e-Marketplace
orders
bids/offers,
RFP/RFQ
Buy-Side Procurement System
5. Business-to-Business Integration Pattern
Supplier Commerce System
orders
bids/offers,
RFP/RFQ
5. Business-to-Business Integration Pattern
Enterprise Applications
Enterprise Application Integration Pattern

Figure 5. e-Marketplace major functions

Application topologies from the business patterns 1 through 3, and 5 in this diagram represent sub-topologies of the composite e-Marketplace application topology. Number 4 is an application topology introduced in this pattern. It is these specific application topologies that are combined to form the e-Marketplace application topology. All of these topologies are described using the diagram conventions shown in Figure 6 on page 23.

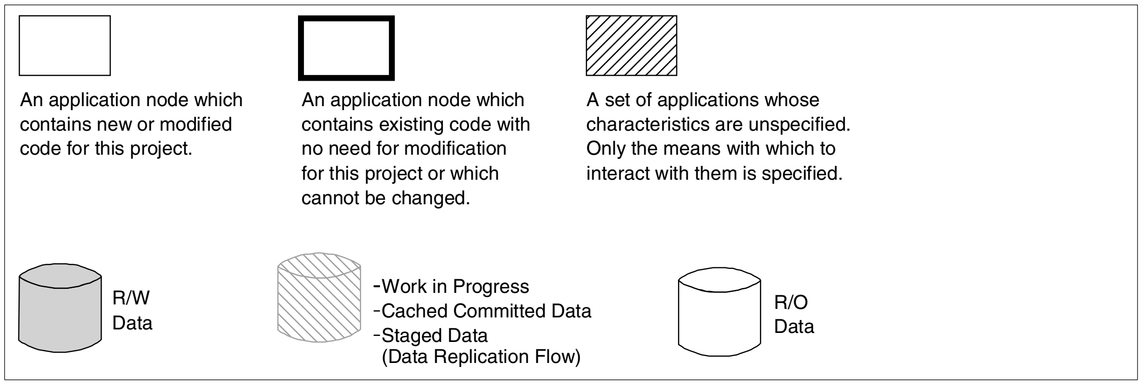

An application node which contains new or modified code for this project.
An application node which contains existing code with no need for modification for this project or which cannot be changed.
A set of applications whose characteristics are unspecified. Only the means with which to interact with them is specified.
R/W Data
-Work in Progress
-Cached Committed Data
-Staged Data
(Data Replication Flow)
R/O Data

Figure 6. Diagram conventions for application topologies

Sub-topology 1 is a standard User-to-Online Buying pattern. It embodies the commerce interaction done by a purchaser in the e-Marketplace. A marketplace contains interactions, such as RPQ/RFP and exchange trading (which are addressed in supporting sections of this business pattern), that go beyond what is typically encountered in a customer-to-business online buying scenario. App 1 in Figure 7 on page 24 handles the matching and selling functions of the marketplace.

Sub-topology 2 is a User-to-Business pattern that addresses non-purchasing interactions with the marketplace, such as a supplier checking order statistics and providing catalog updates. App 2 in Figure 7 on page 24 is a content creation application or an application to provide supporting services, such as performing RFP/RFQ processing or accessing purchase orders.

Sub-topology 3 is a User-to-User pattern. Such a pattern comes into play in approval workflow, in which an approver must sign off on a purchase before it is submitted to the supplier. App 3 as shown in Figure 7 on page 24 is a workflow application for implementing such flows.

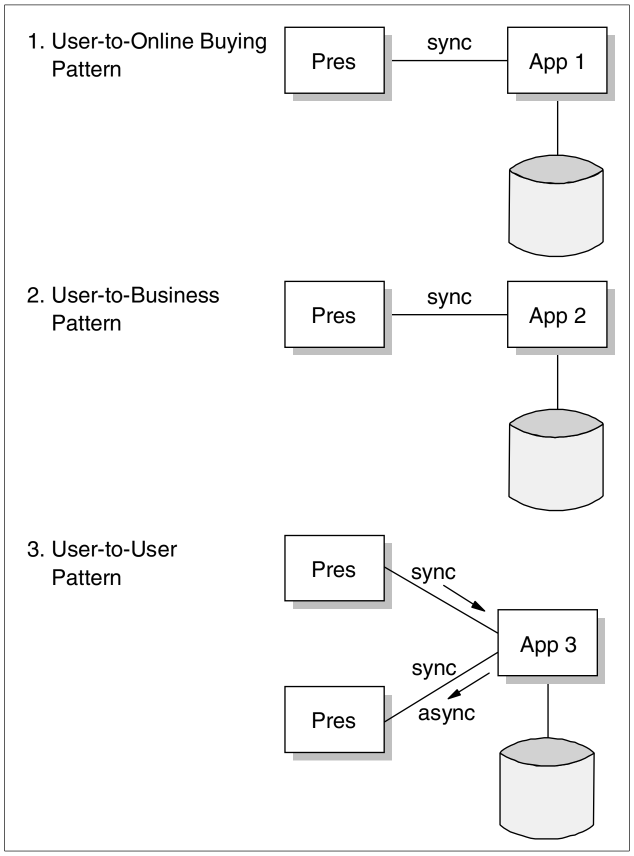


Figure 7. e-Marketplace applications

The next two sub-topologies specify the program-to-program interactions between a supplier and the marketplace or an automated purchaser and the marketplace. Sub-topology 4 does not correspond to one of the high-level business patterns. It is a batch data exchange, such as the programmatic import of potentially sophisticated catalog information into the aggregate catalog of the e-Marketplace. This is a bulk transfer of information, as opposed to an automated data exchange following a managed Business-to-Business protocol, which falls under the Business-to-Business Integration class of integration (sub-topology 5). "App 4" in this topology is an application in the marketplace, that provides aggregation and publishing of the catalog. It is not necessarily the same as the application that manages the catalog. "App 6" is a catalog content application provided by a supplier. It provides content creation and extraction, potentially driven by business rules, and the transmission of this content to the marketplace.

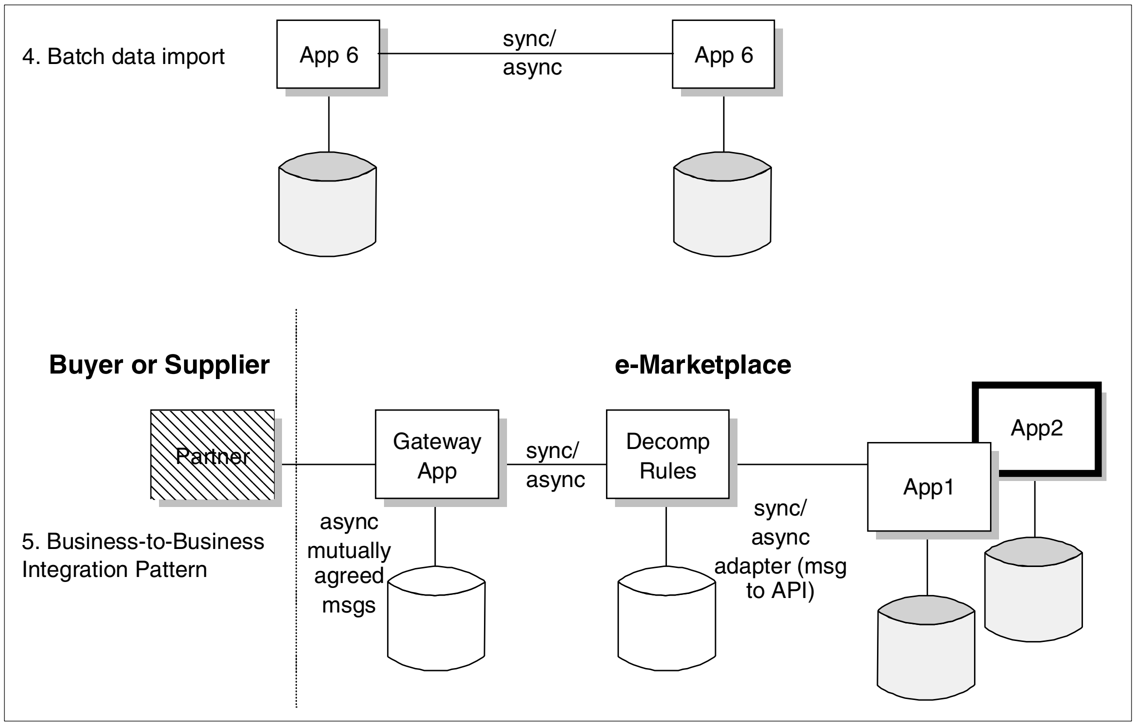


Figure 8. e-Marketplace applications

Sub-topology 5 is the Internet/nanaged topology from the Business-to-Business Integration business pattern. It defines the interaction between the e-Marketplace and the supplier's commerce system, as well as that between a buyer's procurement system and the commerce functions of the e-Marketplace where this interaction is governed by a well defined and executable contract. The App 1 in this topology is the App 1 in sub-topology 1 above. When the commerce functions of the Marketplace integrate with automated buyer or supplier systems, this pattern is applied. Other applications in the marketplace may also need to interact with partner applications. Such an application is App 5.

In addition, the following patterns, the first two of which are currently under development, are related to this application topology and will be referenced:

- Enterprise Application Integration - for the integration of supporting systems and business processes within the market maker's internal network.

- User-to-User - for sophisticated community support beyond the approval flow used in a purchasing transaction. Community support includes user collaboration facilities, such as discussion forums, that augment the core commerce functions of the marketplace.
- User-to-Data - for support of business intelligence by the marketplace and suppliers.

2.2 Subsets of the e-Marketplace topology

An e-Marketplace involves many types of interactions with buyers and sellers. Some of these can be user-driven and others can be carried out programmatically. The degree of programmatic integration of buyers and sellers into the e-Marketplace is a feature that can be used to classify subsets of the full topology defined above. These subsets can represent a phased approach to implementing the e-Marketplace or as means to address specific requirements for B2B commerce.

2.3 Subset 1: Web Integrated e-Marketplace

In this first class of e-Marketplaces, we have an implementation in which all buyer and seller tasks are performed by using a Web browser to log onto the marketplace site and interactively conduct business. The activities supported by such an implementation would include:

- Registration of organizations and users for both buyers and suppliers
- Catalog maintenance
- Product selection by buyers
- Negotiation without automated inventory checking
- Creating and submitting purchase orders
- Notification of sellers of activity via e-mail
- Sellers obtaining purchase orders via browser sessions
- Sellers updating the status of an order in the marketplace
- Buyer reviews of status of purchase orders (could include confirmations, building or shipping schedules, delivery schedules and package identifications)
- All of the above may include approval processes
- Excludes payment processes

Such commerce capability corresponds to an initial or first phase implementation of an e-Marketplace. It is a good basis upon which to add automated integration with buyers and sellers. Referring to our initial picture of an e-Marketplace, Subset 1 would look like Figure 9.

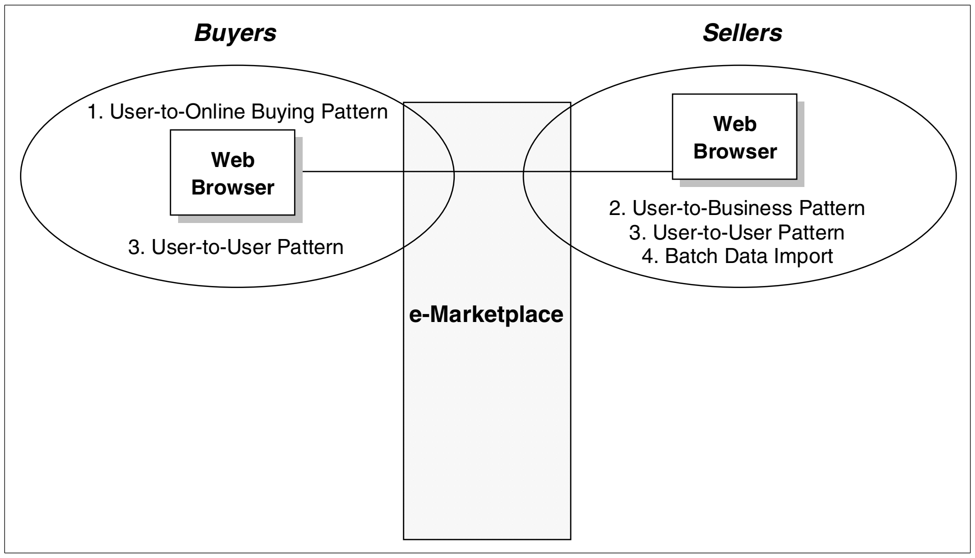


Figure 9. Application Topology - Subset 1

It is built from sub topologies 1 through 3 and is fundamentally built around the User-to-Online Buying pattern.

2.4 Subset 2: e-Marketplace With automated supplier integration

The second class builds on the first by adding programmatic integration with suppliers. This is a business to business integration capability added to the Web-based features of Subset 1. Essentially speeding the supplier processes, it augments Subset 1 with the following:

- Automatic response to RFQ (in cases where this is possible)
- Real time inventory check/availability to promise
- Automatic order transmission
- Automatic order status changes
- Automatic catalog maintenance (Distributed catalog)
- Excludes payment processes

This subset fills in more of the e-Marketplace picture as shown in Figure 10

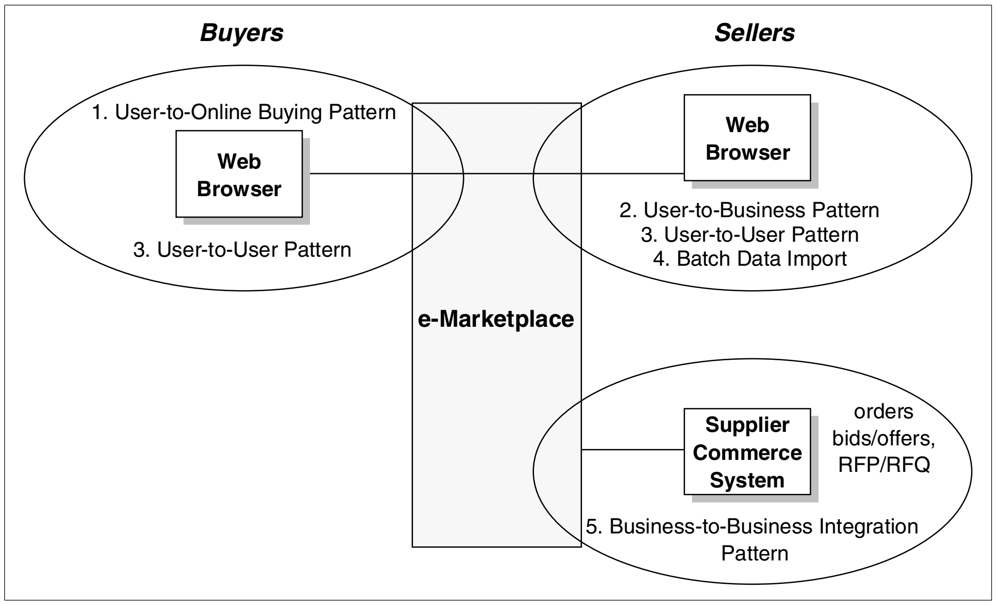


Figure 10. Application Topology - Subset 2

Subset 2 is built from sub-topologies 1 through 5. It essentially adds B2B process integration to the user to online buying basis of the previous subset. This can be a second evolution of the e-Marketplace. In the first subset, buyers and suppliers had no application development or middleware requirements to join the e-Marketplace. They simply needed Web browser access to the Internet. Subset 2 suppliers who wish to participate in an automated fashion in the e-Marketplace must support a B2B protocol with application development and additional middleware.

2.5 Subset 3: Fully integrated marketplace

Here, the automated access to the e-Marketplace is provided to both buyers and sellers. This allows procurement systems to interact directly with the e-Marketplace. This puts and application development and middleware requirement on buyers who want to take advantage of this automated buying feature. The list of additional capability provided by this subset is shown below.

- Automatic RFQ placement
- Notification of buyer of responses to RFQ

- Automatic order placement
- Automatic registration of new buyers to the e-Marketplace
- Excludes payment processes

This gives us almost all of the full e-Marketplace picture as shown in Figure 11 on page 29

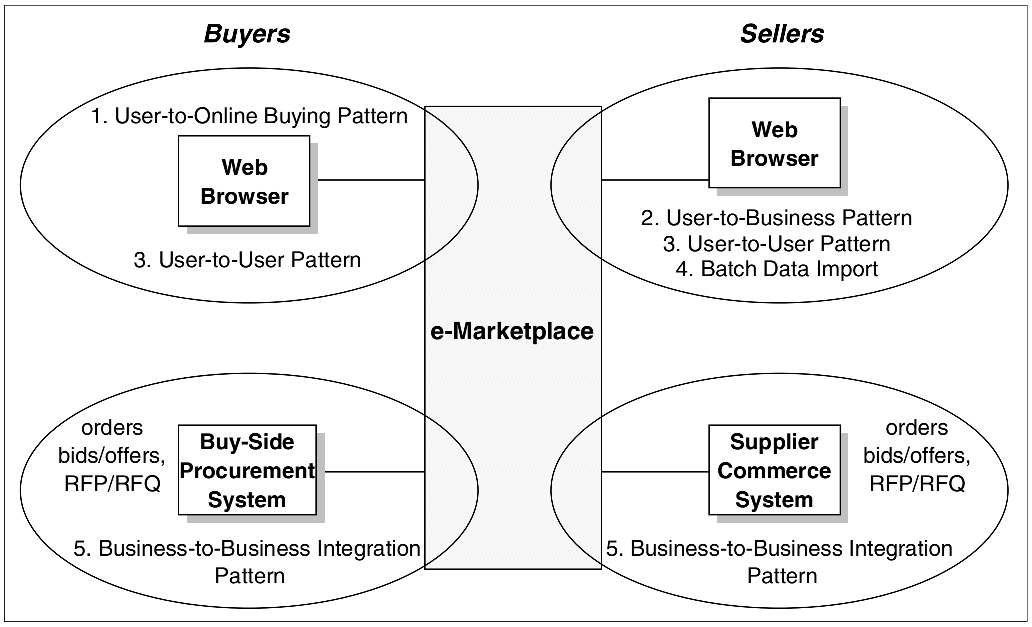

Buyers
Sellers
1. User-to-Online Buying Pattern
Web Browser
3. User-to-User Pattern
Web Browser
2. User-to-Business Pattern
3. User-to-User Pattern
4. Batch Data Import
e-Marketplace
orders bids/offers, RFP/RFQ
Buy-Side Procurement System
5. Business-to-Business Integration Pattern
Supplier Commerce System
orders bids/offers, RFP/RFQ
5. Business-to-Business Integration Pattern

Figure 11. Application topology - subset 3

2.5.0.1 Full e-Marketplace

To complete the picture of an e-Marketplace, the marketplace provider must add to subset 3 the support of payment processes and the integration of the its internal businesses with the commerce functions provided to the marketplace members. This implies the addition of Enterprise Application Integration patterns. More sophisticated supplementary services for e-Marketplace members can also be provided by applying User-to-User patterns to support online community features.

Chapter 3. e-Marketplace runtime topology

In an e-Marketplace, the available runtime topologies depends largely on the integration requirements of the e-Marketplace participants identified by the application topology. For example, a relatively advanced e-Marketplace may provide for automated real-time integration and delivery of product and catalog data from each of the suppliers in the e-Marketplace. This data would then be aggregated into the hub's catalog ready for buyers to purchase those products immediately in the e-Marketplace.

In this chapter, we discuss the runtime topologies and *subsets* of these topologies which relate to the application topologies presented in Chapter 2, "e-Marketplace application topology" on page 21. The topologies presented begin with a simple, Web-integrated e-Marketplace and extend to a completely integrated e-Marketplace comprising automated integration of both buyers and sellers.

While this chapter mentions, at a high level, some of the B2B integration requirements within an e-Marketplace, it does not provide the low-level details of the runtime topologies to support this integration.

We recommend that you read and understand the application topologies discussed in Chapter 2, "e-Marketplace application topology" on page 21 prior to reading this chapter.

3.1 Overview

e-Marketplace runtime topologies often combine the infrastructures of both the e-commerce and integration architectures. Because an e-Marketplace has potential interactions with multiple buyers and sellers, we must consider runtime topologies that address the integration requirements of these participants - even if full integration of buyers and suppliers is not scheduled to be implemented immediately.

The runtime topologies relating to Marketplace Edition implementations are presented as *subsets 1,2* and *3*, where each subset reflects additional enhancements to the basic runtime topology presented in 3.3.1, "Emerging basic runtime topolog" on page 37. Each of these subsets relate to particular application topologies discussed in Chapter 2, "e-Marketplace application topology" on page 21. Accordingly, it is useful to consider topology subsets 2 and 3 as evolutionary, as both of these can be phased in as successors to the previous topologies as we introduce integrated buyers and suppliers.

© Copyright IBM Corp. 2000

Each subset is depicted using *nodes*, representing the core areas of functionality. Some of these nodes may be new to you and others may be familiar. For a description of the functionality at each of these nodes, you should refer to 3.2, “Node types in the e-Marketplace” on page 32.

3.2 Node types in the e-Marketplace

In this section, we discuss the attributes of the nodes presented in our runtime topologies. Some of the nodes discussed in this section represent new areas of functionality specific to an e-Marketplace environment and as a result they may be unfamiliar to you. You may also be surprised to find how many nodes are actually involved in a typical e-Marketplace, particularly as we introduce automated buyer and supplier integration. While it may appear daunting at first, some of the functionality of these nodes can often be implemented under the same application servers as the online-buying application. Also, each node does not necessarily mean a separate physical piece of hardware and the combining of numerous nodes into a single hardware device is common.

3.2.1 Commerce server

The Commerce server node combines the functions of the Web server and the application logic of the online buying front-end. Commonly, the components and interactions on this node are:

- Online buying application
- Transactional Web server
- Connections to database server nodes
- Connections to back-end order processing systems

3.2.1.1 Online buying application

The online buying application provides the functionality for online purchasing of products and services. Typically this involves shoppers accessing a site, browsing the product catalog, adding items to their interest list and submitting and paying for the order. Becoming increasingly common are buying metaphors based on negotiations such as RFQ and Exchange.

3.2.1.2 Transactional Web server

The transaction Web server (TWS) serves the public and user-specific information to the user’s Web browser. The TWS function provides robust services allowing users to communicate with shared applications and data repositories such as accessing hypermedia documents and interacting with applications offered by the site. In an e-Marketplace, the TWS Web server is provided by the e-Marketplace and would typically reside on the e-Marketplace premises within its enterprise network. The transactional Web server incorporates the functions of the Web server and the application server.

Typically, data contained on the TWS node is data that is user-bound via a Web browser such as:

- HTML/JSP
- Application programs such as Java applets

3.2.2 Web server redirector

To allow the Web server and the application server to exist as separate entities, we use a Web server and a redirector. The redirector provides a gateway function between the Web server and the application server and ensures that HTTP requests for servlets or JSPs are forwarded on to the application server.

Typically, the Web server and the redirector sit within the demilitarized zone (DMZ) and the application server resides behind the domain firewall in the enterprise network where it gets the benefit of greater security. Traditionally, static Web content is served by the Web server directly from the DMZ where security is less important.

3.2.3 B2B gateway

The B2B gateway manages the interactions between the trading partners within the e-Marketplace based on an executable contract. An example of an executable contract is a Trading Partner Agreement (TPA) - an extensible markup language (XML) document that defines the general ground rules for such functions as pricing quotations, orders, and acceptances. Its also describes the format for the communications that will be used and provides specifics on actions, security and error handling.

When referring to the actions within a TPA, the trading partner is required to implement service interfaces corresponding to these actions. The service interfaces are started by an executable version of the TPA and is executed under the application server. Some service implementations may also require message brokering services in order to interact with enterprise applications. This can be provided by the Business Protocol Manager or delegated to a dedicated message brokering server.

For more information about standards in B2B interactions, you can visit the Organization for the Advancement of Structured Information Standards (OASIS) at:

```
http://www.oasis-open.org
```

3.2.4 Database server

The database server node represents the data repository for the transactions conducted within the e-Marketplace and relates to the specific business transactions conducted by the user. For example, the data repository may store the order and delivery information for an online order or serve up the banking transaction history for an online banking customer.

It is important to note that the mode of database access is perhaps the most important factor determining the performance of this Web application, in all but the simplest cases. One approach is to collapse the database accesses into a single call or very few calls. This can be achieved via coding and invoking Java stored procedure calls on the database.

3.2.5 Purchaser

In the e-Marketplace, the purchaser node represents a personal computer that supports a commercial Web browser. It will become increasingly common for this node to also be a pervasive computing device capable of displaying content delivered by the e-Marketplace.

The level of browser is expected to support secure sockets layer (SSL) and some level of dynamic hypertext markup language (DHTML). Most online buying applications will send a “cookie” to the browser on this node in order to maintain the shopping session. The cookie contains the session id which is used to reestablish the conversation between each of the user’s interactions with the online buying program.

3.2.6 Integration server

The primary role of the integration server is to coordinate the activities for application integration between the application server and the back-end systems by brokering, transmitting and reformatting messages. Typically, the following functions are performed by the integration server:

- Translate the protocols from the front-end systems to protocols understood by the back-end systems.
- Decompose message constructs from the front-end systems to requests for transactions or data from the back-end system.

- Recompose the replies from the back-end systems.
- Provide the management for the *units of work* required of a number of back-end transactions as a result of a request from the fron- end system.

The primary purpose of the integration server is to relieve front-end systems from the complexities associated with interacting with multiple enterprise applications spanning potentially multiple physical locations. Typically, the front-end server issues a single message to the integration server which deconstructs the message and commences the necessary interactions, transactions and data retrieval. A second purpose for the integration server is to provide additional security between the front-end and the back-end systems. The integration server resides behind the domain firewall and as such, the front-end system does not require any knowledge regarding the addressing of the data repositories. It is common that servers located inside the DMZ do not have direct access to the systems and data repositories required in the transaction but rather to send a message to the integration server to handle these requirements.

3.2.7 Notification server

The notification server provides notification messages which can relate to the current state of a process or when a particular condition is reached in the system. It may provide user-oriented messages such as e-mail containing a completed purchase order or system-oriented message, which may send notification messages to another system.

3.2.8 Workflow server

The function of the workflow server node is to manage the flow of operations for users and applications within the e-Marketplace. The scope of workflow management is categorized into the groups *Macro* and *Micro*.

- Macro

  At the macro level, the workflow server manages functions at a broad level such as governing a complete business process.
- Micro

  At the micro level, the workflow server manages lower-level tasks such things as the flow of user interaction associated with an online product purchase.

3.2.9 Mail server

Commonly, e-Marketplace users are sent notifications via e-mail. The mail server on this node is a POP3-compliant application.

3.2.10 Public key infrastructure (PKI)

PKI is a collection of standards-based technologies and commercial services to support the secure interaction of two unrelated entities (for example, a public user and a corporation) over the Internet. In the context of the topologies defined in this redbook, PKI supports the authentication of the server to the browser client, using the SSL protocol.

3.2.11 Domain name service (DNS)

The DNS node assists in determining the physical network address associated with the symbolic address (URL) of the requested information. The DNS is that of the Internet service provider, although DNS is also implemented on the accessed site.

3.2.12 Protocol firewall and domain firewall

Firewalls provide services that can be used to control access from a less-trusted network to a more-trusted network. Traditional implementations of firewall services include:

- Screening routers (the protocol firewall in this design)
- Application gateways (the domain firewall)

The two firewall nodes provide increasing levels of protection at the expense of increasing computing resource requirements. The protocol firewall is typically implemented as an IP router, while the domain firewall is a dedicated server node.

3.2.13 Search engine

The search engine services user requests to search the catalog of the e-Marketplace.

3.2.14 Delivery gateway

A trading partner's application portal on the Internet is represented by the delivery gateway. It runs under a Web application server and provides an authentication and communications protocol conversion point. In this topology, it runs inside the demilitarized zone, where it acts as a security and routing gateway to the enterprise network.

3.2.15 Personalization

The personalization functions define the roles, organizations and individual players in the e-Marketplace. These roles imply actions that can be performed and data that can be seen. An example of personalization is the definition of different *views* of the catalog for different members.

3.2.16 Content management and aggregated catalog

This node represents the functionality supporting the creation of the data that resides on the database server and commerce server nodes. This data includes catalog information describing items offered by suppliers. The content manager includes methods for allowing multiple suppliers to provide this catalog data. It also represents the function to manage and stage that data into production on the servers. The functionality of this node is quite broad, and might be thought of as encompassing an entire subsystem.

The timely synchronization of several Web servers is sometimes achieved by using a Shared File System as the content storage, capitalizing on the replication capability of this technology.

3.3 Core runtime topologies

In this section, we present the basic high-level runtime topologies that are predominant in many existing commerce implementations. These topologies are becoming known as “proven” runtime topologies because they have existed for some time and are implemented in many production sites. These proven topologies represent the *core* runtime topologies generally considered at the core of an e-Marketplace. It is important to note that the additional subsets presented in this chapter all inherit from this foundation and extend it appropriately.

3.3.1 Emerging basic runtime topolog

This runtime topology and its associated variation is still an emerging pattern, although it is based on a proven User-to-Online Buying runtime topology. It is being reverified as more implementations are completed and cataloged.

This basic runtime topology as shown in Figure 12 is commonly implemented as the foundation for e-Commerce sites and is usually extended in line with the load and functionality requirements of the site.

In this topology, the Web server and the application server are combined to form the commerce server node residing within the demilitarized zone (DMZ). This is separated by a protocol firewall to the outside world and a domain firewall to protect the directory services and primary data repository. Most of the application logic is executed on the commerce server node under the application server and served to the client via the Web server.

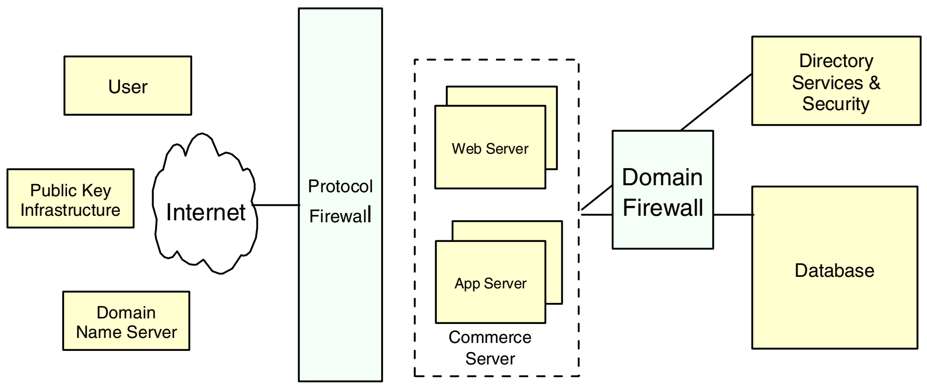


Figure 12. Proven basic runtime topology

While this runtime topology is simple to implement, it has a number of inherent limitations such as:

- Limited availability and failover capability.
- No support for horizontal scalability, as there is only one application server.
- Limited vertical scalability options. However, additional processing power and memory can assist in this area.
- Security for the application server applications is limited to that provided by the protocol firewall. While this is often acceptable for static HTML content residing on the Web server, it is generally inappropriate for securing critical applications.
- The number of simultaneous connections to the Web server is restricted to its capacity.

3.3.2 Emerging basic variation

This variation on the basic runtime topology shown in Figure 13 increases the number of simultaneous client connections by adding to the number of application and Web servers available within the commerce server node. A load balancer is also introduced to distribute incoming requests to the most appropriate (determined by a particular algorithm) Web/application server. The benefits of this approach are increased failover capabilities in addition to the ability to scale horizontally by adding Web and application servers as required.

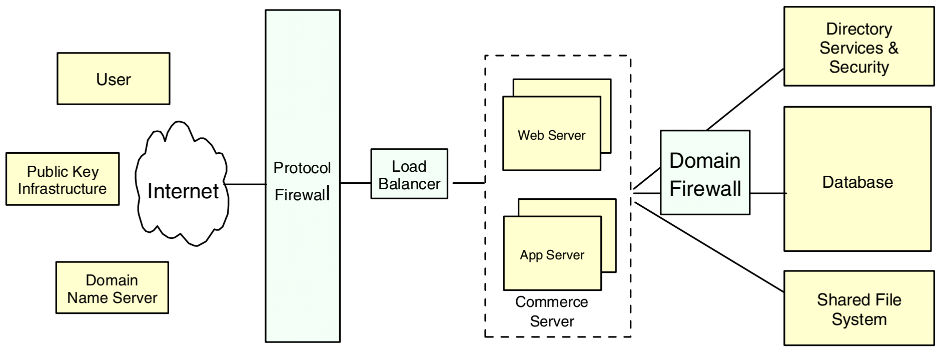


Figure 13. Proven basic variation runtime topology

While this topology improves the availability and performance offered by the the previous topology, it still has a number of potential problems. The primary issue lies in the lack of separation between the Web server and the application server. A common resolution to this is to place the application server behind the domain firewall, where it benefits from the extra security provided by the firewall. You should also be aware that there are limitations to the performance gains achieved through the addition of Web/application servers and it is largely dependent on the extent of the workloads on back-end resources and other subsystems.

3.3.3 Advanced Runtime Topology

The topology and variations described above may prove constraining for e-Marketplaces that are very large or very sophisticated. For those e-Marketplaces, some of the nodes in the topology must be split into multiple nodes with expanded roles and new nodes will need to be introduced. In addition, some of the processing that is shown above in the DMZ should be moved behind the domain firewall. The e-Marketplace characteristics that might drive the need for an advanced runtime topology include a combination of the following:

- Very large user and supplier membership and high usage levels
- Integrated supply change services
- Advanced site management
- Virtual procurement capability
- Multilevel authorization schemes

Work is underway to document this advanced runtime topology. For more details you should visit the patterns Web site at:

`http://www.ibm.com/software/developer/web/patterns`

3.4 e-Marketplace runtime topology subsets

This section discusses three enhancements to the basic proven runtime topology discussed in the previous section but includes additional nodes specific to the e-Marketplaces. You will notice that these topologies do not implement the "proven variation" runtime topology; however, this could be implemented if required. Starting with a simple, Web-integrated e-Marketplace in subset 1, we progress to a fully integrated e-Marketplace in subset 3 that features real-time automated integration of both buyers and suppliers.

3.4.1 Subset 1 - Web integrated e-Marketplace

The Web integrated e-Marketplace provides the simplest runtime topology for an e-Marketplace implementation. This topology provides for the integration of suppliers and buyers via a Web interface only. For suppliers, this topology means manual data exchange with the e-Marketplace using XML scripts or HTML data entry. The obvious benefit of this model is that suppliers can participate in the e-Marketplace without the need to undertake any application development or employ middleware services.

The buying process

In this subset, the process for buying from the online catalog is described below:

1. The buyer logs into the commerce server via the main entry page. If the user is registered with the e-Marketplace, a user profile will be gathered based on information contained in the database server. In the Marketplace Edition, this profile will contain the user's *role*.
2. The personalization engine in the Marketplace Edition presents the buyer with a personalized view of the catalog and other services offered in the e-Marketplace.
3. The buyer proceeds to interact with the site through the static and dynamic pages provided by the site.
4. Items of interest to the buyer are added to the buyer's shopping cart. The database server persists this information in addition to persisting data required to manage the session state. A cookie is delivered to the buyer's browser allowing the commerce server to track the interactions with the site and to reestablish the connections to the shopping cart. Other options

are available to the buyer with regard to product pricing. For example, the commerce server can provide other negotiation mechanisms, such as RFQ, auctions, or exchange.

5. When the buyer purchases the shopping cart items, a purchase order is generated and stored by the commerce server which is subsequently made available to the appropriate suppliers.

The supplier process

1. Like the buyer, the supplier logs into the commerce server but is identified as a supplier via the *supplier* role.

1. The supplier can retrieve the purchase orders generated by the buyer via a Web interface by interacting with the commerce server. They could also respond to RFPs/RFQs at this point.

2. The supplier is also able to provide its catalog data for the e-Marketplace by interacting with the Content Management/Aggregated Catalog node. This may be via an interactive Web-based session or via a mass import of product data in XML format.

In this runtime topology, user authentication and security is provided by the directory and security services node. Additionally, encrypted transmissions are used to protect sensitive data such as the transmittal of credit card numbers.

Figure 14 depicts this runtime topology.

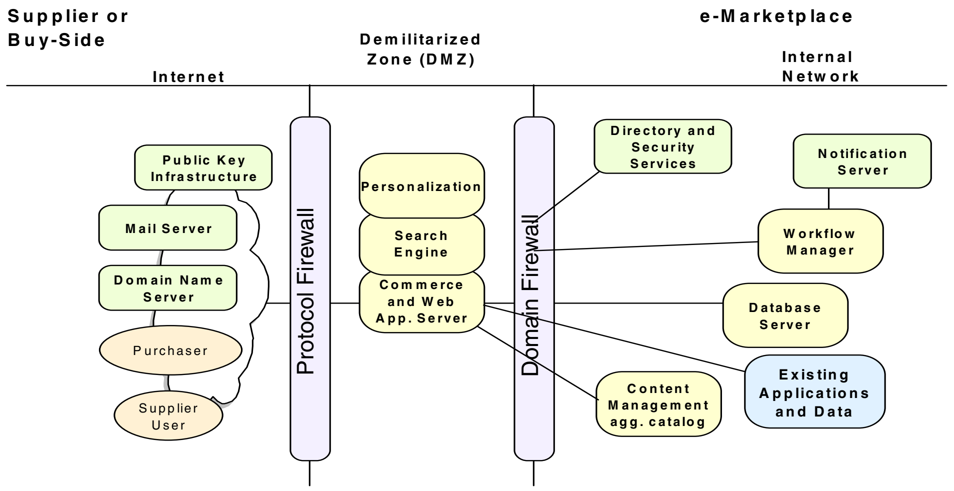


Figure 14. Web integrated e-Marketplace

3.4.2 Subset 2 - e-Marketplace with supplier integration

The next evolution of the e-Marketplace introduces the supplier as an integrated node. By doing this, we also to introduce the Delivery Gateway node and the B2B gateway node.

These additional nodes provide automated integration of data between the supplier and the e-Marketplace. Generally, data fed back and forth between the e-Marketplace and the supplier is in an industry standard form such as XML documents.

Supplier integration is a key aspect of the e-Marketplace and provides, for example, the ability for suppliers to automatically update product data into the e-Marketplace as it become available.

In addition, it allow suppliers to:

- Receive RFPs, RFQs, and purchase orders automatically by the commerce server
- Provide real time inventory information to the e-Marketplace such as the quantity and availability of stocks
- Automatically receive order data and order status changes
- Integrate their business process with the e-Marketplace via the B2B gateway node and the associated delivery gateway node residing within the DMZ.

Figure 15 shows the runtime topology for the integration of suppliers:

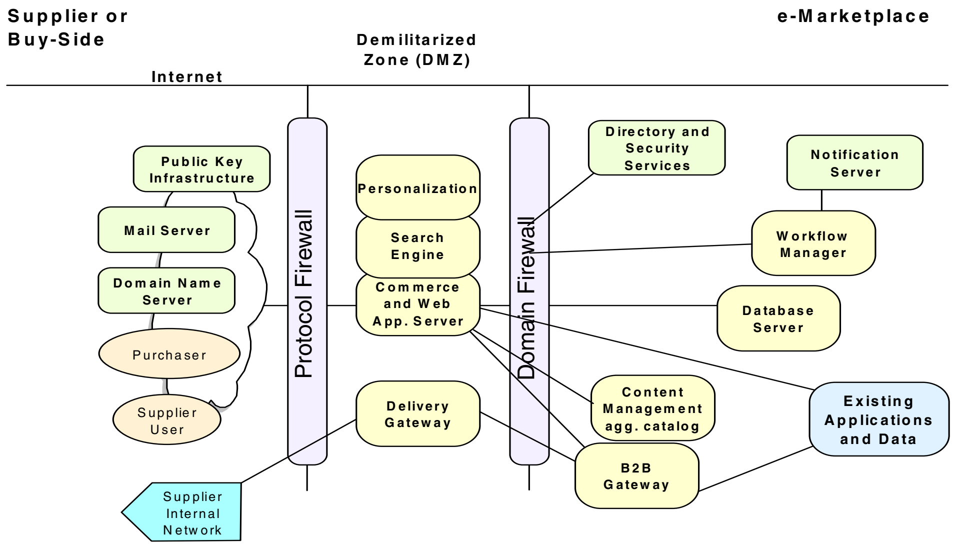


Figure 15. e-Marketplace runtime topology with supplier integration

3.4.3 Subset 3- e-Marketplace with full integration

The third runtime topology builds on the previous two topologies and introduces the buyer as an integrated node with the e-Marketplace. The buyer node allows for buyer-side procurement systems to integrate with the e-Marketplace for interactions such as:

- Automatic order placement
- Automatic RFQ placement and notification of response
- Automatic buyer registration with the e-Marketplace

Again, the integration with the buyer’s system is based on XML documents issued between the commerce server and the buyer’s procurement system.

This runtime topology is unaltered from subset 2, with the exception of the addition of the *buyer node* as shown in Figure 16.

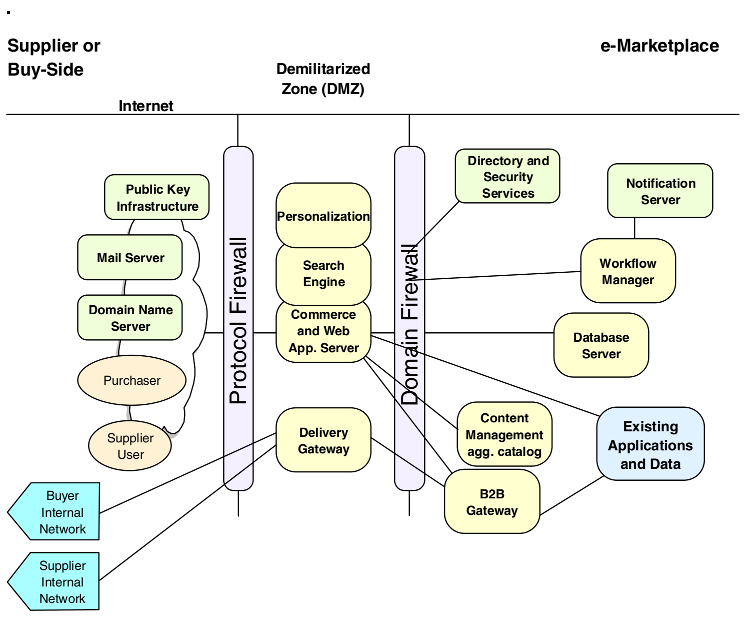


Figure 16. Buyer and supplier integration - a complete e-Marketplace

Chapter 4. e-Marketplace product mapping

The product mapping provides the product and version specific to the logical node. The strength of the IBM WebSphere Application Server family of products for e-business is that the e-business solution, design skills and tools can be used on one platform and easily migrated to other platforms.

The platform chosen should fit into the customer's environment and ensure quality of service, such as scalability and reliability to ensure the solution can grow with the e-business.

The product mapping identifies the platform, software product name, and version. The IBM Application Framework for e-business provides support for many platforms, including IBM AIX, IBM OS/400, IBM OS/390, Sun Solaris, HP-UX, Linux, and Windows NT/2000.

The framework provides the flexibility to allow you to develop and test the application on your runtime platform and easily deploy the application on the customer's platform of choice.

Note

It is common for a company to have a mixture of platforms within the whole integrated e-business solution. The requirement for integration with a mixed platform environment makes the Application Framework for e-business a very appealing choice with its support for many platforms.

When this redbook was written we had only the AIX version of the WebSphere Commerce Suite, Marketplace Edition available, so we have listed product mappings only for the AIX platform.

Note that these levels of code could change from the limited availability stage, as we have installed it, to the final release stage upon general availability.

4.1 AIX product mapping

Figure 17 on page 46 provides an example product mapping for the logical runtime topology using AIX-based nodes for a Web-integrated e-Marketplace. Figure 17 on page 46 illustrates the hardware and software components typically used in a WebSphere Commerce Suite, Marketplace Edition for AIX implementation of an e-Marketplace.

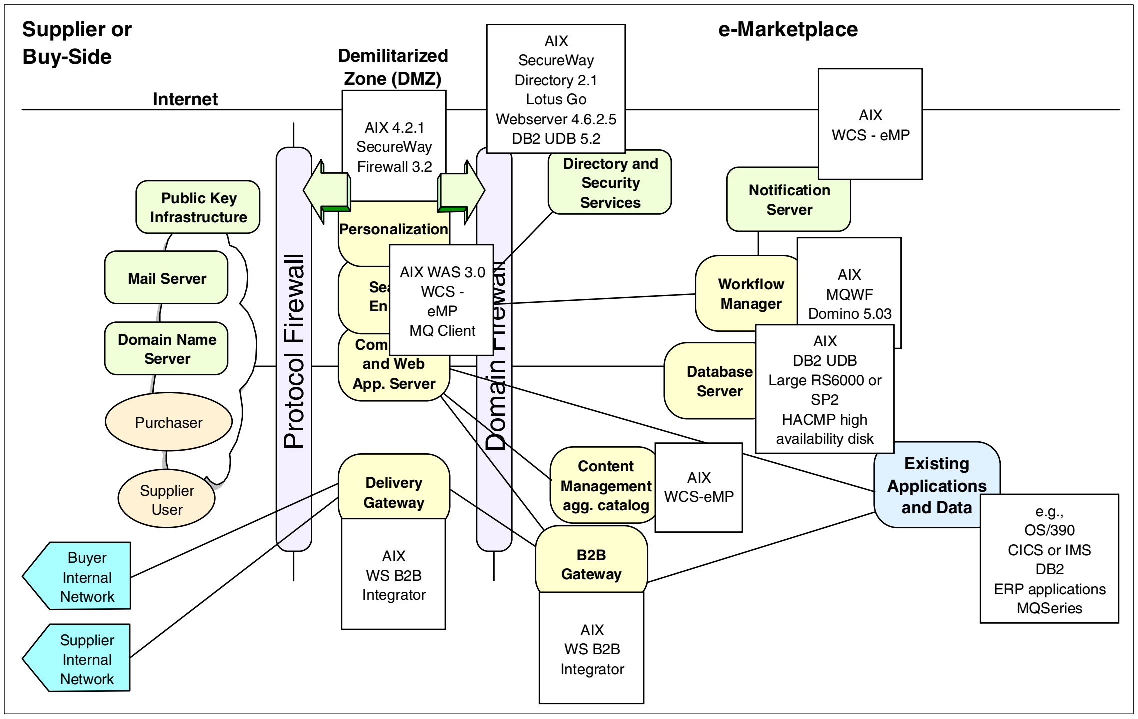

Supplier or Buy-Side
Internet
Demilitarized Zone (DMZ)
e-Marketplace
AIX 4.2.1 SecureWay Firewall 3.2
AIX SecureWay Directory 2.1 Lotus Go Webserver 4.6.2.5 DB2 UDB 5.2
AIX WCS - eMP
Public Key Infrastructure
Mail Server
Domain Name Server
Purchaser
Supplier User
Protocol Firewall
Domain Firewall
Personalization
AIX WAS 3.0 WCS - eMP MQ Client
Commerce and Web App. Server
Directory and Security Services
Notification Server
Workflow Manager
AIX MQWF Domino 5.03
Database Server
AIX DB2 UDB Large RS6000 or SP2 HACMP high availability disk
Existing Applications and Data
e.g., OS/390 CICS or IMS DB2 ERP applications MQSeries
Content Management agg. catalog
AIX WCS-eMP
Delivery Gateway
AIX WS B2B Integrator
B2B Gateway
AIX WS B2B Integrator
Buyer Internal Network
Supplier Internal Network

Figure 17. AIX product mapping

4.1.1 Detailed product mapping - AIX platform

The section provides detailed tables for the major logical nodes within the runtime topology mapping the nodes to specific software product names and versions for the AIX platform. Please note that because the Business-to-Business e-Marketplace Pattern is an emerging pattern and that when we wrote this book WebSphere Commerce Suite, Marketplace Edition was not yet released, these product mappings are our best indication of how we expect an e-Marketplace will be implemented, but we have not been able to test or implement all these nodes, so it is likely that these product mappings will change over time.

Table 1. Commerce server node - AIX platform

Product name	Version
AIX operating system	4.3.3
IBM JDK	1.1.8
IBM JDK 1.1.8 PTF	6 / 7
IBM HTTP Server (Apache)	1.3.6.2
IBM DB2 UDB Universal Database Extended Edition - client	6.1.0.6
IBM WebSphere Application Server Advanced Edition	3.0.2.1
IBM WebSphere Commerce Suite, Marketplace Edition for AIX	V 4.1

Table 2. Database server node - AIX platform

Product name	Version
AIX operating system	4.3.3
IBM DB2 UDB Universal Database Extended Edition - client	6.1.0.6

Table 3. Firewall nodes - AIX platform

Product name	Version
AIX operating system	4.3.3
IBM SecureWay Firewall	4.1

Table 4. Business protocol manager node - AIX platform

Product name	Version
AIX operating system	4.3.3
BPF in WebSphere B2B Integrator	3.0

Table 5. Directory and security service node - AIX platform

Product name	Version
AIX operating system	4.3.3
IBM SecureWay Directory	3.1.1.2

Table 6. Notification server node

Product Name	**Version**
IBM WebSphere Commerce Suite, Marketplace Edition for AIX	V4.1

Table 7. Workflow manager node

Product Name	**Version**
MQSeries Workflow	3.2.2

Part 2. Business-to-Business patterns: e-Marketplace guidelines

© Copyright IBM Corp. 2000

Chapter 5. Performance guidelines

In this chapter we cover a wide range of topics dealing with performance. Good performance starts at the network hardware layer and goes up all the way through to the application layer. We provide some guidelines on performance and will discuss some general topics that should be considered when trouble-shooting or tuning performance in an e-Marketplace application. An e-business infrastructure consists of many layers and we discuss these layers starting at the hardware layer, then the network layer, the operating system layer, and the different software layers like the Web sever and the application. We discuss some specific topics in each one of the layers. This chapter is not about how to tune your applications or infrastructure for performance, as that would be too large a subject for this book. Instead we have created a list of different areas that you should consider when looking at performance in a typical e-Marketplace.There are two ways to improve performance in your e-Marketplace application; one is to keep adding hardware until there is no more space for it, or you can tune your current hardware, network and software to provide optimal performance. We strongly believe that the best way to improve performance is by tuning your hardware, network, and software. There are times when adding new and faster hardware is the only way to get better performance out of your applications, but there are also times when a lot of time and money can be saved if proper sizing of the hardware is done up front.

5.1 Hardware performance

Sizing your hardware is one of the most important issues in performance tuning. You can get better performance by adding more hardware or by tuning your current hardware. When considering hardware performance you have to look at all of the pieces of the hardware, including the CPU, memory, network card, and I/O ports.

5.1.1 CPU

While there are several things that can cause a lot of CPU utilization, one of the more crucial is server-side includes (SSI). Server-side includes cause a lot of CPU utilization when SSI is activated on the Web server the Web server has to parse every single html files for the SSI directed. SSIs should be used sparingly; if you have to use a lot of SSIs then you should consider upgrading your CPU to provide better performance for your applications. URL suffix processing will also cause a lot of CPU utilization. URL suffix processing is when a URL does not exactly match the templates on the PASS directive. For example, if the URL is /index.html and your file name is actually

/index.sambo, the IBM HTTP Web Server will have to do suffix processing and find and return the file index.sambo. The file will eventually be returned, but the IBM HTTP Server will use a lot of CPU cycles to process this type of request.

5.1.2 Memory

One thing that can cause an e-Marketplace application to perform poorly is memory starvation. Most of the time adding more memory to a server will make certain applications perform better, but before you spend money on adding memory to your Web server or application server you need to monitor your applications to find out which application is using the most memory and why. There are several things that can cause a lot of memory utilization, including server-side includes, and poor use of threads. When monitoring your application make sure that it is not using any extra threads for any reason, since unused threads can affect the memory and CPU utilization.

5.1.3 Network card

The network card is another piece of the hardware layer that needs to be looked at with a microscope to make sure it's being used properly. Monitoring of the network card can tell you a lot of things about your Web server or application server and your network. If the network card is constantly being used at 55% to 100% utilization for long periods of time this can cause your application response to slow down. If a network card is constantly being used at 55% to 100% utilization there will be a lot of packets lost and a lot of retries for sending packets because of packet loss. Packets being lost will cause the memory and CPU utilization to increase. If you are using a 10/100 network card make sure your server is running on a network segment that is capable of handle 100 mega bits; if it isn't, tconsider moving your server or rewiring that segment for 100 mega bits access. If your network segment is already at 100 mega bits then you might consider moving to FDDI ,which will give you more throughput and up to 1 giga bit of throughput.

5.1.4 I/O

I/O is an area that should be considered when looking at all the different bottlenecks in your e-Marketplace. I/O takes a lot of memory and CPU cycles. I/O is the input and output of reading and writing files to and from the memory and to and from mass storage devices such as hard drives. To speed up I/O add more memory to your system, or upgrade your CPU, or use a faster RAID controller. If you are not using a RAID controller, consider SCSI hardware and a fast SCSI RAID controller. This will help you protect your data and reduce I/O.

5.2 Network performance

Now let's discuss how to measure network performance. Sometimes it is very hard to figure out where the latency is in your Internet or intranet. Several tools that can help you to determine where the latency is: traceroute is common on most UNIX platforms and tracert can be used on the Windows NT platform. Both of these tools will show you the number off hops, and milliseconds, and the size of the packet was that was used to get to your destination.

Ping can also be used to find latency. It sends ICMP messages, which routers will use to handle the TCP segments of your packet. You will not be able to see some routes with ping because some routers can be configured to ignore all ICMP traffic. Another tool that can be helpful is netstat, which provides a lot of information about your current connections and routing tables, including interface statistics, masquerade connections, netlink messages, and multicast memberships. For more information on netstat, refer to the man pages in most versions of UNIX. Most of the information that is available in netstat for UNIX is also available in netstat for Windows NT.

In the next sections we will discuss the network protocol, frame size, duplexing, hop count, bandwidth utilization, retries or pipelining and network card buffers. All of these different things can be tuned within your network.

5.2.1 Protocol

Open protocols such as TCP/IP and HTTP, are usually very easy on system resources. Most companies that are building Internets and intranets are using open protocols that can be implemented on multiple platforms and their API's and source code are freely available. Multiple layers should be considered when tuning protocols. A good example of this is that HTTP running on top of TCP, which also runs on IP, which also runs on Ethernet. If you tune one protocol you will also have to tune all of them if you want to gain any performance improvements.

5.2.2 Fixed frame size

A frame is data that is transmitted between two network points as a unit, complete with addressing and necessary protocol information. Frames are usually transmitted serially bit by bit and contain a header field and a trailer field that "frames" the data. Some frames, called control frames contain no data. The size of frame that is sent between two points on a network can cause the network bandwidth to be used up, and as a result your server will have to keep trying to resend its frames until they are completely sent. With fixed frame sizes your server will always send exactly the same amount of data in each frame so there isn't a large frame that will use all of your bandwidth.

5.2.3 Duplexing

Duplexing means that both ends of the network link can send and receive signals at the same time. Half-duplexing is a way of bidirectional communication between two points, but the signals can only flow in one direction at a time. Simplexing means that communication can only flow in one direction and never flow in the opposite direction. Most networks are set up using full duplexing so that communication on the network flows in both directions at the same time. Full duplexing will greatly enhance your network performance.

5.2.4 Hop count

A hop is a trip that a data packet or a frame takes from one router to another, or to an intermediate point in another network. The larger the number of hops from the request point ,the longer the request will take. This is also true for an HTTP request. If your Web server is 15 hops from the Internet then every time an HTML page is from your Web server, the user will have to wait for all 30 hops until the request is received: 15 hops to the Web server and 15 hops back to the user. A good rule is to have your Web server no more than two hops from your Internet connection. But because the closer you are to the Internet the more likely you are to have security problems you need to balance security and performance.

5.2.5 Bandwidth utilization and errors

When your company first set an Internet connection to an Internet service provider (ISP), the size of connection to the Internet may have been adequate, but as time passed, your company may be relying more and more on its Internet connections to do business within the company and outside of the company. Periodically the utilization of your company's Internet connection should be evaluated for its total utilization. If your connection to your ISP is 55% or and you are using Ethernet, your network may have a lot of collisions. If your network is spending more time resending the packets that collided than actually sending a packet and completing each round trip, then your ISP connection needs to be upgraded.

5.2.5.1 55% rule
The 55% rule is a good rule to remember when considering whether or not to upgrade your Internet connection. To determine how much of the bandwidth is being used between your company and your ISP, use a sniffer of the LAN segment that runs from your ISP to your entry point or gateway into your company, or ask your ISP for an utilization report on your connection.

5.2.6 Retries or pipelining
Retries or pipelining is the number of packets that your server sends before returning an acknowledgment saying that a connection cannot be made. The larger the number of retries, the more network traffic that will be generated before an error message is sent to your server. Most operating system IP stacks have a way of controlling the number of tries before an acknowledgment is sent. This number should be set low to reduce network traffic that is caused by error acknowledgment.

5.2.7 Network card buffers
Ethernet cards have buffers, as do serial cards, but Ethernet card buffers are usually considerably larger since they process more information. An 8-bit Ethernet card will have an 8 KB buffer and a 16 bit Ethernet card will have a 16 KB buffer. Yyour Ethernet card is 8 or 16 bit your Ethernet card will process either 8 or 16 KB of data. At 10 Mbps, an Ethernet card will fill an 8 KB buffer every 6.6 milliseconds and at 100 Mbps a Ethernet card will fill a 16 KB buffer every 1.3 milliseconds. If your network card is working really hard to handle user requests, consider upgrading your network card and the network segment where your server is.

5.3 Security
Balancing security and performance can be very complicated. Too much security in the wrong part of your network will place an extremely heavy load on your network, resulting in very slow response times. Some types of security will cause a lot of resource overhead and other types of security have very low resource utilization.

5.3.1 Secure Socket Layer or Secure Hypertext Transfer Protocol
Secure Socket Layer (SSL) is a program layer created by Netscape for adding security to the message transmissions of a network. In SSL, the programming for keeping messages confidential is contained in a program layer between an application and the Internet’s TCP/IP layers. The sockets method is used to pass data between a client and a server program in a

network or between program layers in the same computer. SSL uses the public-and-private key encryption system, and also includes the use of a digital certificate. It is an integral part of all Web browsers. SSL adds another layer to your network protocol which means that it has to be tuned with all the other protocol layers to get better performance out of network protocols.

Secure Hypertext Transfer Protocol (HTTPS) is a protocol developed by Netscape and built into most browsers for encrypting and decrypting page requests as well as the pages that are returned by the Web server. HTTPS uses SSL as a sublayer under its regular HTTP application layer. HTTPS uses port 443 instead of HTTP port 80 in its interactions with the lower layer, TCP/IP. SSL uses a 40-bit key encryption algorithm, which is considered to be adequate encryption for doing commerce exchange over the Internet. Both SSL and HTTPS use a 40-bit key encryption algorithm to encrypt requests, which will add anywhere from a 20% percent to 40 percent overhead to a server's CPU utilization.

5.3.2 Encryption

Encryption refers to the conversion of data into encrypted text or ciphered text, decrypting is the process of converting encrypted data back into its original form.

To decrypt data you will need a key to recover the contents of an encrypted signal. The key uses an algorithm that decrypts the work that the encryption algorithm does.

Computers can be used to "break" the cipher. The more complex the encryption algorithm is, the more difficult it is to break the encryption.

Encryption/decryption is a good idea when carrying out any kind of sensitive transaction, such as a credit-card purchase online. The stronger the cipher, the harder it is for unauthorized people to break the encryption.

The use of encryption is a strain on system resources and can cause performance to degrade severely, depending on how many encrypted requests your server has to process. To balance security and performance, only encrypt the transactions that need to be encrypted and not all transactions. If your application needs full encryption for all transactions that are accessing your data, consider moving to a platform that can handle the encryption or consider using hardware encryption. Hardware encryption is usually a specialized expansion board added to your server to do the encryption without putting any strain on the CPU or memory of your system.

5.3.3 Authentication

Authentication is the process of determining whether a user or a process is, in fact, who or what it declares to be. Private and public computer networks commonly use logons and passwords for authentication. Knowledge of the password is assumed to guarantee that the user is authentic. Each user registers initially, using an assigned or self-declared password. On each subsequent use, the user must know and use the previously declared password. The use of digital certificates issued and verified by a Certificate Authority (CA) as part of a Public Key Infrastructure is becoming the standard way to perform authentication on the Internet.

5.4 Operating system

The operating system is one of the most important areas when talking about performance, because it controls all of the hardware. If the operating system is not tuned properly then no matter how fast your hardware is you will not get any performance gains.

5.4.1 Memory

Most servers run many different applications, each one requiring memory. In most operating systems you can set the priority of execution for applications, which means that they each get the memory required.

5.4.2 TCP/IP stack

Transmission Control Protocol/Internet Protocol (TCP/IP) is the basic communications language or protocol of the Internet. It can also be used as a communications protocol in private networks, called intranets, and in extranets. Every system that accesses the Internet directly must have a copy of TCP/IP.

TCP/IP is a two-layered program. The higher layer, the Transmission Control Protocol, manages the assembling of a message or file into smaller packets that are transmitted over the Internet and received by a TCP layer that reassembles the packets into the original message. The lower layer, the Internet Protocol, handles the address part of each packet so that it gets to the right destination. Each gateway computer on the network checks this address to see where to forward the message. Even if some packets from the same message are routed differently from others, they will be reassembled at the destination. TCP/IP uses the client/server model of communication in which a computer user (a client) requests and is provided a service (such as sending a Web page) by another computer (a server) in the network. TCP/IP communication is primarily point-to-point, meaning each communication is from one point (or host computer) in the network to another point or host computer. TCP/IP and the higher-level applications that use it are collectively said to be "stateless" because each client request is considered a new request unrelated to any previous one (unlike ordinary phone conversations that require a dedicated connection for the call duration). Being stateless frees network paths so that everyone can use them continuously. (Note that the TCP layer itself is not stateless as far as any one message is concerned. Its connection remains in place until all packets in a message have been received.)

Many Internet users are familiar with the even higher layer application protocols that use TCP/IP to get to the Internet. These include the World Wide Web's Hypertext Transfer Protocol (HTTP), the File Transfer Protocol (FTP), Telnet which lets you log on to remote computers, and the Simple Mail Transfer Protocol (SMTP). These and other protocols are often packaged together with TCP/IP as a "suite."

Personal computer users usually get to the Internet through the Serial Line Internet Protocol (SLIP) or the Point-to-Point Protocol (PPP). These protocols encapsulate the IP packets so that they can be sent over a dial-up phone connection to an access provider's modem.

Protocols related to TCP/IP include the User Datagram Protocol (UDP), which is used instead of TCP for special purposes. Other protocols are used by network host computers for exchanging router information. These include the Internet Control Message Protocol (ICMP), the Interior Gateway Protocol (IGP), the Exterior Gateway Protocol (EGP), and the Border Gateway Protocol (BGP).

Parameters that impact the different protocol layers are as follows:

- The *no* parameters are the initial network options that affect TCP, UDP and IP and are independent to the adapter type.
- The MTU, or maximum transmission unit, is the largest possible packet size that can be sent on a specific physical medium (Ethernet, token-ring, SP switch, and so on).
- The adapter queues specify the number of packets that can be queued on a specific adapter while it is sending or receiving data. These are specific to an adapter even if there are other adapters of the same type. For a review of the TCP/IP layer model and to clarify the interrelationships, let’s break this down further, step by step:

1. An application performs a write request. Data is copied from the application's working buffer to the socket send buffer.
2. The socket layer passes the data to TCP or UDP.
3. For remote networks, if the data is larger than the maximum segment size (MSS), TCP breaks the data into fragments that comply with the MSS.
4. For local networks, if the data is larger than the MTU, TCP breaks the data into fragments that comply with the MTU.
5. UDP leaves the fragmentation to the IP layer.
6. The interface layer makes sure that no packet exceeds the MTU.
7. The packets are then placed on the adapter output queue, and transmitted to the receiving system.
8. The receiving host places the incoming packets on the adapter's receive queue. They are then passed up to the IP layer.
9. The IP layer then determines if any fragmentation has taken place due to the MTU. If so, it puts the fragments back to their original form and passes the packets to TCP or UDP.
10. TCP reassembles the original segments and puts them on the socket receive buffer in kernel memory or UDP passes the data on to the socket receive buffer in kernel memory.
11. The application's read request causes the appropriate data to be copied from the socket receive buffer to the buffer in the application's working area.

There are many parameters that can affect your network performance. At the device driver layer the transmit queue size is marked by the parameters xmt_que_size. The receive queue size is marked by rec_que_size. At the interface layer is the enforcement of the MTU or segment size as it pertains to what type of network media is being used: Ethernet, token-ring, or others. At the transport layer, performance parameters are set by tcp/udp send/recvspace. The socket layer, between the transport and application layers, has the parameter sb_max, which determines the maximum amount of memory or mbuf space that can be used by TCP or UDP for socket buffers for each socket. Lastly, the parameters listed above all impact system memory.

5.4.3 Web server

A Web server is a program that, using the client/server model and the World Wide Web's Hypertext Transfer Protocol (HTTP), serves the files that send Web pages to Web users (whose computers contain HTTP clients that forward their requests). Every computer on the Internet that contains a Web site must have a Web server program (or else the site files must be sent to a computer that has a Web server program). The most popular Web servers are Apache, a Web server for both 32-bit Windows and UNIX-based operating systems; Microsoft's Internet Information Server (IIS), which comes with the Windows NT server; and Netscape's FastTrack and Enterprise servers. Other Web servers include Novell's Web Server for users of its NetWare operating system and the IBM family of Lotus Domino servers. IBM also provides an Apache based server, IBM HTTP Server powered by Apache (IHS). Web servers often come as part of a larger package of Internet- and intranet-related programs for serving e-mail, downloading requests for FTP files, and building and publishing Web pages. Considerations in choosing a Web server include how well it works with the operating system and other servers, its ability to handle server-side programming, and publishing, search engine, and site building tools that may come with it.

5.4.4 Process handling

This category is primarily related to the HTTPD processes.

MaxClients is the limit on the total number of simultaneous HTTP requests that IHS can serve. Since IHS uses one child server process for each HTTP request, this is the limit of the number of child server processes that are able to run simultaneously. The default value is 150 and maximum value is 2048. The value of this directive can significantly impact your application performance, particularly if it is too high. The optimum value depends on your application. In general:

- Use a lower MaxClients value for a simple, CPU-intensive application.
- Use a higher MaxClients value for a more complex, database-intensive application with longer wait times.

For example, exceptional performance on simple servlets such as “HelloWorld” and “Snoop” have been achieved using values as low as 25. A good approach to tuning this parameter is to start with a setting of 50 and capture the performance under normal load. Repeat your test with settings of 40 and 60. Use this data to refine your tuning. Use small increments and decrements rather than large ones. During these tests, be sure to watch the server CPU utilization. Do not increase the MaxClients setting if the CPU utilization reaches 100% busy and doing so causes server response time to exceed your response time criteria.

StartServers is the number of child server processes that are created when IHS is started. The default value is 5.

MaxSpareServers specifies the largest number of idle HTTPD child processes that are not handling any requests.

MinSpareServers specifies the lowest number of idle HTTPD child processes that are not handling any requests.

If there are fewer active processes than MinSpareServers, then the parent process creates new child processes at a maximum rate of 1 per second. If there are more active processes than MaxSpareServers, then the parent process kills off the excess child processes.

The default value of MaxSpareServers is 10 and MinSpareServers is 5.

These directives can also impact your application performance. For optimum performance runs, keep the MaxClients, the StartServers and the MaxSpareServers directives equal so that CPU is not expended creating and destroying HTTPD child server processes.

MaxRequestsPerChild restricts the number of requests handled by each child HTTPD process. Once this value is reached, the child process terminates. This parameter limits the lifetime of an HTTPD client process to prevent it from using too much memory resource in case of memory leaks. The number specified can be fairly high if stable operation is expected. The default value is 10000.

ListenBacklog is the maximum length of the queue of pending connections from the clients. Generally no tuning is needed or desired. However on some systems it is desirable to increase this when under a TCP SYN flood attack. The default value is 511.

5.4.5 Logging

Web server logs, in particular the access log, record important information about the use of the Web server. However, these logs can become very large and should be pruned and/or archived regularly. Logging can have a surprisingly large impact on response time and throughput. For this reason, you will often find that vendors turn logging off for benchmarking purposes.

5.4.6 SSI

Server-side includes (SSI) allow you to insert information into CGI programs and HTML documents that the server sends to the client. When server-side include processing is enabled, the Web server will parse each byte of every HTML file and CGI program searching for the existence of an SSI directive and, if found, process it. This is a great feature for processing dynamic content, but it requires a large amount of CPU processing. SSI processing can be controlled by the use of the imbeds directive. If you do not use SSIs, set imbeds off in the /etc/httpd.conf file.

5.4.7 CGI-BIN

The IBM Application Framework for e-business defines the infrastructure for developing e-business solutions. This includes the relationship between the Web server and server-side Java elements. The analysis of the performance characteristics of this environment provides some useful guidelines for e-business solution designers, including:

1. Any of the techniques used to connect a Web server to application logic (CGI, in-process API, Java Servlets, etc.) should be insignificant when compared to the time required to execute the application logic.
2. Most existing Web applications have been implemented using CGI. Analysis indicates that Java Servlets provide four to ten times better throughput compared to CGI.
3. Java Servlets generally run faster if they are instantiated and preloaded in servlet.properties. Servlet invocation by class name rather than instance name is slower.

5.4.8 Caching

Integration of Web application servers with back-end systems also raises several performance issues, including:

1. Pre-allocating and caching resource managers:

   Some of the cost associated with accessing resource managers from middle-tier objects involves establishing connections to those resource managers. By pre-allocating and reusing connections, it is possible to greatly reduce the overhead of defining new ones.

2. Caching facilities local to the middle tier:

   Caching state information accessed from back-end systems into the middle-tier environment can promote good performance. This is achievable through pre-fetch and look-aside algorithms that ensure that the back-end will only be accessed as often as is absolutely necessary. Note that it is common, especially when integrating with existing legacy information systems, to access back-end state information required by one object and, in the process, to encounter state information needed by one or more other, related objects. Caching this information as it becomes available may significantly improve overall system performance. Finally, a smart caching mechanism can ensure that updates are only made to a back-end system when the underlying state information of a given object has actually changed.

3. Optimistic locking mechanisms:

   In some cases, applications place large numbers of unnecessary locks on resource managers. These locks may be unnecessary if no attempts to use the resource concurrently actually occur. In these situations it would be better to place no locks on the resource managers, and instead check for conflicting usage as part of transactional commit.

   Specifically, you should consider locking all affected resources only once: at the end of a given unit of work. Then compare the relevant resource manager values at that time with the values obtained when the unit of work began. In the absence of intervening changes, the current unit of work can be committed. Otherwise, it can be rolled back with an exception returned to the client. In either case, you may be able to reduce overall locking overhead and improve total system throughput and performance. Note that this “optimistic” locking strategy is inappropriate for some types of applications and domains. When updates to the same resource occur on a frequent basis, traditional (or “pessimistic”) locking mechanisms should be used. It’s also worth mentioning that multiple resources should typically be accessed by multiple applications in the same sequence to reduce potential “deadlock” situations.

4. Implementation “pushdown”:

   Object-based solutions that must coexist with legacy systems should complement (versus compete with) these systems. For example, a query invoked on a virtual collection of objects whose state maps to a relational database manager should first be translated into native SQL statements. These SQL statements, processed using the optimization technology provided by the database manager, can then return a result set whose values implicitly identify candidate objects satisfying the query. In this way, a minimal amount of processing is required in “object space”, leaving the bulk of the work to be handled instead by resource managers that have been perfecting high performance solutions over the course of many years. As a final point that is specific to our query example, it’s notable that an object-based implementation of query should also be able to return multiple elements back from a single method call, and that the query result set should be demand-driven (meaning that objects should only be activated on the server if a client actually requests them).

5.4.9 Web server stay alive

These directives deal with the persistent connection feature of the HTTP V1.1 specification. With HTTP V1.0, each HTTP session establishes a new TCP connection. If your home page has a lot of images, you will need to establish TCP connections many times to send all data for one page. The persistent connection feature is designed to avoid this behavior. After one session is finished, the connection still remains and the next request can re-use the connection. If IHS gets an HTTP/1.1 request, IHS can re-use the connection until it receives the connection close request. The directives to understand are:

- KeepAlive - Whether or not to allow persistent connections (more than one request per connection). Set to "off" to deactivate. The default is on.
- KeepAliveTimeout - Number of seconds to wait for the next request. The default value is 15. To avoid waiting too long for the next request, you can specify the number of seconds to wait. Once the request has been received, the Timeout directive will apply.
- MaxKeepAliveRequests - The maximum number of requests to allow during a persistent connection. Set to 0 to allow an unlimited number.
- Timeout - Sets the number of seconds the IHS waits for these three events:
  - Time taken to receive a GET request.
  - Time taken between receipt of TCP packets on a POST or PUT request.
  - Time taken between acknowledgments on transmissions of TCP packets in responses. The default value is 300.

5.5 Application server

For an application server in WebSphere V3 there are a number of settings that have an impact on performance. In addition to the settings for the application server itself there are also a number of settings for the various components of an application server; servlet engine, Web application, and the EJB container, that can affect performance. The following parameters are set using the WebSphere Application Server Administrative Console:

- Java Virtual Machine (JVM)
- Threads

These parameters will be discussed in detail in the following sections.

5.5.1 Selecting a JVM

High-performance JVMs are critical for good performance. Here are some considerations when evaluating JVMs:

1. Java compiler and virtual machine technology changes rapidly. Today a JIT version of a compiler may provide the best performance for your solution; Tomorrow it may be a static compiler, and next week there may be a new technique.
2. A JVM should be certified for portability by a recognized certification authority such as JavaSoft.
3. JVMs that have been optimized for a specific operating system tend to perform better than other JVMs.
4. Systems with symmetric multiprocessors (SMP) tend to perform better because of the thread support in Java. Use JVMs that effectively support SMP systems.
5. Evaluate the trade-off between the stability of the current release of a JVM and the performance enhancements available in the newest beta release and/or production version.

Generally speaking, the selection of a JVM should be of little concern to you. The performance of applications can often be better improved by improving the runtime characteristics of the algorithms used. However, there are circumstances, such as computationally intensive programs, where an improvement in the performance of the underlying JVM will add to the solution's overall performance.

5.5.2 Threads

Each application server has its own thread pool from which it uses threads to process remote method invocations. This pool size varies throughout the servers lifetime. Threads are created when needed and destroyed when there are too many idle threads. Each application server has a setting for thread pool size. Consider increasing the thread pool size if analysis shows that the maximum number of threads in the pool are frequently in use.

5.5.3 Caching

The addition of memory to a system almost always improves performance. This is because physical I/O is a relatively expensive operation in terms of latency. It makes intuitive sense that by, dedicating memory in a Web server to store frequently accessed HTML pages and images, you will improve performance. As a rule of thumb, your Web server should have enough RAM to accommodate all network buffers, frequently used applications, images and HTML, including those mounted via DFS. This is especially important for dynamically generated pages that can be reused. For the Web server there are several places you can cache your static items to help improve performance:

- Use a network router such as the IBM 2216 Nways Multiaccess Connector.
- Use a Web proxy cache such as the one found in WebSphere Performance Pack.

5.6 Java Virtual Machine

Java is an interpreted language. Java source code must first be compiled into portable bytecodes that can then be interpreted in the Java Virtual Machine(JVM) on the local system. Naturally, when looking for performance improvements for your Java applications, the quality of the JVM is the place to start. For example, all JVMs implement an advanced feature called Garbage Collection (GC) to boost the programming productivity and to avoid the common pitfalls of traditional programming languages: memory leaks. However, GC can slow down the Java program execution because most GCs utilize a “stop and copy” technology. In ongoing research, GCs have been developed that do not exhibit any of the problems described above. These GCs run incrementally and take many characteristics of the Java language into account.

5.6.1 Just -In-Time compiler

A Just-In-Time (JIT) compiler converts bytecodes into native code on-the-fly with some optimization, and it should run much faster than just a Java Virtual Machine (JVM). A JIT works well for computationally intensive programs that execute the same segment of instructions repeatedly. But it does not yield significant improvement for programs that are I/O intensive or cause a lot of garbage collection. This is because the program code takes up such a small amount of time and thus any optimization would become negligible. In addition, JITs are limited in the extent of optimization they can do because their compile is a runtime cost. Also, since JIT compilers normally do not have the “global view” of the executed code, the code they generate is of poorer quality than the code generated by static compilers.

5.6.2 Adaptive compiler

Adaptive compilers try to overcome the weaknesses of JIT compilers with adaptive optimization techniques. The idea is to intelligently select, compile and optimize frequently used and/or resource-intensive portions of the code, so-called “spots.” Once these spots have been determined, optimization techniques known from traditional compilers are applied to compile a highly optimized version of the code. The compiled code is then stored in a cache, ready to be executed when the spot is executed again. Note that as with a JIT, results of compilation are not kept between runs/users.

5.6.3 Static compiler

A static compiler compiles Java source code into the underlying machine’s native code that is then executed without interpretation. This approach is similar to traditional program development where the code in written, compiled and, if necessary, debugged. Static compilation can also be applied to the bytecode generated by some Java compilers. This approach is applicable when the original Java source code is not available. Static compilation of Java bytecodes is possible because this code is the same across Java implementations and builds the basis for the portability of the Java language. However, it is important to note that the portability of the Java application will not be affected by static compilation.

5.7 Database

Does your application requirement call for a database? If your Web site is going to access a database exclusively, plan your database around your Web site.

5.7.1 Indexes

When you build your indexes, build them so that you correlate with data that will be searched for. Otherwise a lot of extra cycles and disk storage will be wasted.

5.7.2 Standard Query Language (SQL)

SQL can cause a lot of over head such as CPU utilization and memory utilization. There are some good best practices for SQL.

1. Cache the most used data if possible.
2. Control how many update queries are performed.
3. Precompile as much of SQL as possible. This will greatly reduce the execution time of your SQL.
4. Use stored procedures where possible.
5. Limit table locking and data locking.

6. Debug all of your SQL before it goes to production. Make the SQL worked properly before it is moved into production. A bad SQL statement can crash your system.
7. Do not give everyone access to run all SQL statements: give only the user group that needs the data the rights to query its own data.

5.8 References on performance

- *Web Performance Tuning* by Patrick Killelea, published by O'Reilly
- *WebSphere V3 Performance Tuning Guide*, SG24-5657-00.

Chapter 6. Technology options

This chapter provides guidelines on the technology options that should be considered when developing e-Marketplace applications, using WebSphere Commerce Suite, Marketplace Edition for AIX.

The IBM offerings for developing commerce sites include various products that are fully explained on the e-business Web pages. Full technical documentation, references and whitepapers are obtainable from:

`http://www.ibm.com/software/webservers/commerce/`

This chapter builds *Patterns for e-business: User-to-Online Buying Pattern using WebSphere Commerce Suite V4.1*, SG24-6156-00 which is prerequisite reading.Chapter 6 of that redbook fully describes all of the prerequisite technology options. We will only concentrate on the additonal software needed to complete the e-Marketplace offering.

The e-Marketplace runtime topology is a combined e-commerce and application integration infrastructure. It includes browser connections to buyers and sellers, and B2B application integration with suppliers and sellers.

6.1 Web clients

This section is organized as follows:

- Web client overview
- Web browser
- Markup languages
- Client-side scripts
- Java applets

6.1.1 Web client overview

There are two types of Web clients:

- Application client

  Application clients are primarily large Java applets or Java applications. These clients provide rich graphical user interfaces compared to HTML clients. They may communicate with the Web application server over a number of protocols including HTTP, IIOP, MQ, etc. Application clients communicate with the Web application server primarily to receive data rather than pre-formatted HTML pages. All of the user interface processing is performed on the client side. In addition, under this model,

© Copyright IBM Corp. 2000

some parts of the business logic can also be processed on the client-side. These kind of client applications are not covered in this book.

- Web browser client

  A Web browser client is an application client that uses a Web browser such as Netscape Navigator or Microsoft Internet Explorer. Web browser clients use the HTTP protocol to communicate with the Web application server to display HTML. In addition, they are capable of processing client-side JavaScript for enhancing navigation to perform simple input validation and to handle simple errors.

The Application Framework recommends “thin clients” with little or no application logic. Applications are managed on the server and downloaded to the requesting clients. The client portions of the applications should be implemented in HTML, dynamic HTML (DHTML), XML, and Java applets as seen in Figure 18 on page 70.

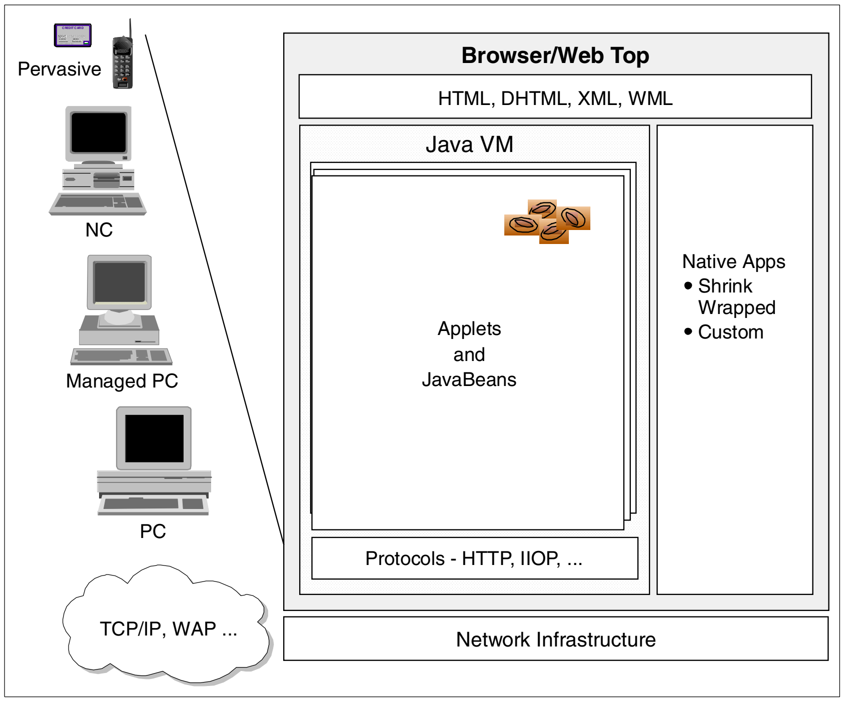


Figure 18. Web client technology model

6.1.2 Web browser

A Web browser is a fundamental component of the Web client. For PC-based clients, the browser typically incorporates support for HTML, DHTML, JavaScript and Java. Some browsers are beginning to add support for XML as well. Under user control, there is a whole range of additional technologies that can be configured as “plug-ins”, such as RealPlayer from RealNetworks or Macromedia Flash.

As an application designer you must consider the level of technology you can assume will be available in the user’s browser. You can add logic to your application to enable slight modifications based upon the browser level. Also, when adding features that require a plug-in, you need to consider what portion of your intended user community will have that capability.

For an e-business application that is to be accessed by the broadest set of users with varying browser capabilities, the client is often written in HTML with no other technologies. On an exception basis, limited use of other technologies, such as using JavaScript for simple edit checks, can then be considered based on the value to the user and the policy of the organization for whom the project is being developed.

6.1.3 Markup languages

In this section we discuss the most important markup language used in the development of Web applications.

6.1.3.1 HTML

HTML is a document markup language with support for hyperlinks, that is rendered by the browser. It includes tags for simple form controls. Many e-business applications are assembled strictly using HTML. This has the advantage that the client-side Web application can be a simple HTML browser, enabling a less capable client to execute an e-business application.

The HTML specification defines user interface (UI) elements for text with various fonts and colors, lists, tables, images, and forms (text fields, buttons, checkboxes, and radio buttons). These elements are adequate to display the user interface for most applications. The disadvantage, however, is that these elements have a generic look and feel, and they lack customization. As a result, some e-business application developers augment HTML with other user interface technologies to enhance the visual experience, subject to maintaining access by the intended user base and compliance with company policy on Web client technologies.

Almost all browsers support HTML V3.2, which is often the lowest common denominator for building the clientside of an application.

6.1.3.2 Dynamic HTML

DHTML allows a high degree of flexibility in designing and displaying a user interface. In particular, DHTML includes cascading style sheets (CSS) that enable different fonts, margins, and line spacing for various parts of the display to be created. These elements can be accurately positioned using absolute coordinates.

Another advantage of DHTML is that it increases the level of functionality of an HTML page through a document object model and event model. The document object enables scripting languages such as JavaScript to control parts of the HTML page. For example, text and images can be moved about the screen, and hidden or shown, under the command of a script. Also, scripting can be used to change the color or image of a link when the mouse is moved over it, or to validate a text input field of a form without having to send it to the server.

Unfortunately there are several disadvantages with using DHTML. The greatest of these is that two different implementations (Netscape and Microsoft) exist and are found only on the more recent browser versions. A small, basic set of functionality is common to both, but differences appear in most areas. The significant difference is that Microsoft allows the content of the HTML page to be modified by using either JScript or VBScript, while Netscape allows the content to be manipulated (moved, hidden, shown) only using JavaScript.

Due to the browser incompatibility issues, DHTML is not recommended in environments where mixed levels and brands of browsers are present.

6.1.3.3 XML, XSL

Extensible Markup Language (XML) allows you to specify your own markup language with tags specified in a Document Type Definition (DTD). Actual content streams are then produced that use this markup. The content streams can be transformed to other content streams by using (XSL eXtensible Stylesheet Language).

XML is a framework for defining document markup languages and is predicted to become the primary approach to document exchange over the Internet. In simple terms, a document markup language is a set of elements (frequently called tags) that have one or more of the following functions:

- Describe the structure of the document

- Describe the content of the document
- Control how the document is presented to the user

XML and Hypertext Markup Language (HTML) are derived from the more complex Standard Generalized Markup Language (SGML). SGML's complexity and high cost of implementation spurred the interest in developing alternatives.

HTML is the most widely used markup language for Web documents. As the popularity of HTML increases, the limitations of the language have become more apparent. The limitations of HTML include:

- Restricting the user to a relatively small set of tags.
- HTML authors cannot create their own HTML tags. Commercially available Web browsers have no knowledge of tags that are not part of the HTML standards.
- Control presentation is contained in the same file as the tags that describe the document content.
- Although HTML 4 and Cascading Style Sheets enable HTML authors to separate content from presentation, HTML 4 remains weak in its ability to describe the content of a document.

XML overcomes many of the limitations of HTML and other markup languages, while providing capabilities that are not a part of the earlier languages. In the XML document, the tag names convey the meaning of the data they contain. The structure of the document is easily discerned and follows a pattern. In contrast, the HTML tag names reveal little about the meaning of their content and the structure is not particularly useful for manipulating the document and exchanging it between applications.

XSL is a language for expressing style sheets and provides two major functions with XML:

- Language for transforming XML documents
- XML vocabulary for specifying formatting semantics

An XSL style sheet specifies the presentation of a class of XML documents by describing how an instance of the class is transformed into an XML document that uses the formatting vocabulary.

6.1.4 JavaScript

JavaScript is a cross-platform object-oriented scripting language. JavaScript is a great utility language for Web applications because of the browser and document objects that the language supports. Client-side JavaScript provides the capability to interact with HTML forms. You can use JavaScript to validate user input on the client and help improve the performance of your Web application by reducing the number of requests that flow over the network to the server.

ECMA, a European standards body, has published a standard (ECMA-262) which is based on JavaScript (from Netscape) and JScript (from Microsoft) called ECMAScript. The ECMAScript standard defines a core set of objects for scripting in Web browsers. JavaScript and JScript implement a superset of ECMAScript. You can find the ECMAScript Language Specification at

http://www.ecma.ch/stand/ECMA-262.htm.

To address various client-side requirements, Netscape and Microsoft have extended their implementations of JavaScript in Version 1.2 by adding new browser objects. Due to Netscape and Microsoft extensions beyond the standard JavaScript 1.2, the extensions must detect the browser being used and select the correct statements to run.

The use of JavaScript on the server side of a Web application is not recommended, given the alternatives available with Java. Where your design indicates the value of using JavaScript, for example for simple edit checking, use JavaScript 1.1, which contains the core elements of the ECMAScript standard.

6.1.5 Java applets

Java applets provide the most flexible user interface (UI) technology that can be run in a Web browser. Java provide a rich set of UI elements in comparison to the equivalent HTML UI elements. In addition, Java's rich programming language provides an infinite set of UI elements than can be built and used. There are many libraries available that offer common UI elements, such as tables, scrolling text, spreadsheets, editors, graphs, charts, etc.

A Java applet is a program written in Java that is downloaded from the Web server and run on the Web browser. The applet to be run is specified in the HTML page using an APPLET tag:

```
<APPLET CODEBASE="/mydir" CODE="myapplet.class" width=400 height=100>
   <PARAM NAME="myParameter" VALUE="myValue">
</APPLET>
```

In this example, a Java applet called myapplet will run. Parameters provide an effective way to send data to an applet with the use of the PARAM tag. The applet has access to the parameter data and can easily use it as input to the display logic.

Java applets can also request a new HTML page from the Web application server. This provides an equivalent function to the HTML FORM submit function. The advantage of loading a new HTML page with an applet is that it can do the obvious (a button being clicked) or the unique (the editing of a cell in a spreadsheet), unlike HTML forms.

Java applets seldom consist of just one class file. Large applets may reference hundreds of class files. Making a request for each of these class files individually can tax any server and also tax the network capacity. To address this issue, Java provides a means to package the class files into a JAR or CAB file. This reduces the number of load requests from hundreds to just one. Netscape and HotJava support JAR files simply by adding an `ARCHIVE="myjarfile.jar"` variable within the APPLET tag. Internet Explorer uses CAB files specified as an applet parameter within the APPLET tag. In both cases, executing an applet contained within a JAR or CAB file provides faster load times than individual class files. While Netscape and Internet Explorer use different APPLET tags to identify the packaged class files, a single HTML page containing both tags can be created to support both browsers. Each browser simply ignores the other's tag.

A disadvantage of using Java applets for UI generation is that this requires a version of Java that is supported by the Web browser. Often, the Java Virtual Machine (JVM) contained within the browser lags behind the most current and capable versions of the Java Developers Kit (JDK), which presents a dilemma. Do I force my user community to use a specific version of a browser, or do I use an older version of the Java technology that is supported within the Web browsers for the client-side application?

Note

The leading browsers from Netscape and Microsoft support variant levels of JDK 1.1 and each have different security models for signed applets.

Another disadvantage of Java applets is that any classes such as widgets and business logic that are not included as part of the Java support in the browser must be loaded from the Web server as they are needed. If these additional classes are large, the initialization of the applet may take from seconds to minutes, depending upon the speed of the connection to the Internet.

Due to the shortcomings discussed, the use of Java applets is not recommended in environments where mixed levels and brands of browsers are present. Small applets may be used in rare cases where HTML UI elements are insufficient to express the semantics of the client-side Web application user interface. If it is absolutely necessary to use an applet, care should be taken to include UI elements that are core Java classes whenever possible.

6.1.6 C++ CGI

CGI scripts can be written in interpretive languages such as Perl, TCL, REXX and UNIX shell scripts. Alternatively, CGI can be written in C++, as is the case in WebSphere Commerce Suite. When the CGI request arrives at the Web server, it forks off a new process to run the script and sets up the environment variables for this process. Included in the variables are any form input as well as information about the requestor such as the browser or its level. Any response from the CGI script is processed by the Web server according to its content. Typically the response will be an HTML stream that will then be sent back to the client and appear in the client's browser window.

The sequence of events for a CGI script are as follows:

1. A client at a browser clicks a link that contains the /bin-cgi/ phrase (positioned immediately after the host name portion of the URL).
2. The Web server intercepts the request and looks in its CGI-BIN directory for the script named in the URL portion following the /cgi-bin/ keyword.
3. A process is forked off to run the interpreter that will process the script.
4. The process is passed a set of environment variables that include both system information and any parameters that were attached to the name of the cgi-bin script command name.
5. The script is interpreted (runs) and in the process of executing, usually will generate an HTML stream mixed with data to return to the client. The HTML stream is constructed on the fly per the design and function of the script.
6. The process terminates and resources are freed.
7. The client sees the resulting HTML stream as yet another Web page or item on a Web page. Page counters are a typical small CGI script item.

6.2 WebSphere Application Server

This section is focused on Java server-side programming for the Web application server. The Application Framework for e-business recommends the technology model seen in Figure 19 on page 77, for a Web application server.

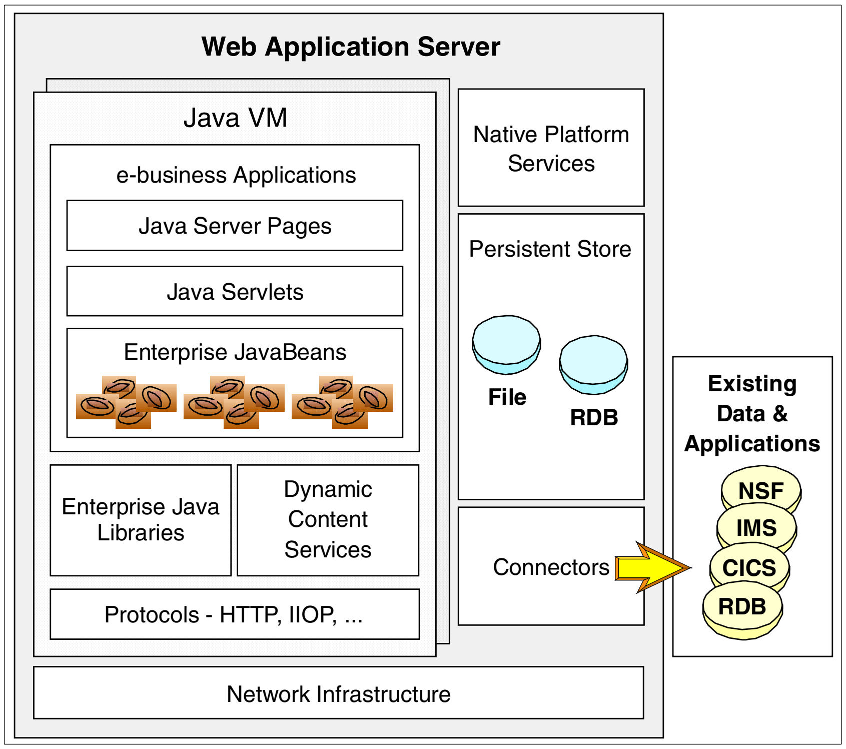


Figure 19. Web application programming environment

While there have been many other models for a Web application server, the Framework has had great industry support. For more details on the Java APIs discussed in this section, see *Java Enterprise in a Nutshell*.

Before we look at the technologies and APIs available in the Web application programming environment, we need to review the operational components on this node. The selection of the Web/HTTP server and the Java Virtual Machine (JVM) are critical to the application in areas such as robustness, performance and availability.

In Chapter 7, “Application design guidelines” on page 89, we discuss the Model-View-Controller (MVC) design structure for user interfaces. In summary, the MVC for the Web application programming model states the following:

- Model

  The model is represented to the view and interaction controller by using a set of JavaBean components.

- View

  The view is generally best implemented using JavaServer Pages.

- Controller

  The interaction controller, which is primarily concerned with processing the HTTP request and invoking the correct business or UI logic, often lends itself to an implementation as a servlet.

6.2.1 XML

XML and XSL style sheets can be used on the serverside to encode content streams and parse them for different clients, thus enabling you to develop applications for both a range of PC browsers and for the emerging pervasive devices. The content is in XML format and an XML parser is used to transform the XML to output streams presented within XSL style sheets. This general capability is known as transcoding and is not limited to XML-based technology.

The appropriate design decision here is how much control over the content transformation you need in your application. You will want to consider when it is appropriate to use this dynamic content generation and when there are advantages to having servlets or JSPs specific to certain device types.

XML is also used as a means to specify the content of messages between servers, whether the two servers are within an enterprise or represent a business-to-business connection. The critical factor here is the agreement between parties on the message schema, which is specified in an XML DTD document.

An XML parser is used to extract specific content from the message stream. Your design will need to consider whether to use an event-based approach, for which the SAX API is appropriate, or to navigate the tree structure of the document using the DOM API. For more information visit:

`http://www.sun.com/xml/`

6.2.2 JavaServer Pages (JSP)

JavaServer Pages were designed to simplify the process of creating pages by separating Web presentation from Web content. In the page construction logic of a Web application, the response sent to the client is often a combination of template data and dynamically generated data. In this situation, it is much easier to work with JSP technology than to do everything with servlets.

The chief advantage JSPs have over Java Servlets is that they are closer to the presentation medium. A JavaServer Page (JSP) is an HTML page. JSPs can contain all the HTML tags that Web authors are familiar with. A JSP may contain fragments of Java code which encapsulate the logic that generates the content for the page. These code fragments may call out to JavaBeans to access reusable components and back-end data. JavaServer Pages are compiled into servlets before being executed on the Web application server. JavaServer Pages are the recommended choice for implementing the "View" for the Web browser client. For those cases where the code required is large percentage of the page content, and the HTML minimal, writing a Java Servlet is a better choice, for readability and maintenance reasons.

The current level of the JSP API is 1.1. To learn more about JSPs visit:

`http://java.sun.com/products/jsp/`

6.2.3 Java Servlets

Java Servlets provide a replacement for CGI-based techniques in Web application programming. Servlets provide a component-based, platform-independent method for building Web-based applications, without the performance limitations of CGI program. They interact with the servlet engine, running on the Web application server, through HTTP requests and responses, which are encapsulated as objects in the servlet.

One of the attractions of using servlets is that the API is a very accessible for a Java programmer to master. Servlets are a core technology in the Web application programming model. They are the recommended choice for implementing the "Interaction Controller" classes that handle HTTP requests received from the Web client.

The current level of the servlet API is 2.2. To learn more about Java Servlets visit:

`http://java.sun.com/products/servlet/`

6.2.4 JavaBeans

JavaBeans component architecture enables developers to create reusable software components that can then be assembled together using visual application builder tools, such as, IBM's VisualAge for Java or IBM WebSphere Studio. JavaBeans also known as “Beans” may be visual or non-visual.

Recommended uses of Beans with JSPs and servlets:

- As the client interface to the “model layer”. An “interaction controller” servlet will use this JavaBean interface.
- As the client interface to other resources. In some cases this may be generated for you by a tool.
- As a component that incorporates a number of property-value pairs for use by other components or classes. For example, the JavaServer Pages specification includes a set of tags for accessing JavaBean properties.

The current level of the JavaBeans API is 1.01. To learn more about JavaBeans visit:

`http://java.sun.com/products/beans/`

6.2.5 Enterprise JavaBeans (EJB)

Enterprise JavaBeans (EJB) are distinguished from JavaBeans in that they are designed to be installed on a server, and accessed remotely by a client. The EJB framework provides a standard for server-side components with transactional characteristics.

The EJB framework specifies the responsibilities of the EJB developer and the EJB container provider. The intent is that the “plumbing” required to implement transactions or database access can be implemented by the EJB container. The EJB developer specifies the required transactional and security characteristics of an EJB in a deployment descriptor (this is sometimes referred to as declarative programming). In a separate step, the EJB is then deployed to the EJB container provided by the application server vendor of choice.

There are two types of Enterprise JavaBeans:

1. Session
2. Entity

Session bean characteristics:

- Executes on behalf of a single client
- Can be transactional.
- Can update data in an underlying database.
- Is relatively short lived.
- Is destroyed when the EJB server is stopped. The client has to establish a new session bean to continue computation.
- Does not represent persistent data that should be stored in a database.
- Provides a scalable runtime environment to execute a large number of session beans concurrently.

Entity bean characteristics:

- Represents data in a database.
- Can be transactional.
- Shared access from multiple users.
- Can be long lived (lives as long as the data in the database).
- Survives restarts of the EJB server. A restart is transparent to the client.
- Provides a scalable runtime environment for a large number of concurrently active entity objects.

Typically an entity bean is used for information that has to survive system restarts, while in session beans, the data is transient and does not survive when the client's browser is closed. For example, a shopping cart containing information that may be discarded uses a session bean, and an invoice issued after the purchase of the items is an entity bean.

An important design choice when implementing entity beans is whether to use Bean Managed Persistence (BMP), in which case you must code the JDBC logic, or Container Managed Persistence (CMP), where the database access logic is handled by the EJB container.

The business logic of a Web application often accesses data in a database. EJB entity beans provide a convenient way to wrap the relational database layer in an object layer, hiding the complexity of database access. A single business task may involve accessing several tables in a database. Modeling rows in those tables with entity beans makes it easier for your application logic to manipulate the data.

The current level of the Enterprise JavaBean API is 1.1. The most significant changes from EJB 1.0 are the use of XML-based deployment descriptors and the need for vendors to implement entity bean support to claim EJB compliance. To learn more about Enterprise JavaBeans visit:

`http://java.sun.com/products/ejb/`

6.2.6 Additional enterprise Java APIs

In developing a server-side application, you may also need to be familiar with the following enterprise Java class libraries:

- Java Naming and Directory Interface (JNDI)

  This package provides a common API to a directory service. Service provider implementations include those for LDAP directories, RMI and CORBA object registries. Sample uses of JNDI include:

  - Accessing a user profile from an LDAP directory
  - 4.1 in their other applications.
  - Locating and accessing an EJB Home

- Remote Method Invocation (RMI)

  RMI and RMI over IIOP are part of the EJB specification as the access method for clients accessing EJB services. RMI can also be used to implement limited function Java servers.

- Java Message Service (JMS)

  The JMS API enables a Java programmer to access message-oriented middleware such as MQSeries. Such messaging middleware is a popular choice for accessing existing enterprise systems.

- Java Transaction API (JTA)

  This Java API is for working with transaction services, based on the XA standard. With the availability of EJB servers, you are less likely to use this API directly.

6.3 WebSphere Commerce Suite technology

To build and run an e-Marketplace site, you need software that provides a secure, scalable, and dependable environment that you can trust. WebSphere Commerce Suite, Marketplace Edition for AIX is based upon WebSphere Commerce Suite software that powers some of the busiest, most successful e-commerce sites available online. The WebSphere Commerce Suite, Marketplace Edition for AIX is packaged with the following award-winning software to provide a complete end-to-end solution:

- IBM DB2 Universal Database provides a scalable, reliable database to store your data.
- IBM HTTP Server provides a powerful Web server and is based on the popular Apache Web server.
- IBM WebSphere Application Server for servlets and JSP technology.
- IBM SecureWay Directory is a robust directory server.

In the following sections we briefly discuss the technology of Marketplace Edition and describe where it differs from that of WebSphere Commerce Suite. More details of the Marketplace Edition technology can be found in Chapter 7, “Application design guidelines” on page 89.

6.3.1 Commands, tasks and overrideable functions

Marketplace Edition reuses the C++ based architecture of WebSphere Commerce Suite where it provides the necessary functionality. An example is in the orders infrastructure. New functionality in Marketplace Edition is implemented in a Java framework which is discussed in 7.6, “Marketplace Edition application logic” on page 102.

Commands perform a specific business process, such as adding an item to the shopping cart, processing an order, updating a shopper’s address book, or displaying a specific product page. A command represents a static piece of business process that delegates the implementation of specific pieces of business logic to tasks. This separation between commands and tasks allows different implementations of the same command to be performed depending upon merchant parameters.

While commands and overridable functions are actual pieces of code, a task is not. A task is a contract between a command and an overridable function, or between two overridable functions. These rules dictate the following:

- The work that the caller expects the overridable function to perform
- The parameters that the caller passes to the overridable function
- The parameters and other results that the caller expects the overridable function to return

WebSphere Commerce Suite defines three types of tasks: view tasks, process tasks and exception tasks.

Overridable functions provide extension points with which you can customize the way the server functions. Overridable functions are written in C++ and are implemented as platform-specific CGIs for the WebSphere Commerce Suite runtime platform, such as Windows NT, AIX, Solaris, OS/400, and OS/390

For more information on commands, tasks and overrideble functions see *Patterns for e-business: User-to-Online Buying Pattern using WebSphere Commerce Suite V4.1*, SG24-6156-00 and also the online manual that comes with WebSphere Commerce Suite 4.1, which is called Commands, Tasks, Overridable Functions

6.3.2 Database

The WebSphere Commerce Suite, Marketplace Edition is implemented using a relational database. The database contains all the information that is used by the Marketplace Edition system, including information about individual organizations, members, items, and prices, etc.

The Marketplace Edition database consists of a collection of tables which reuse and extend the tables existing in WebSphere Commerce Suite.

6.3.3 Net.Data

WebSphere Commerce Suite used Net.Data to build dynamic pages by executing macros that retrieved data from the Commerce Suite database.

Net.Data is an interpretive programming language that is used to facilitate interaction between users and the database. In the WebSphere Commerce Suite system, a Net.Data macro is a file that retrieves data from the WebSphere Commerce Suite database and displays it as a formatted Web page. Typically, it contains functions that execute SQL queries, HTML tags (which can also contain JavaScript code), and Net.Data statements. The SQL statements search for and retrieve information from the WebSphere Commerce Suite database, the HTML defines the layout of the search results, and the Net.Data statements control the flow of the output.

6.3.4 WebSphere Application Server integration - Marketplace Edition

WebSphere Commerce Suite, Marketplace Edition provides support for JavaServer Pages, Java Servlets, and JavaBeans using the WebSphere Application Server. Marketplace Edition does not employ EJBs in the current implementation.

6.3.4.1 JavaServer Pages in the Marketplace Edition

The Marketplace Edition uses JavaServer Pages (JSPs) extensively to build dynamic displays. WebSphere Application Server provides the runtime support for JSPs. When a customer requests a JSP page, a JSP-enabled engine (in this case, the WebSphere Application Server) interprets the JSP tags and scriptlets, creates the content in the form of an HTML page, and returns it to the browser.

Using JavaServer Pages to create dynamic HTML pages allows you to separate the task of coding data retrieval from the task of creating the display format. For example a Java programmer can develop Java beans to retrieve data from the database, while a media specialist can design the look and feel of the page.

6.3.4.2 Java Servlets in the Marketplace Edition

In the past, server-side requests were often handled by CGI (Common Gateway Interface) programs. Servlets provide advantages over CGI programs in that they can run with less overhead, are more portable and are easier to maintain. Marketplace Edition makes extensive use of servlets.

The servlet engine is the function of the Web application server that manages servlets. It manages the creation and destruction of servlets, specifying which servlets should be automatically loaded into memory at startup and which should be loaded upon initial request.

6.3.4.3 JavaBeans in Marketplace Edition

The Marketplace Edition makes extensive use of Java beans for both commands and data. Command beans include model, view and error beans. Data beans are used to provide formatted data for display in JSPs. We describe the use of beans in Marketplace Edition in more detail in Chapter 7, “Application design guidelines” on page 89 and in 10.5, “Marketplace Edition programming model” on page 209.

6.3.5 Connectors

e-business connectors are gateway products that enable you to access enterprise and legacy applications and data from your Web application. Connector products provide Java interfaces for accessing database, data communications, messaging and distributed file system services.

IBM provides a significant set of e-business connectors with tool support, for CICS, Encina, IMS, MQSeries, DB2, SAP and Domino. IBM connectors are based on the Common Connector Framework (CCF). For resources on System/390, IBM is delivering native connectors based on CCF.

The command bean model allows you to code to the specific connector interface(s) of your choice while hiding the connector logic from the rest of the Web application.

6.3.5.1 JDBC and SQLJ

The business logic in a Web application will access information in a database for a database-centric scenario. JDBC is a Java API for database-independent connectivity. It provides a straightforward way to map SQL types to Java types. With JDBC you can connect to your relational databases and create and execute dynamic SQL statements in Java.

JDBC drivers are RDBMS specific, provided by the DBMS vendor, but implement the standard set of interfaces defined in the JDBC API. Given common schemas between two databases, an application can be switched between one and the other by changing the JDBC driver name and URL. A common practice is to place the JDBC driver name and URL information in a property or configuration file.

There are four types of JDBC drivers from which you can choose, based on the characteristics of your application:

- Type 1: JDBC-ODBC bridge drivers. This type of driver, packaged with the JDK, requires an ODBC driver and was introduced to enable database access for Java developers in the absence of any other type of driver.
- Type 2: Native API Partly Java drivers. This type of driver uses the client API of the DBMS and requires the binaries for the database client software. This type of driver offers performance advantages but introduces native calls from the JVM.
- Type 3: Net-protocol All Java drivers. A generic network protocol is used with this type of driver. Portability is a major advantage of this type of driver, but it has the limitation that it requires intermediate middleware to convert the Net-protocol to the DBMS protocol.
- Type 4: Native-protocol All Java drivers. This type of driver is portable and uses the protocol of the DBMS. Type 3 and 4 drivers are well suited for applets that access a database server on an intranet, since they only require Java code to be downloaded.

An important technique used to enhance the scalability of Web applications is connection pooling, which may be provided by the application server. When application logic in a user session needs access to a database resource, rather than establishing and later dropping a new database connection, the code requests a connection from an established pool, returning it to the pool when no longer required.

SQLJ provides a simplified syntax for JDBC that allows you to write SQL-like statements directly in your Java source code. The SQLJ preprocessor generates static SQL providing better performance than dynamic SQL. SQLJ will also generate iterator Java classes. These iterators allow you to navigate query results using a very simple “get next” protocol.

As your design takes shape, based on its desired performance and sophistication you may see the need to investigate SQLJ or enterprise Java beans. The most recent level of the JDBC specification is 2.0, but many JDBC drivers you use will still implement 1.0.

6.4 Where to find more information

For more information on topics discussed in this chapter see:

- *WebSphere Studio and VisualAge for Java Servlet and JSP Programming*, SG24-5755-00
- Flanagan, David, *JavaScript: The Definitive Guide*, Third Edition, O'Reilly & Associates, Inc., 1998
- Maruyama, Hiroshi, Kent Tamura and Naohiko Uramoto, *XML and Java: Developing Web Applications*, Addison-Wesley 1999
- Flanagan, David, Jim Farley, William Crawford and Kris Magnusson, *Java Enterprise in a Nutshell*, O’Reilly & Associates, Inc., 1999
- For information on the IBM Application Framework for e-business:
  `http://www.ibm.com/software/ebusiness/`
- For information about the ECMAScript language specification:
  `http://www.ecma.ch/stand/ECMA-262.htm/`
- To learn more about Java technology;

  see `http://www.javasoft.com/products/`
- To learn more about the IBM Application Framework for e-business, see *IBM Application Framework for e-business: Understanding Technology Choices* white paper found at
  `http://www.ibm.com/software/ebusiness/buildapps/understand.html/`
- To learn more about XML visit: `http://www.ibm.com/developer/xml` or `http://www.sun.com/xml/`.

Chapter 7. Application design guidelines

Applications running in the Marketplace Edition are Java-based applications that run under the WebSphere Application Server. Accordingly, many of the considerations involved in the design of these applications are similar to those for conventional e-commerce sites developed with WebSphere Commerce Suite 41.

In this chapter, we present these considerations in the context of the components, technologies and techniques available to you within the Marketplace Edition application environment. Beginning with an introduction to the features and core technologies offered in the Marketplace Edition, we progress to discussing the core components of the Marketplace Edition environment by looking at both the application elements it contains as well as the general structure of an e-Marketplace application. Following this, we look more closely at the technical details of how some of these technologies work and interact with the other components inside the e-Marketplace. Finally, we cover a number of topics including session management, security, and performance to assist you in identifying potential areas of weakness in your application design.

7.1 High-level feature summary of the Marketplace Edition

It is important that you have an understanding of the components and features available in the Marketplace Edition in order to design the best possible application for your problem domain.

While this is covered thoroughly in Chapter 10, “Marketplace Edition overview” on page 195, some of the primary components and features in Marketplace Edition are presented in this section as they may have an impact on how you approach the design your applications.

7.1.1 Features

At a high level, the components available in the Marketplace Edition span the areas of:

- Ordering
- Membership and registration
- Access control
- Catalog and catalog aggregation
- Approval flow

- Negotiation subsystem - exchange, RFQ and auction
- Pricing and contracts
- Reporting
- Management of the hub business

As you can see, there is a great variety of application components offered in the Marketplace Edition product. The result of this is that your applications and customizations are likely to be far more complex than traditional online buying applications.

7.1.2 Modifications to WCS 4.1

While the basis of the Marketplace Edition is WebSphere Commerce Suite 4.1, there are a number of significant enhancements to this foundation.

Some of the areas of change are:

- Database schema
- Aggregate catalog (comprising of data from multiple supplier catalogs)
- C++ programming model
- The addition of user roles
- Approval flows

7.1.3 Processes

Modifications have been made to the traditional process of online buying. No longer are we limited to simply adding an item to our shopping cart. We now have new buying metaphors such as RFQ and Exchange in addition to the Auction facility. We also have changes to how pricing is determined - base prices, contractual prices agreed via RFQ or bids via the auction system.

In addition to this, we have to think about the integration of suppliers and buyers. We can choose to integrate these entities via Web clients or via more sophisticated methods such as real-time integration using message broking technologies.

7.1.4 Players

The players that we have in a typical e-Marketplace are significantly different from a conventional, e-commerce implementation. We are no longer limited to the simple many-to-one buyer and supplier model typical of e-commerce sites.

More specifically, we now have:

- Multiple buyers
- Multiple suppliers
- Multiple administrators - both at the hub level and organization level
- Integration to buyer and supplier systems involving *users and roles* in each of these organizational entities

This is discussed in more detail in Chapter 10, “Marketplace Edition overview” on page 195 which provides use-case scenarios for each player in the Marketplace.

7.2 Understanding the Marketplace Edition technologies

The technologies on which your Marketplace Edition application is based are described briefly in this section. Primarily, development is centred on Java through the use of JSPs, servlets and JavaBeans. However, this can be augmented where necessary through the use of C++ and Net.Data.

Note

Some of the information provided in this section is based on information contained in the *Servlet and JSP Programming with IBM WebSphere and VisualAge for Java,* SG24-5755. We recommend that you read this to gain a thorough understanding of servlets, JavaBeans, JavaServer Pages, the WebSphere Application Server, and the VisualAge for Java programming environment.

7.2.1 Servlets

Servlets are Java server-side components that provide the request/reply mechanism between a client and the server over protocols such as HTTP. Servlets run within the Java Virtual Machine (JVM) of the application server and are typically responsible for executing the business logic required for a particular client request.

Figure 20 shows a high level view of a typical client-to-servlet interaction.

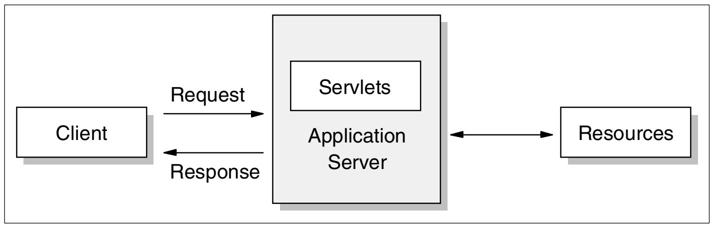


Figure 20. High-level flow of a servlet

Servlets are based on a Servlet API that is found in the following two Java packages:

- javax.servlet
- javax.servlet.http

These packages differ in that javax.servlet is a protocol-independent version of the Servlet API where javax.servlet.http works specifically with the HTTP protocol.

The Marketplace Edition makes heavy use of servlets to provide the core functionality for controlling interactions between the client and the server for commonly performed tasks within the e-Marketplace. You can also develop your own servlets to perform custom functionality.

7.2.2 JavaServer Pages

JavaServer Pages (JSPs) provide the ability to visually create static and dynamic-content Web pages using tools similar to HTML page editors. JSPs are, Java server-side technology and are responsible for generating and formatting the output that is sent to the client-side browser.

Creating a JSP page is very similar to creating an HTML page with the exception of additional tags supporting the embedding of dynamic content. In a JSP, dynamic content is commonly displayed to the user by embedding the properties of JavaBean objects inside the JSP. Dynamic content may also be generated as a result of script-language elements that can also be embedded with the JSP.

How JSPs work

The *first* time a request is made by the client to display a JSP page, the JSP engine residing on the application server parses the contents of the JSP page and creates a servlet source code file based on its contents. The servlet is then compiled by the Java compiler resulting in a compiled class file. The class file is then instantiated and made available to service requests and provide responses to the output stream. The servlet remains in service until its is explicitly stopped, either individually or by stopping the application server.

Figure 21 shows the lifecycle of a JSP for the first time invocation.

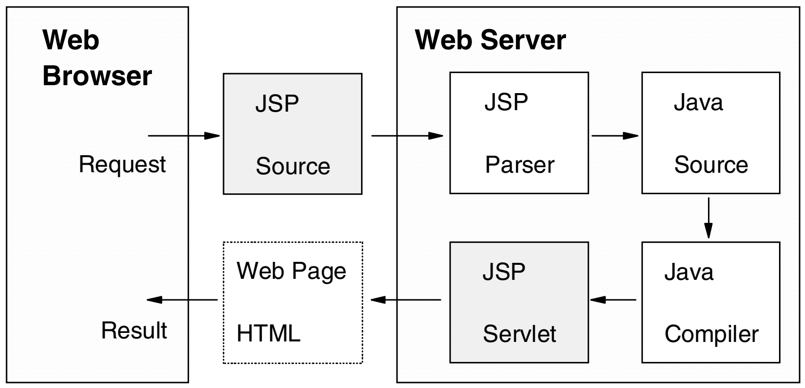


Figure 21. JSP lifecycle for the first invocation

Because a JSP is compiled into a servlet, subsequent requests for the JSP simply invoke the normal request and response mechanisms as per standard servlets.

Most of your application customizations in Marketplace Edition will use JSPs at the presentation layer. WebSphere Commerce Suite, Marketplace Edition for AIX supports both the JSP .91 specification and the JSP 1.0 specification.

7.2.3 JavaBeans

JavaBeans are platform-independent reusable software components that you develop to encapsulate the functionality performed in your applications. JavaBeans are based on Sun Microsystem's JavaBean specification which defines the fundamental architecture that all JavaBeans conform to. Sun's definition states that these beans "can be manipulated visually in a builder tool".

JavaBeans support the following features:

- Properties
- Methods
- Events

- Introspection
- Persistence

Properties are attributes of the bean that have *set* and *get* methods that can be called from other components. Events define a framework for one component to notify another when something noteworthy happens. The JavaBeans specification defines a set of conventions for defining properties, methods, and events. Using this convention, builder tools can analyze the bean to allow for visual manipulation. In addition, beans should implement the persistence mechanism allowing the customized JavaBeans state to be stored and retrieved when necessary. JavaBeans can be either visual or non-visual; however, they should all allow manipulation by visual builder tools. This is usually done by using Java introspection techniques.

Further details on the JavaBeans Specification can be found at:

`http://java.sun.com/beans/index.htm`

7.2.4 Net.Data

WebSphere Commerce Suite uses Net.Data macros to retrieve database data and format that data into an HTML-formatted Web page. Net.Data macros contain functions that execute SQL queries, HTML tags (which can also contain JavaScript code), and Net.Data statements. SQL queries define the information retrieved from the WebSphere Commerce Suite database, the HTML defines the layout of this information, and the Net.Data statements control the flow of the output.

A Net.Data macro usually contains the sections:

- Define
- Function Definition
- HTML Input
- HTML Report

7.3 General application design guidelines

When developing any Web-based applications the architect must consider a core set of design criteria. These are covered briefly below.

- Client

  Design of the application is determined by the type of application required on the client, applets or HTML pages, for example. Web applications in

Marketplace Edition are based on thin-client technology employing JSPs to display content to the client, servlets to control the interactions and JavaBeans to perform the business processes and encapsulate the required data.

- Member integration

  Member integration involves the application design elements associated with interactions with e-Marketplace members such as suppliers and buyers. Our applications much consider the implications of integrating these members, potentially in real time, using message broking and message transport technologies. Alternatively, your design may simply involve Web client batch updates for this integration.

- Look and feel

  Our marketplace must be designed with the appropriate look and feel in line with the products and services offered. The technologies used in the look and feel should not place a heavy load on the browser client.

- Content management

  Content management technology is important in e-Marketplace sites that also offer value-add portal facilities. Often, in addition to products, an e-Marketplace provides up-to-the-minute broadcasts on industry events, technologies and movements. Content management software provides these information feeds and should be a consideration in your design, even if its implementation is not an immediate deliverable.

- Statelessness

  The state of interactions between the client and the server in a Web application is stateless by default. We must consider techniques to maintain user information across interactions, such as the use of cookies and URL rewriting.

- Application speed and performance

  Users should not have to wait excessive times for their requests to be processed. Web application designers should consider the overhead associated with database requests, servlet processing and real-time integration the e-Marketplace members' systems.

- Security

  Security planning involves selecting the correct security model for the solution. Consider user authentication, protocol encryption services, DMZs and firewalls and application level security, such as command security, which is discussed in 7.9.3, “Command access” on page 119.

7.4 Application elements

Application design in the Marketplace Edition implements many of the attributes associated with the Web Application Programming Model, which is promoted as part of the IBM Application Framework for e-business.

This primary attributes that have driven the development of this framework are:

- Ease and speed of development and deployment

  Applications is this framework are component-based server-side Java applications providing rapid development time, separation of development tasks and ease of deployment.

- Independence of client device

  By developing applications using a thin-client architecture, we open our options with regard to the devices we wish to support. Content can be easily rendered to support new types of client devices, using the IBM transcoding technology for example.

- Portability

  Java is a standards-based, open development platform that allows us to migrate our applications to other Java-supporting server environments with minimal effort.

- Extend existing core assets

  We must have the ability to interface to and leverage existing enterprise systems and business processes.

In support of all of these concepts, WebSphere Commerce Suite, Marketplace Edition for AIX offers a complete environment of components, which are described in this section.

7.4.1 Clients

Clients are responsible for accepting user input, sending that input to the server and subsequently accepting and rendering the response from the server into a format suitable to the user. There are two type of clients, HTML clients and application clients.

- HTML clients

  HTML clients represent the thin client browser-based applications developed in a Marketplace Edition application. The responsibilities of the client are to render the browser content, submit requests to the server, and receive responses from the server over HTTP.

- Application clients

  Application clients are Java-based applications that also communicate with the server. However, the applications typically provide the UI to render the application content as opposed to the server-side UI generated for HTML clients. Application clients offer a greater variety of protocols in addition to HTTP with which to communicate with the server, IIOP for example. Application clients are considered “fat clients”, since they often contain business-rule logic that is executed on the client side.

- Which one is better?

  HTML clients are generally the most popular presentation layer used to provide the UI and interaction metaphor for the user. This is primarily due to the following advantages that it offers over application clients:

  - It is browser independent and therefore platform independent. Nearly all browsers can display the HTML produced by the server, allowing for clients to operate on multiple platforms such as Linux or Macintosh.
  - Download time for a page is minimal. With an application client, the application is often slow to download and adds an additional burden on the client requiring the correct Java runtime environment for it to operate under.
  - All of the logic to generate the content is serverside and is therefore easy to maintain, modify and redeploy if required. Server-side logic adds the benefit of additional security provided at the server.

7.4.2 WebSphere Commerce Suite

The WebSphere Commerce Suite represents the grouping of a number of important components for the operating environment. It is comprised of a number of sub-components described below.

- Web server

  The Web server and application server work together to receive and respond to HTTP requests and responses from the client. The Web server handles the work for static HTML pages. However, it passes any request for a dynamic page (JSP for example) to the application server. The majority of pages in a Marketplace Edition application are dynamically generated pages.

- Application Server

  The application server represents the heart of the Marketplace Edition. It is responsible for coordinating the tasks required to generate a response document to the client including:

  - Responding to requests for dynamic data.
  - Controlling the business logic that must be executed based on the information contained in the request.
  - Coordinating the interaction with the appropriate back-end systems if required.
  - Generating the response document for delivery back to the client.
- e-Marketplace server

  For the purposes of providing a logical grouping, the e-Marketplace server represents the components which provide the e-Business services within the e-Marketplace. It encompasses the following sub-components:

  - Marketplace engine and business logic

    This represents the business logic and services available within Marketplace Edition. It represents the boundary for the procedures and processes conducted in the e-Marketplace applications such as the ordering process, RFP/RFQ, auctions, pricing and contracts, and so on. The e-Marketplace server represents the “engine room” of an e-Marketplace application.

  - Page construction services

    These services generate the dynamic pages displayed to the client utilizing Java Servlets, JSPs and JavaBeans.

  - Connectors

    Connectors provide the Java programmer with interfaces to other systems and data repositories using application-specific protocols. Many connectors are available including connectors for SAP, MQ Series, JDBC, CICS, IMS, Encina and Domino.

  - Application messaging services

    This component provides the functionality for intercommunication between applications using message delivery services such as those provided by MQ Series.

  - Business process integration and workflow services

    This extends on the application message services by providing a complete message broking, routing and transaction services.

The functions of the e-Marketplace server may be split over multiple physical boxes.

- Commerce database

This is the master repository for data pertaining to the e-Marketplace. It includes buyer shopper and supplier definitions, the aggregate product catalog, shipping, tax and a host of other e-Marketplace-related data.

- Application services

  Marketplace Edition provides a number of inherent application services such as:

  - Session Management

    This refers to the application server's ability to manage the state of each user's interaction with the server using the HttpSession API.

  - User tracking

    User-specific data can be automatically persisted and retrieved through the WebSphere Application Server's UserProfile API. This feature does not require the programmer to manually code the persistence logic.

  - Security and authentication

    The WebSphere Application Server provides the security features required of e-Business sites such as authentication, authorization, data integrity, protocol encryption and non-repudiation mechanisms (some e-commerce transactions represent agreements that cannot be repudiated). Technologies such as SSL and LDAP are employed in these areas.

  - Systems management

    Complete support for systems management is provided allowing the site to scale and perform strongly through mechanisms such as load balancing and failover. The Marketplace Edition offers a host of services in this area, which are covered in Chapter 9, "System management guidelines" on page 133.

7.5 Application Structure

The application structure within a Marketplace Edition site is composed of two application areas - the Web application and the Marketplace server processes.

7.5.1 Web application

The applications that you develop in the Marketplace Edition typically implement the Model-View-Controller design pattern. This design pattern is inherent in Marketplace Edition as the JavaBeans, servlets and model beans which you use in your application are provided as part of the environment.

The exception to this is if you add your own servlets and JavaBeans which perform logic *outside* of the functionality provided by Marketplace Edition.

7.5.1.1 Model-View-Controller design pattern

The Model-View-Controller design pattern refers to the responsibilities and calling sequence of the Java components used to process a request from the client and return a response document to the client. This pattern is a common pattern used in many Web applications and provides a decoupled approach to separating the functional components. Decoupling of the components in this way allows the development of each component to be performed independently of each other. It also allows for changes to the Web application to occur in a modular way, minimizing the effect on the system.

The responsibilities of these components are as follows:

- The *model* component provides the business logic of the application via model beans and data beans. These beans interact with the necessary data repositories and business rules.
- The *view* provides the page display capabilities and is represented by the JSPs within the application.
- The *controller* provides the decoupling of the interactions between the model and the view. It does not require knowledge of the view component. Controllers are servlets that are responsible for coordinating the appropriate model beans to perform the business logic for the request.

The MVC design pattern is illustrated as logical Java components in Figure 22.

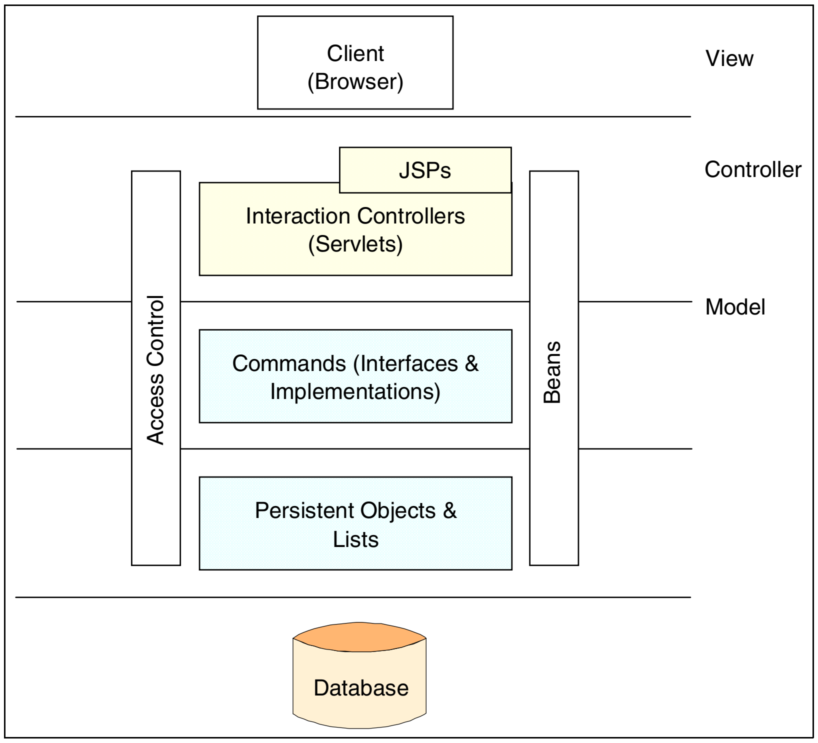


Figure 22. Marketplace Edition Java programming model

7.5.2 e-Marketplace server

The e-Marketplace server represents a conceptual grouping of the processes performed by the Marketplace Edition, such as commencing an auction at a specified time or the processing of an order. The e-Marketplace server runs within the WebSphere Application Server.

Figure 23 shows the relationship between the e-Marketplace server, the application server, external data repositories and integrated buyers/suppliers. For non-integrated buyers and suppliers, a Web interface would exist at the *client browser* level.

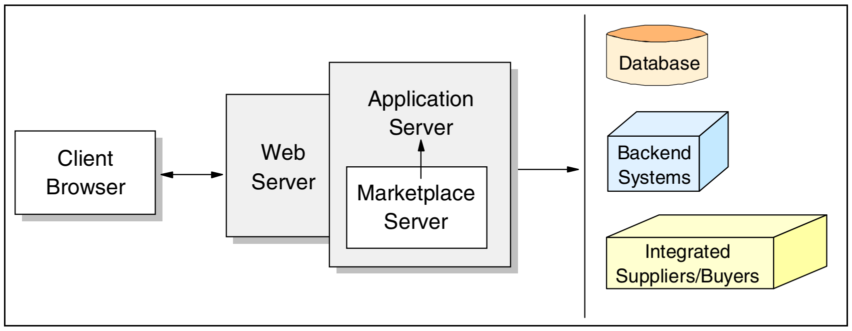


Figure 23. Relationship between the Marketplace server and the application server

Requests for services to the back-end systems, database and integration layers are made through the application server.

7.6 Marketplace Edition application logic

When customizing a Marketplace Edition application, you will build most of it by utilizing the Java-based application development components offered by the Marketplace Edition programming environment. Now that you have an understanding of primary components of the Marketplace Edition, we will now take a more specific look at the application development technologies available to assist us in building an application.

The applications you develop in the Marketplace Edition will usually utilize the following development components:

- Interaction controllers
- JSPs
- Commands
- Command factory
- Data beans
- Data bean manager
- Net.Data
- C++ programming model
- External systems

7.6.1 Interaction controllers

Interaction controllers (ICs) provide the entry point for commonly performed tasks in your Marketplace Edition application. ICs are Java Servlets that operate under, and are accessed from, the WebSphere Application Server.

Java Servlets are protocol and platform independent server-side Java components. They implement a simple request and response framework for communication between the client and the server. The Java Servlet API is used to develop your servlets and offers a set of Java classes that define a standard interface between Web clients and the Web application server. The Servlet API is composed of the following two packages:

- javax.servlet
- javax.servlet.http

An example of a commonly used interaction controller in the Marketplace Edition is the CategoryDisplay controller. If you are familiar with the ICs available in WCS 4.1, you may recall coding URLs such as:

```
http://server/webapp/commerce/CategoryDisplay?merchantid=1&cgrfnbr=10
```

There are many predefined ICs provided by the Marketplace Edition which allow you to easily perform common tasks such as this. You can also write your own ICs to perform any additional processing specific to your application.

Most Java classes for ICs in Marketplace Edition are derived from a special class called EMPIController. This class provides:

- IC level access-control
- Sub-classes implement authorizedPerform()
- Convenience functions to get parameters and provide session management functions.

In addition, all IC invocations are routed through the EMPBaseServlet. By deriving from this class, Marketplace Edition ICs have many core housekeeping tasks performed for them each time a request is made. Some of the tasks performed by the EMPBaseServlet are logging of user information, creating URL parameter lists for passed parameters and extracting user authentication information. The result of this derivation means that each IC is relieved of having to duplicate this functionality since it is performed automatically through subclassing.

Once EMPBaseServlet has completed its processing and passed control through to the derived IC, the IC typically performs the following steps:

1. A request is made to the command factory to obtain the relevant command.
2. The command is configured with data specific to the command usually by calling the command's *setter* methods.
3. The execute() method of the command is called. This executes the business logic associated with the command.
4. Once the business logic has executed, the IC makes *getter* requests to the command to obtain any data made available by the execute() method.
5. Often, the IC will then provide a redirection to a JSP page. However, this is not always the case and it may perform other tasks, such as initiate further commands or send e-mail.

7.6.2 JavaServer Pages

As described earlier in this chapter, JSPs represent the presentation layer or view component of the Model-View-Controller design pattern. Typically, JSPs in the Marketplace Edition implement the view or presentation logic and use various ViewBeans as the basis of the content displayed. Usually, the JSP page does not have any knowledge of the data contained within it and knows simply how to render that data in a suitable format for the user.

When designing your application, it is important to remember that the IBM Application Framework for e-business promotes the thin-client design metaphor, which means that business logic is never contained within the presentation layer of your Web applications, but rather resides on the server. There are a number of benefits of this design technique:

- It allows for separation of development tasks. For example, JSP (GUI) developers do not require a knowledge of the business logic or processes.
- It ensures that the business logic is server-based for easy maintenance.
- It encourages reuse of the server-side components.
- There are no distribution issues such as client JVM versions or browser versions, since the client is simply a formatter of data.

Typically, JSP pages are displayed as a result of redirection by an interaction controller: however, they may also be called directly via URL. This can be seen more clearly in Figure 27 on page 113.

In the Marketplace Edition, you are no longer faced with the dilemma of having to decide whether to use Net.Data or JSPs. All presentation-level development is performed using JSPs.

7.6.3 Commands

Commands contain the business logic required for a particular request. This logic represents a base unit of business logic and may involve database access, sending an e-mail notification or communicating to enterprise applications via a transport such as MQ Series. An example of some Marketplace Edition commands are GetPrice and AddToCart. As part of satisfying the request, commands often call other commands.

Commands in Marketplace Edition inherit from one of the following superclasses:

- ModelCommand

  ModelCommands are used to perform general business logic such as querying and retrieving data from a database.

- ViewCommand

  ViewCommands are used for commands that redirect to a JSP page and they always contain the getPage() method. Both the CategoryDisplay and ProductDisplay commands are derived from this class.

- ErrorCommand

  ErrorCommands are used to return error conditions.

All commands in Marketplace Edition must implement two methods:

isReadyToCallExecute()

performExecute()

The life-cycle of a typical command is as follows:

1. The IC calls EMPCommandFactory.createCommand() to obtain a handle to the appropriate command object.
2. The setter methods of the command object are called.
3. The execute() method of the command is called.

   This calls the EMPCommandTarget which eventually calls the isReadyToCallExecute() and then the performExecute() methods on the command implementation.
4. The getter methods are called to extract the data obtained by the command.

When structuring commands you should, as a general rule, implement the IC/command sequence for any requests involving a *write* to a database. For other read-only tasks, you can typically access the bean directly such as accessing the static beans `CategoryBean` or `PriceBean` from within a JSP.

7.6.4 Command factory

The command factory is responsible for two core interactions with the interaction controller:

1. To facilitate looking up and returning the appropriate command object for a request made to it by an interaction controller. The command factory is *aware* of the available commands.
2. To govern and enforce the security policies associated with each command object. Because each command has authorization information associated with it via the Marketplace Edition administration interface, the command factory permits or denies access to the command based on this information.

7.6.5 DataBeans

DataBeans contain formatted data that can be easily displayed in a JSP page. Typically, DataBeans are instantiated and populated via a user request for data.

DataBean objects are derived from the abstract parent class DataBean, which governs the mandatory methods required to be implemented for all derived DataBean classes. A useful feature of DataBeans is their meta-data properties which describe the bean's contents and structure.

DataBeans are classified into one of two groups:

- StaticDataBeans

  The StaticDataBean is implemented when the returned data is always "known" or is static. An example of a StaticBean is the CategoryBean which is guaranteed to always returns the same data - name, description, thumbnail image and so on.

- DynamicDataBeans

  The DynamicDataBean is used where the returned data can yield variable results such as extracting the attributes of completely unrelated product types such as a piano and a Palm Pilot. The DynamicDataBean provides a unified interface to explain the data it contains.

7.6.6 DataBean manager

The DataBean manager is responsible for selecting the appropriate command to populate a request for a particular DataBean. Once it has identified the command, it executes the command that subsequently instantiates and populates the DataBean.

7.6.7 Net.Data

While not guaranteed to be the case for subsequent WCS releases, the use of Net.Data macros in the WebSphere is still a prominent part of the order subsystem in Marketplace Edition. However, there is now support for interacting with the order subsystem through the Java-based methods described previously.

Net.Data or JSPs?

Should you chose JSPs or Net.Data macros for implementation of your presentation logic? As a general rule in the Marketplace Edition, you should use the Java-based programming model where possible. For example, any new pages that you design for catalog navigation, product display, or custom functionality added to the site should be developed using JSPs, interaction controller, and JavaBeans. This provides the most compatibility for subsequent releases of WebSphere Commerce Suite products. However, the default pages presented in the order process are created via Net.Data macros, so to modify the presentation or functionality of these pages, you will have to modify the existing Net.Data macros (files with a .d2w extension).

Some additional considerations regarding the use of Net.Data versus JSPs are well summarized in *User-to-Online Buying Pattern using WebSphere Commerce Suite V4.1,* SG24-6156. The considerations raised in this book are:

- At this time JSPs might be slower than Net.Data macros, but there is a patch, that enables JSP caching with the WebSphere Commerce Suite. Once the page is cached, upon first usage, any further request will be served faster.
- Net.Data macros are based on a scripting language and interpreted by a CGI application. Net.Data generated pages are cached by the WebSphere Commerce Suite.
- JSPs are Java-based pages, parsed by the server and compiled into servlets. This approach accesses the latest technology of server-side programming. Future releases of the WebSphere Commerce Suite will fully implement Java Servlets and JSPs for all pages supported by Net.Data.

- A designer tool for Net.Data macros does not ship with the WebSphere Commerce Studio.
- The WebSphere Commerce Studio provides a WYSIWYG designer for JSP generation.
- With JSP pages you can use pure Java code. With this capability you can access the Java-based application server and do further developments on your system.
- Net.Data does not allow you to access the Java-based application server. You have to develop your own C++ modules.

7.6.8 C++ programming model

The C++ programming model has been available since early Net.Commerce days, and remains a major component of the Marketplace Edition. Essentially, the C++ programming model gives us the ability to further customize the default functionality provided with the Marketplace Edition. By developing our own commands and overridable functions, we can *override* this default behavior with our own logic.

In general, the process observed by Marketplace Edition is:

COMMAND calls a TASK which calls an OVERRIDABLE FUNCTION

These three components are described below:

- Commands

  Typically, commands interact with tasks in the sense that they delegate well-defined elements of business logic to a particular task which in turn maps to a particular overridable function. However, commands are not only limited to this construct and may call other commands.

- Overridable functions (OFs)

  OFs represent C++ classes that allow us to write customized business logic for a given task. OFs are invoked by the overridable function manager.

  An example of an OF is presented in Figure 24. This example is for WCS 4.1 and illustrates how to provide a discount for shopping orders that exceed a particular amount. It is implemented for the GET_ORD_PROD_TOT task.

```
virtual bool Process(const HttpRequest& Req, HttpResponse& Res,
NC_Environment& Env)
{
const doubleDISCOUNT_PERCENTAGE  = 10.0;
const doubleDISCOUNT_TOLERANCE   = 200.0;

String      SqlStatement;
Row         SqlRow;
int         sql_code             = 0;
double      order_prod_total     = 0.0;
double      disc_order_prod_total= 0.0;
bool        bRc                  = false;

// Explicitly passed in for the GET_ORD_PROD_TOT process task
static const StringWithOwnership _ORDER_REF_NUM("ORDER_REF_NUM");
static const StringWithOwnership _CURRENCY("CURRENCY");
static const StringWithOwnership _PRODUCT_PRICE("PRODUCT_PRICE");
```

Figure 24. Example overridable function

```
// Get the Order Ref number from the Environment
const String& OrderRefNum = (const String&) *Env.Seek(_ORDER_REF_NUM);

// Select just the product total for each product in the order
SqlStatement.Clean();
SqlStatement   << " SELECT oyprtot"
               << " FROM   orderpay"
               << " WHERE  oyornbr=" << OrderRefNum;

SQL S1(*DataBaseManager::GetCurrentDataBase(), SqlStatement.nnc_str());
if (S1.getNextRow(SqlRow) == ERR_DB_NO_ERROR)// should only be one row
{
   // Get the total value of the order
   SqlRow.getCol(1).getVal(order_prod_total);
}
else
{
   sql_code = S1.getSQLrc();
   error << indent << "ERROR Code " << sql_code << endl;
   return(false);
}

// Compute the discount if the order is over the appropriate amt
if (order_prod_total >= DISCOUNT_TOLERANCE)
{
   disc_order_prod_total = order_prod_total - (order_prod_total *
(DISCOUNT_PERCENTAGE / 100));
}

// Update the PRODUCT_PRICE explicit output parameter
String& ProductPrice = (String&) *Misc::CheckEnvironmentVariable(Env,
_PRODUCT_PRICE);

ProductPrice = disc_order_prod_total;
return(true);
}
```

Figure 25. Example overridable function

- Tasks

  Unlike commands and overridable functions, tasks are not actual pieces of code but rather a specification for the communication between a command

and an overridable function. They define the inputs and output that a command and overridable function must adhere to.

Using the system administration console, administrators can map different overridable functions (instances of business logic) to a particular task for a given organization within the e-Marketplace. Essentially, one command is mapped to one or more tasks which can execute different business logic depending on the organization.

Figure 26 demonstrates how these components work together in the execution environment.

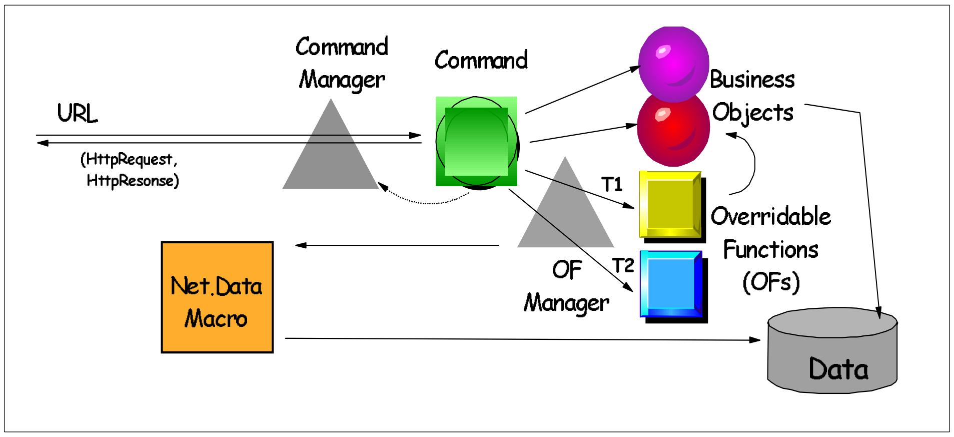


Figure 26. Interaction between command and overridable functions

Many modifications have been made to the C++ programming model to support the Marketplace Edition. These modifications are quite low-level and beyond the scope of discussion in this chapter.

The reference *Commands, Tasks, Overridable Functions and Database Tables,* provided as part of the WebSphere Commerce Suite, Marketplace Edition for AIX software is an excellent resource for understanding these concepts more thoroughly.

7.6.9 External systems

A critical element in our Marketplace Edition application is integration. This integration will often occur between:

- The e-Marketplace and the e-Marketplace back-office systems

- The disparate members of our e-Marketplace - the systems of buyers and suppliers for example.

Integration becomes critical as we introduce the integrated runtime topologies such as *subset 2* and *subset 3* discussed in Chapter 3, "e-Marketplace runtime topology" on page 31. When considering these integration points, we need to answer questions such as:

- Should we attempt real-time interfaces or batch mode operations, or implement via a Web client interface?
- What is the speed of access to these systems?
- Are there connectors available to these systems or do we have to develop them?
- What is the transport mechanism and is message delivery guaranteed?
- Should we use a simple point-to-point connection or a publish and subscribe message broking metaphor?
- Is there the need to transform messages between these systems?
- What is our approach to the integrity of the "unit of work" where the tasks in the unit could be distributed over various systems and locations?

These issues highlight just a few of the complexities when dealing with external systems.

7.6.10 Putting it together

The diagram presented in Figure 27 shows the high-level relationships between the Java application components described in the previous sections.

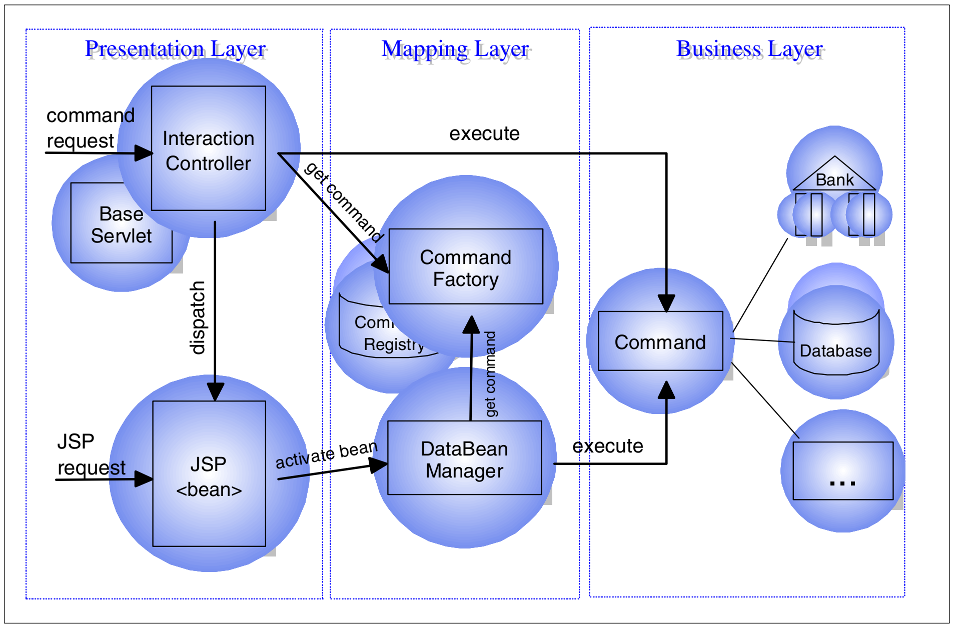


Figure 27. Application components within Marketplace Edition

The entry point into the runtime system is either a WebSphere Commerce Suite command URL, which invokes an interaction controller, or an explicit JavaServer Page (JSP) request. This gives the Web site creator the ability to use an interaction controller (IC) where appropriate or just wire JSP pages together when an IC is not needed. As mentioned previously, an interaction controller/command pair should always be used for requests that write to a database and also for some well defined read-only interactions such as CategoryDisplay or ProductDisplay where the JSP to display contains dynamic data.

If a URL request is made that calls an interaction controller, the EMPBaseServlet (called Base Servlet in the figure) is available to do some necessary setup that all ICs need. The interaction controller will then use the command manager to look up and instantiate the appropriate command, set it up, and run it. In the case of a call to an explicit JSP page, every JavaBean on the page needs a way be filled with the data it represents. This is done by calling the BeanManager, from which you request to have the bean activated. The BeanManager can be thought of as a generic internal interaction controller that uses the command manager to map requests for bean data to the commands that know how to fill them.

7.7 Session management

We can define a session as a series of requests originating from the same user, and the same browser. Because HTTP is a stateless protocol by design, various techniques have been developed to maintain the application state across multiple HTTP requests originating from the same user.

The two primary methods are:

- Cookies
- URL rewriting

The WebSphere Application Server implements the HttpSession API that shields the programmer from the complexities of implementing these techniques. Also, the WebSphere Administrator has various configuration options for session management. These configuration options can influence the application behavior, performance, and failover capabilities and as such, should be given due consideration in the design of your applications.

7.7.1 Cookies and URL rewriting

The WebSphere Application Server can be configured to use cookies, URL rewriting, or both to maintain sessions across multiple HTTP requests. With both mechanisms WebSphere creates a session ID to identify a session uniquely. With cookies, this session ID is sent back to the browser as a field inside the cookie. With URL rewriting, the session ID is appended to the URL and sent back to the browser. During subsequent requests the browser sends the session ID back to the server as part of the cookie or the URL. The server is responsible for retrieving this session ID from the HTTP request and using it to obtain the proper HttpSession for this user.

Early in the high-level design phase it is important to decide whether your application should be developed to support cookies, URL rewriting, or both. Such a decision should be made based on the demographics of the end users of the system. Some users configure their browsers to not accept cookies. If you suspect this may be the case, consider supporting the URL rewriting option. For the majority of applications we recommend using only cookies. The WebSphere Application Server administration console provides a simple way to configure your applications to support either or both of these session management techniques. However, in order to maintain a session state using URL rewriting, the underlying servlets and JSPs must be coded so that every URL you send back to the browser is encoded with the session ID. This can be achieved by using the following techniques:

- Encode all URLs in servlets and JSPs

This can be done using the `HttpServletResponse.encodeURL(String url)` method. This method appends the session ID to the URL that is passed as an input parameter. For example, if you have a servlet that does not support URL rewriting and has the following code:

```
out.println("<a href=\"exampleLink.html\"> Example Link <a>");
```

Then, in order to support URL rewriting, replace all such references to URL links as shown below:

```
out.println("<a href=\"");
out.println(response.encodeURL("exampleLink.html");
out.println("\"> Example Link <a>");
```

- Use `HttpServletResponse.encodeRedirectURL(String url)` to encode all redirects.
- Do not include links to parts of your Web applications in plain HTML files.

In essence, to maintain a session state using URL rewriting, every page that the user requests during the session must be converted to JSPs or servlets. And all links inside such servlets and JSPs must be encoded using encodeURL(). These requirements are necessary for URL rewriting to maintain a session state, since all HTTP requests made by the user must have the session ID appended to the requesting URL. From this discussion, it is clear that the decision to support URL rewriting impacts the code development significantly. This decision also has certain performance implications, since all display pages, including static pages, have to be converted to JSPs or servlets. This adds unnecessary runtime processing. This also means that all static pages that could have been hosted by an information Web server now need to be moved to the Web application server node since all pages have to be converted to JSPs or servlets. Therefore it is important to decide early in the high-level design whether you plan to support URL rewriting.

7.7.2 Session persistence and clustering

Web application availability and performance can be increased by adding duplicate Web application server nodes to the runtime topology. This is achieved by using a load balancer to distribute Web requests across multiple Web application servers. In such a scenario, if one Web application server fails, the load balancer would recognize this event and forward all the subsequent requests to the remaining Web application servers, which increases the overall availability of Web applications. Performance should be improved by distributing the load across multiple machines.

The above scenario can be extended to provide failover support. This is achieved by enabling WebSphere Application Server session persistence and session clustering. When session persistence is enabled, the WebSphere Application Server stores all session data in a JDBC-compliant relational database such as DB2 or Oracle. This is achieved by inserting the session data (name-value pair) into the database as a result of an HttpSession.putValue() method and retrieving the same from the database as a result of HttpSession.getValue() method.

The session persistence is automatically managed by the WebSphere Application Server. Application programmers need not write any special code for this session persistence to occur. However all objects that are being inserted into the session pool must implement the *serializable* interface. Session clustering is a mechanism where more than one instance of the application servers share a common session pool. Essentially, a cluster is the binding of two or more application servers that reside on separate nodes. This allows servlets to execute on any one of these nodes and have access to session data that was created by another node. The WebSphere Application Server exploits its session persistence feature to implement session clustering. Therefore, in order to enable session clustering, session persistence must be turned on. Under this configuration, multiple application server nodes would share the common session database. This allows for session data created by one application server node to be accessed by another application server node during subsequent interactions. All changes to session data are committed to a common session database upon the completion of servlet execution. The session data is still accessible regardless of the failure of an individual node. This provides for complete failover support.

In designing systems that exploit session persistence and clustering features, we provide the following guidelines:

- Session persistence is implemented using a generalized persistence mechanism in order to allow for various types of information to persist in the session pool. Storing large amounts of data in such a generalized session pool could result in performance degradation. Hence the session pool must be used only to store data that is essential during subsequent transactions.
- All objects that must be propagated across the cluster along with the session and must be serializable. We recommend implementing the *serializable* interface for all objects that you anticipate being stored in the session pool. This allows for an easy transition of your applications to a clustered environment.

Further details on session management issues can be found at:

`http://www.ibm.com/software/webservers/appserv/doc/v30/se/web/doc/begin_here/index.html`

7.8 Application performance

While most performance gains are obtained at the infrastructure level, some areas for performance improvement can be made at the application level and should be considered when you are designing applications. One area of concern to Net.Commerce programmers is how caching is implemented when building a site with JSP pages.

7.8.1 JSP caching

In Net.Commerce and optionally in the WebSphere Commerce Suite 4.1 (if using Net.Data rather than JSP), there were a number of options available to the site administrator regarding caching of HTML pages. This provided for significant performance increase in page load times because a typical user request for catalog data has usually be requested previously and therefore resides in the cache. .

When writing applications with the Marketplace Edition, they will usually be developed using the Java programming model - JSPs, servlets and JavaBeans. However, the result of this is that page content is dynamically generated for each response document issued from the server to the client. This means that the same product page output requested potentially millions of time by users must be regenerated each time a request is made for it.

7.8.2 Integrated buyers and suppliers

The integration of buyers and suppliers is an important aspect of an e-Marketplace and one that you should consider in your application design. The design decisions in this area could have a large impact on the site performance, particularly if the e-Marketplace implements real-time integration of these systems.

This book does not deal with the integration of buyers and sellers.

7.9 Security

An important aspect to a successful e-commerce site is security. Your customers will be concerned with the security of their personal information as it is transmitted across the Internet and as it is used throughout your order processing environment. In addition, you should be concerned with securing your information assets and systems.

The WebSphere Commerce Suite provides features to help you implement your security strategy. The security topics discussed briefly in the following sections include:

- Authentication
- User registry
- Access control

The WebSphere Commerce Suite supports client certificate logon as a security mechanism, protecting both the Web site and customer. During the configuration of the WebSphere Commerce Suite, you can specify either basic authentication (a user ID and password) or X.509 certificate authentication, which is an electronic certificate arranged through an external certificate authority to handle electronic authentication of X.509 certificates.

The security concept depends on the business and the security directives of the company.

7.9.1 Authentication

In order to conduct business online, two (or more) parties typically interact. The concept of authentication is the ability for these parties to trust the identity of the other (in other words, each party is who they say they are). The WebSphere Commerce Suite provides the following two modes of authentication:

- Basic

  If you select to use the basic mode of authentication (the default mode), users have a logon ID and password registered in the WebSphere Commerce Suite server user registry.

- X.509

  The Marketplace Edition supports client certificate logon as a security mechanism, protecting both the site and the customer. The X.509 certificate supplements basic authentication for customers entering a site.

  A customer holding this certificate can access a secured WebSphere Commerce Server site, which has been enabled for client certificate authentication.

  Before you can begin using X.509 certificates, you must arrange for a trust relationship with external certificate authorities to handle electronic authentication of X.509 certificates. Certificates are provided by certificate authorities, which can be found on the Web.

7.9.2 User registry

To interact with your site, users may need to register with the site. Unlike the WebSphere Commerce Suite, the Marketplace Edition enforces the use of Lightweight Directory Access Protocol (LDAP) in user registration and subsequent authentications to the e-Marketplace.

Lightweight Directory Access Protocol (LDAP)

LDAP is a client/server protocol for accessing a directory service. It was initially used as a front-end to X.509, but can also be used with stand-alone and other kinds of directory servers. LDAP can be used as a centralized information repository to support information sharing among various clients. LDAP provides a standard way to authenticate users and manage information. This allows you to create a solution in which a user can register.

LDAP is implemented within the Marketplace Edition by using the IBM SecureWay Directory product.

7.9.3 Command access

Access to commands in Marketplace Edition is governed by the Administration interface. For commands, you need to consider who is permitted to execute the command - all buyers, buyers with particular geographic or demographic constraints, or administrators. In the first case, you can forgo access control altogether. In the other two cases, your command must implement access control properly.

All commands issued through a URL can contain the optional parameter `merchant_rn`. If this parameter is specified, its value is the merchant reference number. The behavior for this optional parameter is one of the following:

- The WebSphere Commerce Suite separates commands into two different categories: commands that require resources specific to a merchant (for example, the `CategoryDisplay` command) and commands that do not depend on any merchant resources (for example, the `AddressForm` command). The command security form within the WebSphere Commerce Suite allows administrators to specify SSL enablement for a command and user authentication; however, due to the internal distinction of WebSphere Commerce Suite commands, a command such as `CategoryDisplay` would produce the correct merchant-specific command security behavior, whereas a command such as `AddressForm` would give the mall level command security behavior. To ensure that the merchant-specific command security behavior is always applied, include the `merchant_rn` parameter and value.

- If a command such as `ProductDisplay` has its command security set such that authentication is required, then if the command is executed via a non-authenticated mode, the user is directed to the merchant-specific LOGON_ERR exception task page or the mall level LOGON_ERR exception task page. To direct users to the merchant-specific LOGON_ERR page, include the `merchant_rn` parameter and value. You can set up the merchant-specific LOGON_ERR task page by using the task management form within the WebSphere Commerce Suite Administrator.

The Web is an open environment. Users can observe URLs that are issued and play with them. They can modify them to try to confuse the system, change the values for some parameters, re-issue the same command multiple times, or issue a given command in a different context. Finally, always remember that only trivial systems can be made absolutely safe, and although we have documented some issues, there are potentially many others that we are unaware of at this time. By having a solution that is well designed however, you could be in a position where you could respond quickly to a security exposure, and implement a fix rapidly.

In summary, here are some known issues that you can guard against:

- Your command must have integrity
- Cross reference its parameters
- Have proper access control defined for it
- Guard against common attacks

7.9.4 Integrity

Integrity means that your command needs to be executable - regardless of the state of the system. Although most of the time your command will be designed to work in concert with others, being on the Web means that anyone could issue a URL for your command at any time. Always make sure that your environment is in the proper state, or that you can bring the system back into a proper state, before continuing. For instance, the `OrderDisplay` command will always recompute and lock the order completely before allowing a view to display it. You could execute this command at any time during the shopping process, and the result would always be correct. The `OrderProcess` command can only be called if the order has been properly locked first, and if it is called in any other condition, an error is returned.

You should also be prepared for the *repeat* scenario in which the same command is called more than once in a row: it is always possible for a user to use the Back button, and re-issue the same command a second, third or fourth time. Be sure to well define the semantics of such an action. For instance, calling `OrderProcess` a second time causes an error, whereas calling `OrderItemAdd` a second time simply adds the product again to the order.

7.9.5 Cross referencing

Cross referencing means that all the data you receive should be checked against each other. For instance, your command could receive a product number, and a merchant number, and assume that the product mentioned belongs to the merchant passed. Do not assume; make sure. When you look up the product from the database, simply use the merchant ID to further constrain the query. If the two parameters do not agree, then you have a system error and you can simply return false. In this case either the calling page was not coded properly, or someone is playing around with the site, which means that it is OK to return the system error page.

For another example, consider a ticketing system where particular events are modeled using categories. When a user requests tickets, you might have a situation where your back-end system expects the tickets to all belong to the same event (that is, you cannot order tickets for multiple events at the same time). In this case, you have to cross reference the products you receive to make sure they belong to the same event. Never think that the set of pages put together to access your site is the only way that a user can call your commands.

Chapter 8. Application development guidelines

In this chapter we discuss proven guidelines for managing a successful Internet application development project.The application development project road map is a set of work products that will help in the documenting of the application design and help in the management of the application development.

8.1 The application development project road map

The application development project road map provides a structured way to run a successful development project. It will not work for every type of application development, but we found that it works well on developing complex e-business and Internet solutions.

The project road map is a set of working documents that answer questions about the actual Internet applications to be created. These documents will help you in gathering requirements, designing the applications that adhere to the customer's requirements, tracking the development of the source code, and tracking the testing and deployment of a new Internet application.

8.1.1 Project plan

The project plan should list all the applications that will be developed, with a start date and end date for each application.

8.1.2 Business requirements

The business requirements are typically gathered from the customer, including an assessment of the risks that are involved in doing an Internet development effort.

8.1.3 Application requirements

The application requirements are based on the business requirements and customer interviews and meetings.

8.1.4 Use cases model

The use case model describes the functional requirements of the system that is being developed. Use case diagrams use graphical symbols and text to specify how users in specific roles will use the system. They do not describe how the system works or its internal structure and logic. A sample use case diagram is shown in Figure 28

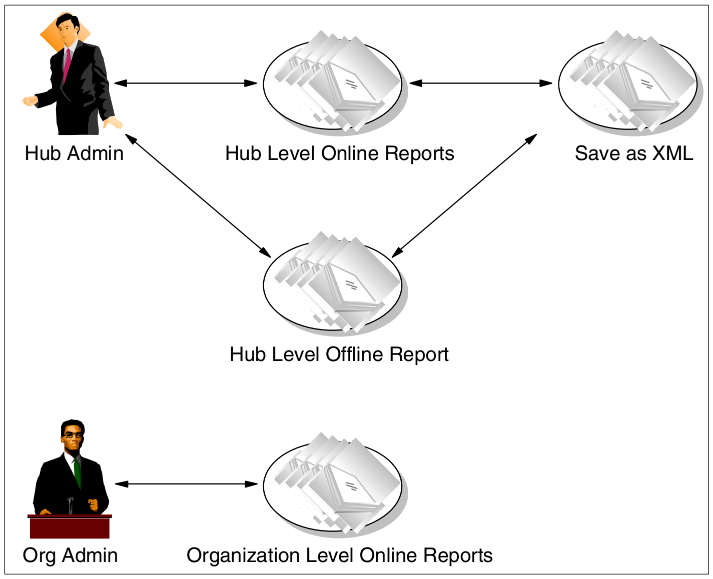


Figure 28. Use case diagram

A use case diagram includes:

- Actors (name, description, status, subclass, superclass and associations).
- Use cases (number, subject area, business event, name, overview, preconditions, description, associations, inputs, outputs traceable to, usability index, and notes).
- Communication associated between actors.
- Relationships between use cases.
- Outcome of processes and error messaging.
- Use case scenarios (number, termination outcome, description, and notes).
- Problem domain concept definitions.
- System decision table and decision map.
- Event table and map.
- System sequence diagram.

A use case diagram establishes the boundaries of the proposed software system and fully states the capabilities to be delivered to the user. Together with the business requirements and the application requirements it fully documents the functionality of the application to be delivered.

8.1.5 Conceptual diagram

A conceptual diagram shows how each part of the application will interact with each other. Typically an application architect will use a modelling tool that uses UML (Unified Modelling Language) to produce this document. Figure 29 shows an example conceptual diagram.

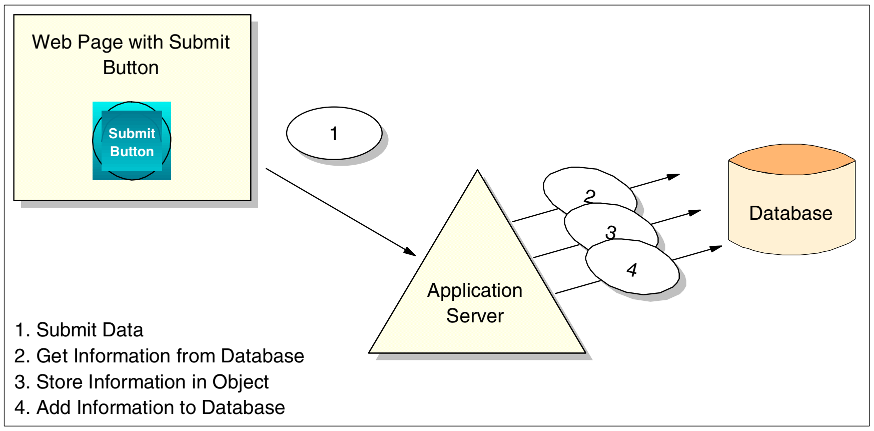

Web Page with Submit Button
Submit Button
1
2
3
4
Application Server
Database
1. Submit Data
2. Get Information from Database
3. Store Information in Object
4. Add Information to Database

Figure 29. Conceptual diagram

8.1.6 Class diagram

A class diagram interprets business, user and system requirements and develops an overall model of what is expected of the software. A good class diagram will achieve the following goals:

- Determine the system boundaries in the context of the objects that make up the system.
- Help in allocating the workload between the objects and the applications.
- Create a starting point for adding new layers to an existing system, or adding new layers to the current application design.
- Help the overall verification of the analysis and design of the system.
- Enhance the quality of the design of the system by making the system more manageable.

- Improve the understanding of the design and system intent by the development team and key customer personnel.
- Allow the actual system construction to be done both iteratively and incrementally.

The class diagram is a structural representation of the software objects, and the relationships between them, that are used to develop a system. Class diagrams should include detailed descriptions of each of the components. If the tool that you are using does not embed the description for each object in your diagram, these descriptions should be documented elsewhere. The class descriptions should include the number of instances for each class, the average size of an instance, and the association volumetrics.The design used for the conceptual view in the class diagram should remain technology neutral. It contains all the classes found in the problem domain. The class diagram is part of the logical design phase or “the how to” phase of the application design. Factors such as concurrency and distribution, coordination and sharing, transactions and persistence, user interface capability, and system interfaces such as communication are also taken into account when developing the class diagram. An example class diagram is shown in Figure 30 on page 127.

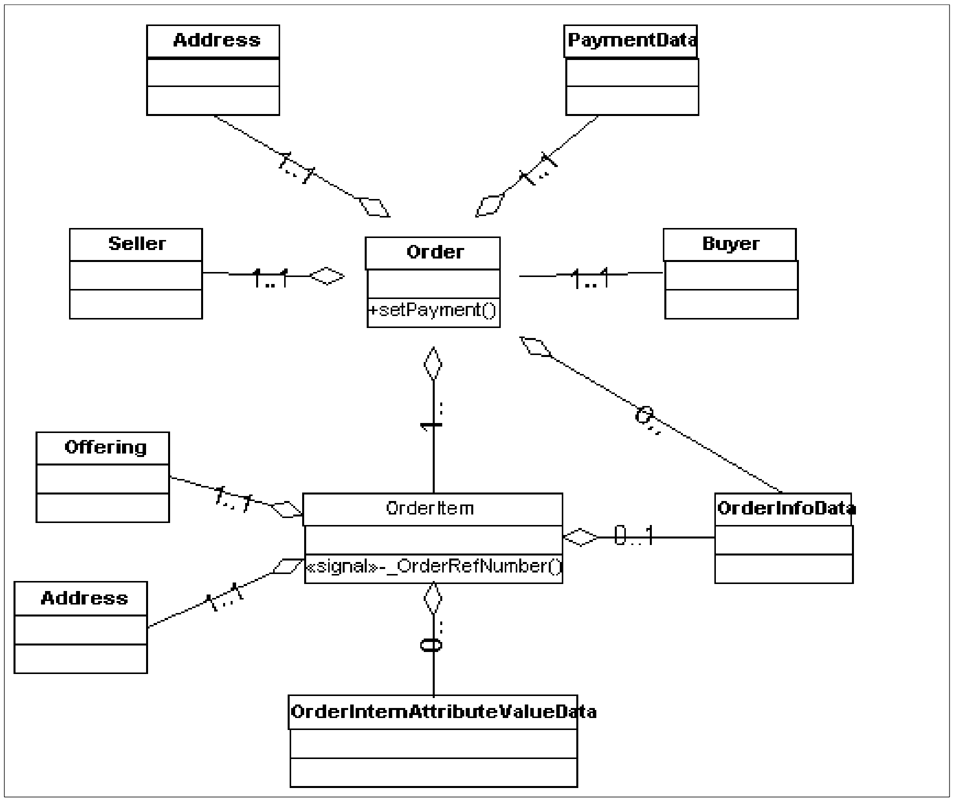


Figure 30. Class Diagram

8.1.7 Code review questionnaire

Code review questionnaires provide questions that you ask the application developer to verify that the application was developed in accordance with the customer's requirements and adheres to the development standards.

8.1.8 Detailed test plan

The application test plan is used to organize application testing after the initial development is complete. A good application test plan will have test cases for each of the components of the application. Most of the time the application developer and application architect will provide input to help create the test plan.

8.1.9 Deployment plan

The deployment plan provides a detailed schedule of events, expected project duration, persons responsible, and event dependencies required to ensure a successful cutover to the new system. The plan should minimize the impact of the deployment on your team, production system, and overall business routines. A good deployment plan anticipates most of the issues you will face during deployment, where you will be vulnerable to the smallest unforeseen glitch. The time you spend trying to obviate such problems is time well spent and will be appreciated when the deployment team is in the midst of the installation.

8.1.10 Maintenance plan

The maintenance plan outlines who will be responsible after the new application have been developed, tested and deployed. This plan will show who will be adding fixes and patches to the new application after it has been deployed, and gives the users a point of contact to submit fixes and bug reports.

8.1.11 The application development project road map reports

These reports help the project manager and the application architect manage and report the status of the development effort to the customer.An example of an application development project road map report is shown in Figure 31

#	Application Name	Priority	1. Business Req.	2. App. Req	3. Use Case	4.Conpetual Diag.	5.Class Diag.	6. Code Dev.	Comments	Developer
1	User Registration	H	DONE	20%	0%	0%	0%	0%	will be late	
2	Group Registration	H	DONE	30%	10%	0%	0%	0%		
3	Buy Product	M	DONE	50%	20%	0%	0%	0%		
4	Auction Product	L	DONE	10%	0%	0%	0%	0%		
5	Offering	M	DONE	35%	0%	0%	0%	0%		
6	Customer Polls	H	DONE	10%	0%	0%	0%	0%		

Figure 31. Sample application development road map report

The report shows the titles of the applications that will be developed, the priority order in which the applications should be developed, the percentage completed of each of the working documents, and an area for comments to note any challenges that the development team is facing that could hold up the development effort.

8.2 Development tool set

A good set of development tools can help the developer quickly write, compile, and debug code. WebSphere Studio is a good tool when developing Web pages and JavaServer Pages (JSP). When developing Java applications for the e-Marketplace, VisualAge for Java is a good choice for a Java IDE (Integrated Development Environment). VisualAge for Java provides a project browser, object browsers, and an integrated debugger.

8.3 Source control

When a development project has more that one developer it is a good practice to use a source control tool, which provides a way to version the application code and a way for several developers to merge their code together. Source control tools will also provide a centralized place for all development code, which makes backups and deployment easier. Some good source control tools are PVCS, Rational Clear Case, and the VisualAge for Java Repository.

8.4 Testing

Testing is one of the most important parts of the development process. The testing methods described here are produced in a separate process, possibly by different team members in parallel with, or after, the coding phase. There are five kinds of tests that we will discuss:

- **Integration testing** verifies proper execution of application components and does not require that the application under test interface with other applications. Communication between modules within the subsystems is tested in a controlled and isolated environment within the project.
- **Usability testing** ensures that the final product is usable in a practical, day-to-day fashion. Whereas functional testing looks for accuracy of the product, this type of test looks for simplicity and user-friendliness of the product.
- **System tests** verify proper execution of the entire application including interfaces to other applications. Both functional and structural types of tests are performed to verify that the system is functionally and operationally sound.
- **Stress testing** processes of a large number of transactions through the system in a defined period of time in order to measure the performance characteristics of the system under peak load conditions. Stress factors may apply to different aspects of the system such as input transactions, report lines, internal tables, communications, computer processing capacity, throughput, disk space, I/O and so on. Stress testing should not begin until the system functions are fully tested and stable. The need for stress testing must be identified in the design phase and should commence as soon as operationally stable system units are available.
- **Performance testing** ascertains whether the system meets the desired level of performance in a production environment. Performance considerations may relate to response times, turn-around times

(through-put), technical design issues, and so on. Performance testing can be conducted using a production system, a simulated environment, or a prototype. Attention to performance issues (for example, response time or availability) begins during the design phase. At that time, the performance criteria should be established. Performance models may be constructed at that time if warranted by the nature of the project. Actual performance measurement should begin as soon as working programs (not necessarily defect-free programs) are ready.

Chapter 9. System management guidelines

In this chapter we focus on the activities involved in systems management and security. This chapter is organized as follows:

- General systems management guidelines
- Product specific systems management guidelines
- Security guidelines
- Backup and recovery guidelines

9.1 General systems management guidelines

Once the application development, testing and user acceptance of your e-Marketplace application is complete, you will have to deploy and manage the application in a production environment, which will require early planning.

We have categorized a set of post-implementation activities that have to be performed on a routine basis as a part of system management. The system management activities usually involve:

- Application management
- Performance monitoring
- Availability management
- Security management
- Disaster recovery
- Operating system and network administration
- Asset management
- Software distribution
- Problem reporting
- Change management

Each of these activities requires highly specific skills and professional experience to perform them competently. Besides the skills factor, you will also have to decide on a set of tools to perform the system management activities.

Beyond the technical challenge that systems management poses, there is also the added pressure from management. In many situations, you will be bound by service level agreements (SLAs), which typically cover system

© Copyright IBM Corp. 2000

availability hours, system utilization and problem resolution response time. These measurements will be collected, tabulated and reviewed on a regular basis by management to ensure accountability and a well-maintained system. Thus, you will also require reporting tools to facilitate the SLA review.

We recommend that you start planning early. Incorporate system management requirements in the early phases of your design, since what you design will affect how you eventually manage it. Conversely, the tools available to manage your system also affect your application design.

In this redbook we consider the key system management activities related to example in the WebSphere Commerce Suite, Marketplace Edition for AIX, such as:

- Application management
- Performance management
- Availability management
- Security management
- Disaster recovery

9.2 Product-specific systems management guidelines

An e-Marketplace application created using WebSphere Commerce Suite, Marketplace Edition for AIX is a combination of HTML pages, JSPs, servlets, and JavaBeans. These resources will be deployed and managed in the WebSphere Application Server and the WebSphere Commerce Suite, Marketplace Edition for AIX environment. There are also other components in an e-Marketplace application, such as the Web server, the DB2 UDB databases and LDAP directory. You would also need to create reports for your e-Marketplace site activities.

Software products needed for creating and maintaining an e-Marketplace are covered in detail in Chapter 10, “Marketplace Edition overview” on page 195. In this chapter we will provide some guidelines on system management tasks for these products, including:

- WebSphere Application Server
- Site Analyzer
- WebSphere Commerce Suite, Marketplace Edition for AIX
- Secure Way Directory
- IBM HTTP Server

- IBM DB2 UDB

9.2.1 WebSphere Application Server Administrative Console

As mentioned earlier the WebSphere Commerce Suite, Marketplace Edition for AIX contains components such as servlets, JSPs, JavaBeans and HTML pages. These components are considered Web resources to the e-Marketplace Web application. Web resources that make up a Web application require a Web application server environment to be deployed, configured and executed. The WebSphere Application Server provides this environment. For optimum availability and performance these Web resources need to be managed and maintained by the WebSphere Application Server.

The WebSphere Administrative Console is used to manage, deploy and configure the Web resources. In our discussion, we will refer to some WebSphere specific terms shown in Table 8.

Table 8. Description of WebSphere terms

WebSphere Terms	Description
Web resources	Refers to servlets, JSPs, JavaBeans and HTML pages.
Web application	Application consisting of Web resources.
Enterprise application	Application consisting of Web applications and EJBs.
Application server	A JVM runtime service that handles user requests from enterprise and Web applications.

The WebSphere Application Server uses core underlying services such as the servlet engine, EJB engines, security application and cluster models residing in the application server. These services manage the Web resources and applications that are hosted by the WebSphere Application Server. Depending on application requirements, you may be managing a single stand-alone application server or multiple application servers that support failover capabilities.

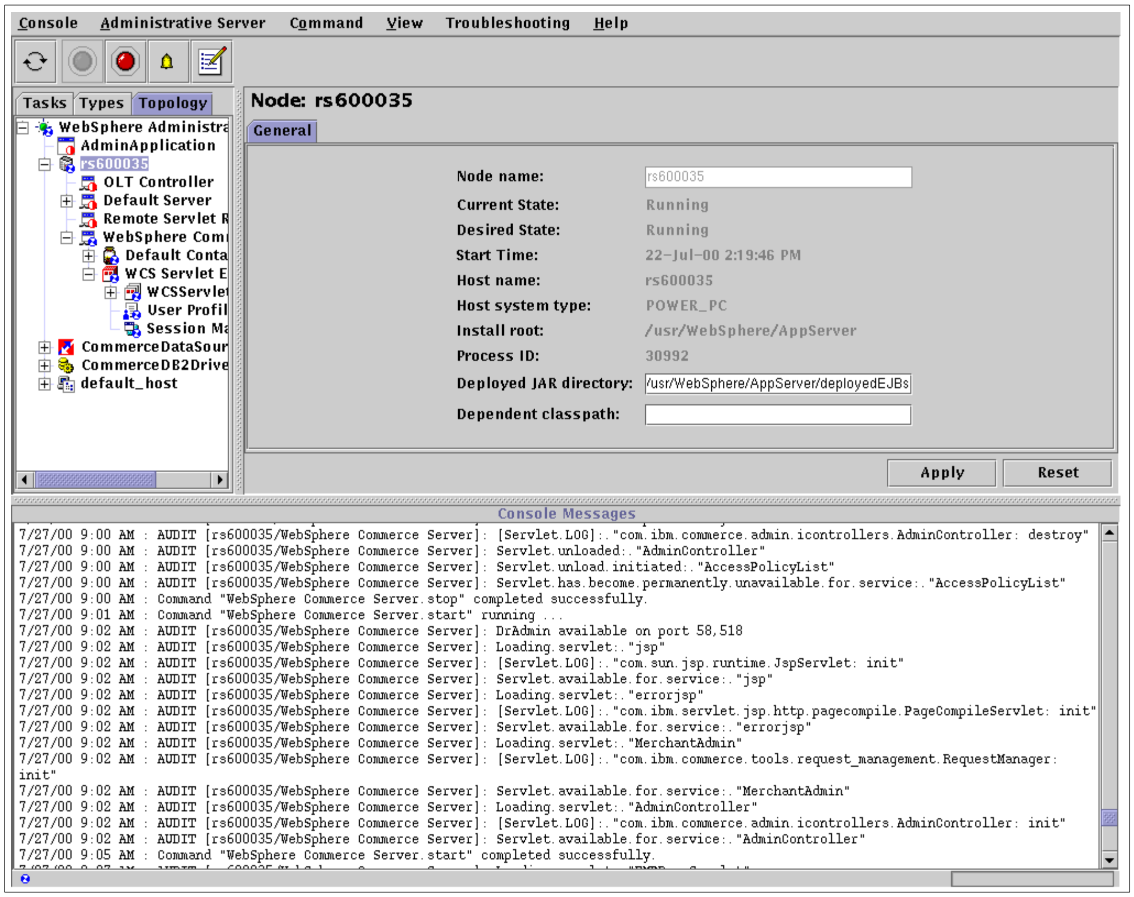
Using the WebSphere administrative model, you can manage, combine, secure, and distribute servlets, enterprise beans, JSP files, and Web pages.

9.2.1.1 Administrative Console and terminology

The WebSphere Application Server provides administrators with a single system view of applications and resources through the WebSphere

Administrative Console. Resources can be administered locally or remotely by the administrator using the WebSphere Administrative Console.

Figure 32 displays the WebSphere Administrative Console.



Node: rs600035

Node name:	rs600035
Current State:	Running
Desired State:	Running
Start Time:	22-Jul-00 2:19:46 PM
Host name:	rs600035
Host system type:	POWER_PC
Install root:	/usr/WebSphere/AppServer
Process ID:	30992
Deployed JAR directory:	/usr/WebSphere/AppServer/deployedEJBs
Dependent classpath:	

```
7/27/00 9:00 AM : AUDIT [rs600035/WebSphere Commerce Server]: [Servlet.LOG]:."com.ibm.commerce.admin.icontrollers.AdminController: destroy"
7/27/00 9:00 AM : AUDIT [rs600035/WebSphere Commerce Server]: Servlet.unloaded:."AdminController"
7/27/00 9:00 AM : AUDIT [rs600035/WebSphere Commerce Server]: Servlet.unload.initiated:."AccessPolicyList"
7/27/00 9:00 AM : AUDIT [rs600035/WebSphere Commerce Server]: Servlet.has.become.permanently.unavailable.for.service:."AccessPolicyList"
7/27/00 9:00 AM : Command "WebSphere Commerce Server.stop" completed successfully.
7/27/00 9:01 AM : Command "WebSphere Commerce Server.start" running ...
7/27/00 9:02 AM : AUDIT [rs600035/WebSphere Commerce Server]: DrAdmin available on port 58,518
7/27/00 9:02 AM : AUDIT [rs600035/WebSphere Commerce Server]: Loading.servlet:."jsp"
7/27/00 9:02 AM : AUDIT [rs600035/WebSphere Commerce Server]: [Servlet.LOG]:."com.sun.jsp.runtime.JspServlet: init"
7/27/00 9:02 AM : AUDIT [rs600035/WebSphere Commerce Server]: Servlet.available.for.service:."jsp"
7/27/00 9:02 AM : AUDIT [rs600035/WebSphere Commerce Server]: Loading.servlet:."errorjsp"
7/27/00 9:02 AM : AUDIT [rs600035/WebSphere Commerce Server]: [Servlet.LOG]:."com.ibm.servlet.jsp.http.pagecompile.PageCompileServlet: init"
7/27/00 9:02 AM : AUDIT [rs600035/WebSphere Commerce Server]: Servlet.available.for.service:."errorjsp"
7/27/00 9:02 AM : AUDIT [rs600035/WebSphere Commerce Server]: Loading.servlet:."MerchantAdmin"
7/27/00 9:02 AM : AUDIT [rs600035/WebSphere Commerce Server]: [Servlet.LOG]:."com.ibm.commerce.tools.request_management.RequestManager:
init"
7/27/00 9:02 AM : AUDIT [rs600035/WebSphere Commerce Server]: Servlet.available.for.service:."MerchantAdmin"
7/27/00 9:02 AM : AUDIT [rs600035/WebSphere Commerce Server]: Loading.servlet:."AdminController"
7/27/00 9:02 AM : AUDIT [rs600035/WebSphere Commerce Server]: [Servlet.LOG]:."com.ibm.commerce.admin.icontrollers.AdminController: init"
7/27/00 9:02 AM : AUDIT [rs600035/WebSphere Commerce Server]: Servlet.available.for.service:."AdminController"
7/27/00 9:05 AM : Command "WebSphere Commerce Server.start" completed successfully.
```

Figure 32. WebSphere Administrative Console

The WebSphere Administrative Console enables administrators to access the administrative server on each node in the administrative domain and provides a view of the domain's topology. It supplies task wizards for managing and combining resources in the topology. The Administrative Console provides three views to facilitate the administrative functions. You can use the tabs in the navigation area to access each view. The three views are

- Tasks: This view provides access to all the administrative tasks, in three major categories:

   - Configuration tasks: This allows you to perform initial configuration of anew application server.
   - Performance tasks: You can use the Resource Analyzer to monitor the performance of your resources.
   - Security tasks: Under this task you are able to configure security for your application.
- Types: This view displays a hierarchy or tree view of the potential resources that can exist on a physical machine in the administrative domain. You can see all the configurable items and how many of each kind already exists. You can see the default properties of each type of resource and you can see the relationships among objects in the administrative domain.
- Topology: The topology view displays all managed nodes in the administrative domain, and for each node, a hierarchy of existing resources associated with that node.

In the WebSphere administrative model, there are several terms that are used frequently:

- Node

  A physical machine that contains Web resources is called a node.
- Administrative Server

  All the Web resources of a node is administered by an administrative server.
- Administrative Repository

  All data for a given node and its administrative data is stored in an administrative repository, typically a DB2 database. The person installing IBM WebSphere Application Server specifies which administrative repository a given administrative server will use.
- Administrative Domain

  If more than one node is administered by an administrative server, typically all node information is stored in one administrative repository, allowing the nodes to know about existence of other nodes so that if desired, nodes would distribute applications among themselves. This type of arrangement is called an administrative domain.
- Topology

  A topology is the collection of all the nodes and their resources in an administrative domain.

- Administrative Resources

  The resources on a node, such as servlet class files and enterprise bean JAR files, are represented as administrative resources in the administrative domain. An administrative resource, such as a servlet, holds configuration information about a "real" resource, such as a servlet file installed on a node. It provides a way to start, stop, and otherwise manage the real resource, perhaps remotely.

- Containment Hierarchy

  The topology of the administrative domain arranges the nodes and the resources within each node in a hierarchical structure referred to as the containment hierarchy.

9.2.1.2 Relationships among administered resources

The containment hierarchy represents a parent child view of resources within a node and the nodes within an administrative domain. The concept of containment hierarchy is relatively simple: in order to have a child a parent must exist. A very close analogy is a directory/file structure. A directory must exist in order to place a file in it.

In the WebSphere Application Server administrative domain context, some specific rules exist. For example, in WebSphere Application Server, an application server resource contains a servlet engine and one or more EJB containers.

Note

It is a feature of the containment hierarchy that an application server can contain only one servlet engine.

The servlet engine contains one or more Web applications, each of which contains one or more servlets. The EJB container contains enterprise beans.

We cannot add a servlet to the administrative domain unless a Web application exists in which to place the servlet. Similarly, an application server and servlet engine must exist to support the Web application containing the servlet.

We will discuss the containment hierarchy for all resources in 9.2.1.4, “Resources you can administer” on page 140.

9.2.1.3 Administrative tasks

This section discusses the general administrative tasks that can be carried out using the IBM WebSphere Application Server including:

- Performing daily administrative operations

  Day-to-day administration involves:

  - Ensuring that resources are available (running).
  - Starting and stopping servers and servlets as necessary.
  - Making incremental adjustments to the configurations of resources in the administrative domain.

  Modifications can be small, such as granting permissions to access a new application, reinstalling an enterprise bean after changing its JAR file, or changing the frequency with which servers are queried to determine their state.

  Large-scale modifications can include introducing new resources into the domain or redefining the mix of resources in an application.

- Configuring applications and their components

  When a servlet, enterprise bean, Java Server Pages (JSP) file, and Web page that work together to perform a particular business logic function, this combination of Java and Web components is called an enterprise application. The WebSphere Application Server provides administrative support for defining and managing enterprise applications and their components. After configuring an application, you can use the administrative facilities to start and stop the application as a logical unit. For example, when you start an application, all of the application's components (servlets, enterprise beans, and so on) start, too. You do not need to start each component of the application separately.

- Controlling access to applications (security)

  After configuring applications, you will likely want to limit access to them. For example, the public should not be permitted to use an application that accesses a database containing sensitive company information. The IBM WebSphere Application Server lets you establish and enforce authentication, authorization, and delegation policies to control access to your applications.

- Analyzing usage statistics and performance

  Resource analysis tools can be used to review current and historical information about resources in the domain. You can monitor performance and load statistics for servlets, enterprise beans, sessions, database connection pools, and server resources.

- Optimizing performance

  Resources in an administrative domain can be cloned to improve performance or availability. For example, application servers can be cloned to form a server group (a collection of identical instances of application server processes).

  Cloning application servers improves the throughput of client remote method invocations by distributing the load among the members of the group. It also improves availability and can prevent a single point of failure.

  You can also use cloning to simplify configuration tasks. For example, you can configure a resource, test the configuration, and then duplicate the resource for use on other nodes in the domain.

- Troubleshooting

  You can:

  - Monitor transactions, forcing outcomes when necessary
  - Analyze resources such as servlets and enterprise beans
  - Trace and debug applications
  - View traces, logs, and messages

9.2.1.4 Resources you can administer

This section provides an overview of the administrative resources you can use the WebSphere Administrative Console to manage.

Some administrative resources represent Java component files on your system, such as servlets and enterprise bean JAR files. Other administrative resources provide support for managing, combining, distributing, and securing these components -- resources such as Session Managers, and servlet redirectors.

Although the administrative domain is comprised of resources, the WebSphere Administrative Console provides task wizards. You can take a task-oriented approach or an resource-oriented approach, or both.

The administrative resources found in the administrative domain's containment hierarchy include:

- Nodes

  Use nodes to specify machines to which you can distribute servers, servlets, enterprise beans, applications, and other resources for workload management.

Nodes are physical machines in an administrative domain. Each node must contain an administrative server. In this way, a node also represents the administrative server process on the node. For example, the administrative console can show that the node resource is in the "running" state. That indicates not only that the physical machine is running, but that the administrative server on the machine is also running. The administrative server must be running in order for the node to be active and operational in the administrative domain.

In the containment hierarchy, nodes contain application servers, generic servers, servlet redirectors, and the children of these resources.

Nodes are contained by the administrative domains to which they belong.

- Servlets

  Use servlets to manage servlet files individually and include them in applications.

  A servlet is a Java program that uses the Java Servlet Application Programming Interface (API) and associated classes and methods. Servlets extend a Web server's capabilities by providing request and response services to clients.

  In the containment hierarchy, servlets do not contain any other resources.

  Servlets are contained by Web applications. In fact, a servlet must be part of a Web application, even if the Web application is comprised of the single servlet.

- Web applications

  Use Web applications to put together a combination of one or more servlets and Web resources (JSP files, Web pages, and Web paths for servlets).

  A Web application is a group of servlets, and perhaps JSP and HTML files, that share the same servlet context and can be managed as a unit.

  In the containment hierarchy, Web applications contain servlets. Note, that Web applications can "contain" Web resources, but Web resources are located under virtual hosts in the containment hierarchy.

  Web applications are contained by servlet engines.

- Servlet engine

  Use servlet engines to extend your Web server's capability to handle requests for servlets and the applications and Web applications that contain them.

A servlet engine is a program that runs within an application server, processing requests for servlets, Java Server Pages (JSP) files, and other types of server-side include coding. The servlet engine creates instances of servlets, initializes them, acts as a request dispatcher, and maintains servlet contexts for use by applications.

Servlet engines contain Web applications.

Servlet engines are contained by application servers.

- Application servers

  Use application servers to extend your Web server's capabilities to handle requests for enterprise beans, servlets, Web applications, and the applications that contain them.

  An application server consists of a Java Virtual Machine (JVM) configuration and an Enterprise JavaBean server process for hosting Enterprise JavaBeans and applications comprised of enterprise beans and other resources.

  The "application server" should not be confused with the “IBM WebSphere Application Server" product. The product can include one or more application server processes for each machine on which it is installed.

  In the containment hierarchy, application servers contain EJB containers and Enterprise JavaBeans.They can also contain servlet engines (one per application server) and their children.

  Application servers are contained by nodes. Multiple application servers can exist on the same node.

- Enterprise Beans

  Use Enterprise JavaBeans to configure Enterprise JavaBean JAR files you want to manage individually or as part of applications.

  An Enterprise JavaBean represents a deployed Enterprise JavaBean. An Enterprise JavaBean is a Java component that can be combined with other Enterprise JavaBeans and Java components to create a distributed, client/server application. There are two types of Enterprise JavaBeans. An entity bean encapsulates permanent data, which is stored in a data source (database), and provides associated methods to manipulate the data. A session bean encapsulates ephemeral (nonpermanent) data and business logic associated with a client session.

  In the containment hierarchy, Enterprise JavaBeans do not contain other resources.

  Enterprise JavaBeans are contained by EJB containers.

- EJB containers

  Use EJB containers to configure container support for Enterprise JavaBeans.

  An EJB container is a runtime resource used to contain and host Enterprise JavaBeans. Properties on the container are used to fine tune the runtime to the requirements of the enterprise beans.

  In the containment hierarchy, EJB containers contain Enterprise JavaBean resources.

  EJB containers are contained by application servers.

- Enterprise applications

  Use enterprise applications to put together a combination of one or more Enterprise JavaBeans, servlets, JSP files, and Web pages.

  An enterprise application is a collection of Enterprise JavaBeans and servlets that represent a user application.

  In the containment hierarchy, applications do not contain other resources. This might seem counter-intuitive because applications are comprised of other resources in the administrative domain, such as servlets and Enterprise JavaBeans. The resources comprising an application reside in different areas of the containment hierarchy. For example, the servlets reside in a Web application under a servlet engine, and the Enterprise JavaBeans reside in a container under an application server.

  Applications are contained only by the administrative domain, indicating that applications are not associated with particular nodes in the domain.

- Data sources

  Use data sources to configure and pool database connections, saving administration time.

  A data source resource represents the logical name of a JDBC-enabled database used by entity beans to store persistent data. Data sources shield the Enterprise JavaBean developer from the underlying physical location of the database. A data source is associated with a driver resource; a driver can have many data sources associated with it. Applications use data sources by looking them up in the Java Naming and Directory Interface (JNDI) name space.

  In the containment hierarchy, data sources do not contain other resources.

  Data sources are contained only by the administrative domain, indicating that data sources are not associated with particular nodes.

- JDBC drivers

Use JDBC drivers to specify the location of the Java code for your database's JDBC driver.

A JDBC (Java Database Connectivity) driver resource represents the installation of a database JDBC driver on a node.

In the containment hierarchy, JDBC drivers do not contain other resources.

JDBC drivers are contained only by the administrative domain, indicating that JDBC drivers are not associated with particular nodes.

- Models

  Use models to copy administrative resources and optionally distribute the "clones" (copies) to remote machines.

  A model is an active template for creating clones of a server or other resource. Cloning can improve performance and availability by distributing the load among nearly identical copies of an administrative resource. The model and clones can be administered efficiently because changing the model settings automatically propagates the same changes to the clones.

  In the containment hierarchy, models do not contain any resources.

  Models are contained only by the administrative domain because models are not associated with particular nodes.

- Servlet redirectors

  Use servlet redirectors to route servlet requests to nodes remote to the Web server.

  A servlet redirector is a process that uses Remote Method Invocation (RMI) over the Internet Inter-ORB Protocol (IIOP) to distribute servlet requests to servlets on machines remote to the Web server.

  In the containment hierarchy, servlet redirectors do not contain any other resources.

  Servlet redirectors are contained by nodes.

- Virtual hosts

  Use virtual hosting to isolate applications. You can make a physical host (node) seem to be multiple hosts, each hosting its own content separately from the others.

  A virtual host is a servlet host that maintains a list of one or more Web applications to which it routes HTTP requests that the servlet engine passes to it. The virtual host also maintains a list of Multipurpose Internet Mail Extensions (MIME) types that it can process.

In the containment hierarchy, virtual hosts contain Web resources (see below).

Virtual hosts are contained only by the administrative domain, indicating that virtual hosts are not associated with particular nodes.

- Web resources

  Use Web resources to define served paths for servlets, JSP files, and Web pages.

  A Web resource is a path representing a point of content on the Web, including text, video or sound clips, images, or programs. The most common Web resource is a Web page address -- a Uniform Resource Locator (URL), although a Web resource can represent a "served path" of a servlet or JavaServer Pages (JSP) file.

  In the containment hierarchy, Web resources do not contain any other resources.

  Web resources are contained by virtual hosts. They can be associated with applications and Web applications on various nodes.

- Session managers

  Use session managers to relate user requests into logical sessions that support interactive, personalized Web site visits.

  A session manager is a process that stores state information about servlets. The session manager tracks and ties together multiple servlets into a session cluster so that data can be shared among applications

  In the containment hierarchy, session managers do not contain any other resources.

  Session managers are contained by servlet engines. Each servlet engine can have one session manager.

- User profiles

  User profiles maintain information about visitors to your Web sites, such as name, e-mail address, and preferences.

  User profiles do not contain any other resources.

  User profiles are contained by servlet engines. Each servlet engine can have one user profile resource, which manages user profiles for all servlets in that servlet engine.

- Generic servers

  Use generic servers to manage non-WebSphere processes within your WebSphere administrative domain.

A "generic" server is a managed process, meaning a process that is invoked and monitored by the administration infrastructure. For example, you can add CORBA servers, RMI servers, and C++ servers to the administrative domain so that you can start and stop them in unison with the applications they support.

In the containment hierarchy, generic servers do not contain other resources.

Generic servers are contained by nodes.

9.2.1.5 Configuring default values

The previous section discussed the many types of resources you can configure in an administrative domain. This section discusses a few important points about configuring resources.

- Each resource has custom and default properties.
- Some configuration changes do not take effect immediately, and can be performed in batches.
- Most properties are not required.

The IBM WebSphere Application Server Version 3 installation program provides a "default administrative configuration" option that populates the administrative domain with administrative resources such as a default application server. The option is part of the custom installation.

In the containment hierarchy, most administrative resources have a required parent. The default configuration provides many of these parents (containers) that support your servlets and enterprise beans. For example, in the administrative domain:

1. A servlet must be contained by a Web application.
2. A Web application must be contained by a servlet engine.
3. A servlet engine must be contained by an application server.
4. An application server must be contained by a node.

Without the default configuration, you would need to configure application server, servlet engine, and Web application resources before configuring a single servlet.

With the default configuration, you can immediately configure your servlet. You can specify that it will be contained by the default application server, servlet engine, and Web application, at least until you are ready to configure your own versions of these prerequisite resources.

The default configuration includes an application server (default server), container (default container), application (default app), and so on. These resources can be found in the Topology tree on the Topology tab.

If you find that the default configuration was not installed, see the Getting Started book for a discussion of your options for acquiring the default configuration.

If you are inexperienced with the Version 3 administrative model, it is highly recommended that you install and use the default administrative configuration in your development environment, and in your production environment if it suits your needs.

9.2.1.6 How centralized administration works

This section discusses how the configurations you specify are kept in the administrative repository for centralized administration.

WebSphere Application Server provides centralized administration of application servers, servlets, and other resources. An administrative server tracks a domain's contents and activities by maintaining an administrative repository. The repository is the database of information about an administrative domain and can be shared by several administrative servers on multiple nodes in the domain.

Each resource in the WebSphere administrative domain corresponds to an object in the repository. For example, when you create an application, a corresponding application object is created in the repository. In this way, the administrative repository contents mirror the contents of the administrative domain.

The repository contains descriptive information about the resources that are configured to run on each node in the domain. For example, the repository contains the names of application servers, the node each server is running on, the enterprise beans installed in each server, and each server's current state (running, for example).

The repository allows you to administer the domain from any machine. All information is stored in a central location. Each administrative server has a central view of configuration information about all resources in the domain. When you modify a resource's configuration, the changes are seen by all administrative servers.

The resources in a WebSphere administrative domain are represented in the administrative repository as entity beans with container-managed persistence

(CMP). The persistent data associated with a resource (for example, the name, current state, and working directory of an application server) is stored in the administrative repository.

Administration occurs through method calls to resource beans in the administrative server. A graphical administrative client (the WebSphere Administrative Console) makes requests on your behalf to an administrative server to access or modify a resource in the domain. An administrative server also communicates with other, remote administrative servers to delegate tasks and to respond to requests.

In the administrative server, session beans invoke methods on the resource beans. For example, you can start, stop, ping, and modify application servers in the WebSphere Administrative Console, which in turn invokes methods on the resource beans for the application servers.

To learn more about this topics please read the WebSphere Application Server Administrative Console online help.

9.2.2 Site Analyzer

IBM WebSphere Application Server Site Analyzer Version 3.0.2 is not one of the core components of the e-Marketplace applications but it is an excellent tool that provides analysis features and customizable reporting options that help you improve your Web site content and performance (content analysis), as well as better understand how a site is used by its visitors (usage analysis). Using Site Analyzer, you can quickly and easily report on everything from aggregate page sizes and broken links to site visit information and errors.

You can customize how your data is viewed by tapping into a set of predefined report elements or building custom reports that collect information specific to a site. Site Analyzer stores information in a built-in database, providing scalability and letting you create trend reports that show Web site content and usage growth, and change over time.

9.2.2.1 Site Analyzer configuration

Site Analyzer uses a client/server configuration. The server portion performs content and usage analysis and the client portion displays results of these analyses.

Note

If you plan to perform usage analysis, you need access to your HTTP server's log files. In addition, you might need to reconfigure the HTTP server logs in order for Site Analyzer to process them correctly.

The client, server, and database portions can be installed on one machine (Personal Configuration) or on multiple machines (Workgroup Configuration). In the Workgroup Configuration, one machine is the server, and one or more other machines are the clients. For more information about IBM WebSphere Application Server Site Analyzer please refer to the IBM WebSphere Application Server online documentation.

9.2.2.2 Starting Site Analyzer

If you are using the DB2 Database provided with Site Analyzer, when you get to the Database panel of the wizard that opens when you first start the Site Analyzer server, enter `jdbc:db2:sadata` for the local database URL or `dbc:db2://fully_qualified_host_name/sadata` for the remote database URL (where `fully_qualified_host_name` is the hostname of your server).

If you are using a previously installed copy of IBM DB2 UDB (local or remote), you must set up your system so DB2 will work with Site Analyzer before starting the Site Analyzer server.

Starting the Site Analyzer server

If you are using a DB2 database not installed by Site Analyzer, make sure you have modified the `wssas` script to refer to the correct DB2 instance. In order for a remote Site Analyzer client to access the Site Analyzer server, you must start the DB2's db2jstrt application on the DB2 machine. This application uses a port number as on optional parameter. Here are the steps to start the Site Analyzer on AIX:

1. Enter `su - <iname>`

   where `iname` is the db2 name specified during installation, db2inst1 by default.

2. Enter `../sqllib/db2profile`
3. Enter `db2jstrt [port]`

   where `port` is any open port on the machine.

4. Enter `exit`
5. Enter `cd /usr/WebSphere/SiteAnalizer`

6. Enter `./wssas`

The first time you start the server a wizard for configuring Site Analyzer opens. To change the settings for the server at a later date, you can use the Preferences panel. On AIX the Preferences panel starts automatically when you start the server. (Enter `wssas` at a command line prompt.)

Tip

To avoid the configuration panel on AIX or Solaris, enter `wssas -noconfig` at a command-line prompt. This starts the Site Analyzer server.

Starting the Site Analyzer client

To start the client on Windows 95, 98, or NT:

1. Click the **Start** menu.
2. Select **Programs**.
3. Select **IBM WebSphere**.
4. Click on **Site Analyzer 3.0**.

As with the Site Analyzer server, the first time you start the client the startup wizard walks you through setting general preferences.

Reporting

Generating reports is one of the main functions of the Site Analyzer. There are a number of steps that are required to be taken before a report can be generated and viewed. Details of report creation and generation is beyond the scope of this book.

For more information please refer to the Site Analyzer online documentation or visit:

```
Http://www-4.ibm.com/software/webservers/siteanalyzer/doc/help/sacontents.
html
```

9.2.3 WebSphere Commerce Suite, Marketplace Edition for AIX

The WebSphere Commerce Suite, Marketplace Edition for AIX provides a rich set of commands, interaction controllers, JSPs and beans to create an e-Marketplace. After an e-Marketplace is created,maintenance and management of the site becomes a critical activity. As with WebSphere Commerce Suite 4.1 the WebSphere Commerce Suite, Marketplace Edition for AIX creates an instance that controls the configuration characteristics of the e-Marketplace. Management and configuration of this instance is achieved through the WebSphere Commerce Suite Marketplace Edition's Configuration Manager. On AIX the configuration manager can be started by taking the following steps:

1. Enter `cd /usr/lpp/CommerceSuite/server/bin`
2. Enter `./start_config`

The WebSphere Commerce Suite, Marketplace Edition for AIX also provides a Web based interface, known as "ncadmin" to manage the characteristics of your e-Marketplace site.

The following sections cover both these configuration and management interfaces.

9.2.3.1 Configuration Manager

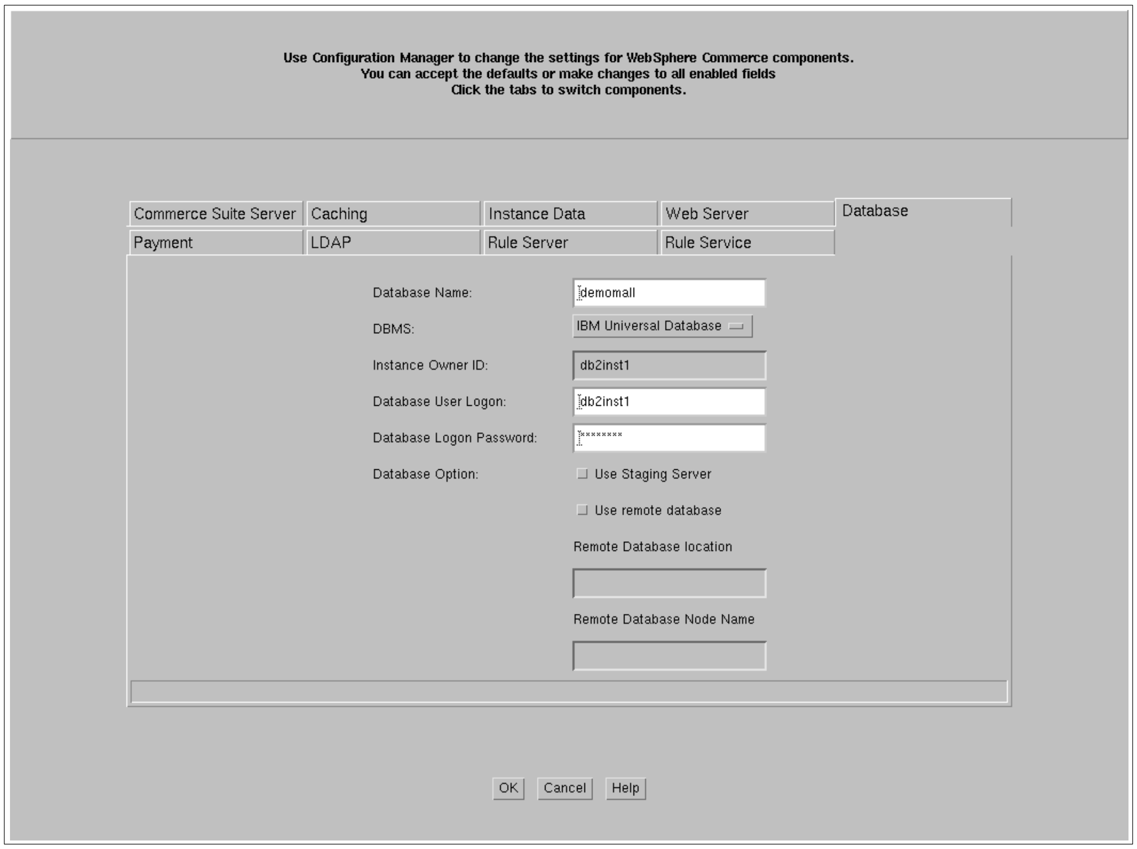
The Configuration Manager tool has a Java-based graphical interface that lets you modify the way the WebSphere Commerce Suite, Marketplace Edition for AIX is configured, without dealing with the intricacies of syntax-sensitive configuration files. Configuration Manager also makes it easy to control many of the administration tasks associated with the WebSphere Commerce Suite, Marketplace Edition for AIX.

Use this tool to perform the following tasks:

- Create a new Commerce Suite instance
- Stop and start a Commerce Suite instance
- Delete a Commerce Suite instance
- Change the configuration settings for a Commerce Suite instance

When you are creating a new instance or changing the configuration of an existing instance, the Configuration Manager displays a set of panels with input fields for all the applicable parameters.

Figure 33 on page 152 shows the Configuration Manager's instance configuration main window with tabs to configuration panels.



Use Configuration Manager to change the settings for WebSphere Commerce components.
You can accept the defaults or make changes to all enabled fields
Click the tabs to switch components.

Database Name:	demomall
DBMS:	IBM Universal Database
Instance Owner ID:	db2inst1
Database User Logon:	db2inst1
Database Logon Password:	********
Database Option:	Use Staging Server
	Use remote database
	Remote Database location
	Remote Database Node Name

Figure 33. Instance configuration window

In here we will discuss the configuration panels.

- Database

  In this tab you specify all your database related information, such as database name, database management system type, instance owner ID, a user logon ID and a password, etc.

- Payment

  Although you can configure your commerce instance to handle payments, in reality in an e-Marketplace almost all payments are handled via invoicing systems. If you have the requirement to enable payment please refer to WebSphere Commerce Suite documentation for installation and configuration.

- LDAP

In this tab you would specify your LDAP information. LDAP is a mandatory component of a e-Marketplace application please refer to the installation information in Appendix A, “Marketplace Edition installation guide” on page 455 for the steps you need to take to enable LDAP correctly.

- Rule server

  This tab allows you to enable Rule server. In an e-Marketplace instance the Rule server is not configured and used.

- Rule services

  This tab allows you to configure the rule services. In an e-Marketplace instance there are no rule services configured and used.

- Web server

  In this tab you specify information pertaining to your Web server.

- Instance Data

  In this tab you specify your e-Marketplace application’s instance root path, configuration file path, log file path and cache file path.

- Caching

  In this tab you specify the type of caching used or if there is no caching. We recommend that during your development phase you turn caching off so that changes made to your categories and/or products would be displayed without having to clear the cache files.

- Commerce Suite Server

  In this tab you specify the name of your e-Marketplace instance, your WebSphere data source and desired authentication mode.

For more information on these functions please refer to the WebSphere Commerce Suite, Marketplace Edition for AIX online documentation.

The configuration parameters entered through the configuration manager are stored in number of configuration files, they are httpd.conf, ncommerce.conf scheduler.conf and srvrctrl.conf.

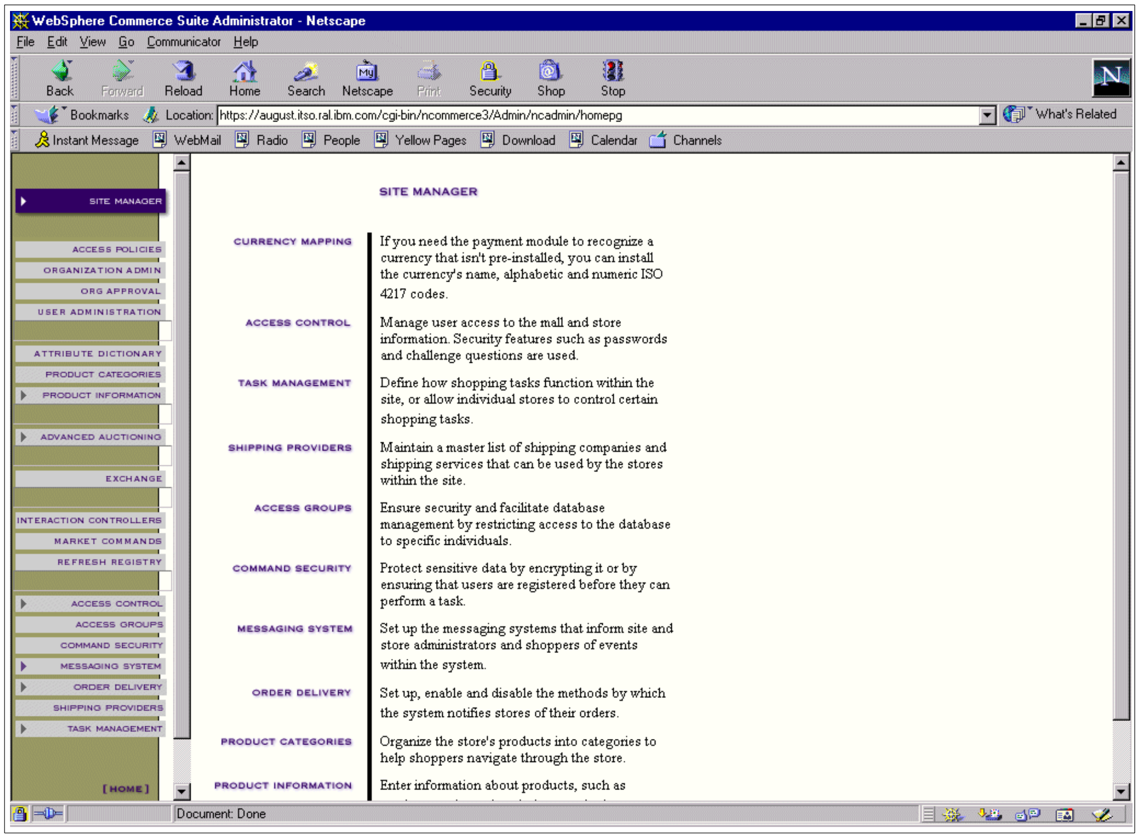
At the time this section was written, there were some configuration parameters needed to be manually entered in the above-mentioned files. These parameters are described in Appendix A, “Marketplace Edition installation guide” on page 455. We recommend you refer to your WebSphere Commerce Suite, Marketplace Edition for AIX product documentation for the most recent information.

9.2.3.2 NCADMIN management

The WebSphere Commerce Suite, Marketplace Edition for AIX's Administrator allows you to create and maintain the characteristics of your e-Marketplace site through a Web based interface. Unlike the Administrator in WebSphere Commerce Suite 4.1, the Administrator in WebSphere Commerce Suite, Marketplace Edition for AIX only provides Site Manager functionality since the concepts of stores and Store Managers no longer apply.

Site manager

The Commerce Suite Site Manager, referred to as the Site Manager, is a collection of online forms that you use to manage some high-level functions for e-Marketplace site. The person who manages these functions is called the e-Marketplace administrator. Figure 34 on page 155 displays the Site Manager window of the WebSphere Commerce Suite, Marketplace Edition for AIX.



SITE MANAGER

CURRENCY MAPPING	If you need the payment module to recognize a currency that isn't pre-installed, you can install the currency's name, alphabetic and numeric ISO 4217 codes.
ACCESS CONTROL	Manage user access to the mall and store information. Security features such as passwords and challenge questions are used.
TASK MANAGEMENT	Define how shopping tasks function within the site, or allow individual stores to control certain shopping tasks.
SHIPPING PROVIDERS	Maintain a master list of shipping companies and shipping services that can be used by the stores within the site.
ACCESS GROUPS	Ensure security and facilitate database management by restricting access to the database to specific individuals.
COMMAND SECURITY	Protect sensitive data by encrypting it or by ensuring that users are registered before they can perform a task.
MESSAGING SYSTEM	Set up the messaging systems that inform site and store administrators and shoppers of events within the system.
ORDER DELIVERY	Set up, enable and disable the methods by which the system notifies stores of their orders.
PRODUCT CATEGORIES	Organize the store's products into categories to help shoppers navigate through the store.
PRODUCT INFORMATION	Enter information about products, such as

Figure 34. Site Manger

Using the Site Manager the you can do the following:

- Access Policies: This form lists all access policies loaded in LDAP. You may create new access policies or modify existing ones.
- Organization Admin: This form allows you to perform organization management tasks, such as creating and editing groups or editing the organization information.
- Org Approval: This form allows you to approve organizations that would like to join the e-Marketplace.
- User Administration: This form allows you to manage user profiles.
- Attribute Dictionary: This form allows you to create and manage the e-Marketplace data dictionary.

- Product Categories: This form allows you to create and manage categories for the e-Marketplace site.
- Product Information: These forms allow you to create and manage product descriptions for the e-Marketplace catalog.
- Advanced Auctioning: These forms allow you to define and manage auction types and rules that apply to them.
- Exchanges: These forms allow you to create and manage trading posts and the matching rules that apply to them. You can also create exchange offerings using these forms.
- Interaction controllers: This form allows you to manage subsystem interaction controllers registered in the e-Marketplace.
- Market Commands: This form allows you to manage the Java-based commands registered in the e-Marketplace.
- Refresh Registry: This action refreshes the registries.
- Access Control: These forms provide an interface to manage administrators of the e-Marketplace or the member organization.
- Access Groups: These forms provide access control to e-Marketplace and organization administrative groups.
- Command Security: This form allows you to manage the command security by providing SSL encryption and authentication.
- Messaging System: These forms allow you to set up the messaging systems to communicate with members in the e-Marketplace.
- Order Delivery: These forms allow you to manage the order delivery methods available in the e-Marketplace.
- Shipping Providers: This form allows you to manage shipping companies who provide services to the e-Marketplace.
- Task Management: These forms allow you to define how tasks function within the e-Marketplace. These only govern the legacy NC commands.

9.2.4 SecureWay LDAP

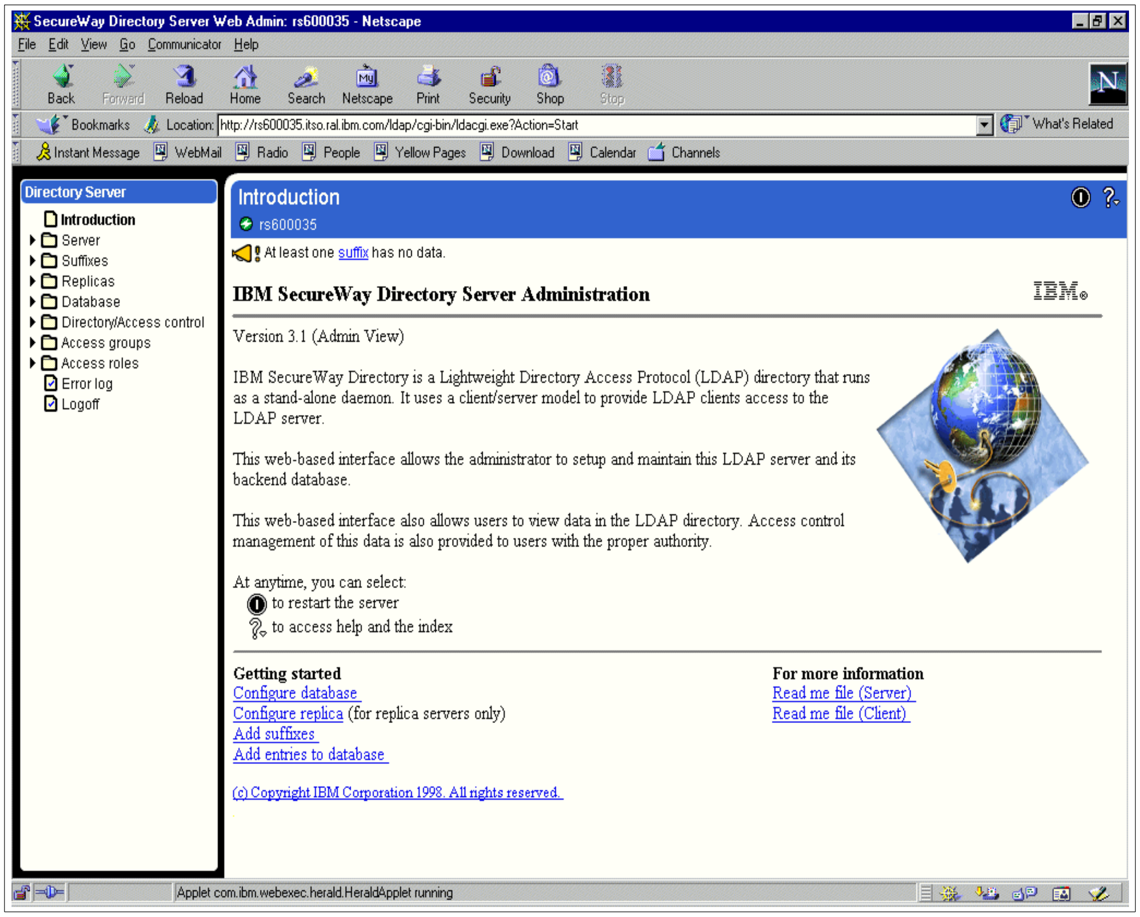
IBM SecureWay Lightweight Directory Access Protocol (LDAP) is used in WebSphere Commerce Suite, Marketplace Edition for AIX to manage user authentication and enforce access control. The majority of user information of e-Marketplace members are stored in the WebSphere Commerce Suite, Marketplace Edition for AIX instance database. LDAP also contains entries about the e-Marketplace members. User information stored in LDAP is used to perform user authentication during the logon. During installation of

WebSphere Commerce Suite, Marketplace Edition for AIX, LDAP is populated with access control policy entries to govern user access to e-Marketplace Command, Interaction Controller and other objects. These entries are used to validate authorization for command execution requests by users. There are two interfaces available to manage LDAP. A Web-based interface allows LDAP server management and a Java-based application is used to manage the LDAP directory.

9.2.4.1 LDAP Server Management Tool

The management of the LDAP server is handled through a Web-based interface. This interface can be accessed by requesting the appropriate URL provided to you during installation of LDAP. In a typical installation the URL is `http://yourhostname/ldap`. In order to access the LDAP Server interface you must log on at the above-mentioned URL.

Figure 35 on page 158 shows the main LDAP server page.


SecureWay Directory Server Web Admin: rs600035 - Netscape

http://rs600035.itso.ral.ibm.com/ldap/cgi-bin/ldacgi.exe?Action=Start

Introduction
rs600035

At least one suffix has no data.

IBM SecureWay Directory Server Administration

Version 3.1 (Admin View)

IBM SecureWay Directory is a Lightweight Directory Access Protocol (LDAP) directory that runs as a stand-alone daemon. It uses a client/server model to provide LDAP clients access to the LDAP server.

This web-based interface allows the administrator to setup and maintain this LDAP server and its backend database.

This web-based interface also allows users to view data in the LDAP directory. Access control management of this data is also provided to users with the proper authority.

At anytime, you can select:
to restart the server
to access help and the index

Getting started
Configure database
Configure replica (for replica servers only)
Add suffixes
Add entries to database

For more information
Read me file (Server)
Read me file (Client)

(c) Copyright IBM Corporation 1998. All rights reserved.

Applet com.ibm.webexec.herald.HeraldApplet running

Figure 35. LDAP server main window

Through this interface you can perform the following actions:

- Using the Server tab you can:
  - Modify the LDAP server properties
  - Configure the LDAP Server SSL properties
  - Cchange master/replica configuration of the LDAP server
  - View the LDAP Server status
  - View the LDAP server connection information
  - Start and stop the LDAP server
- Using the Suffixes tab you can:

- List, add and delete suffixes.

- Using the Replica tab you can:
  - List, add, delete and edit replicas.
- Using the Database tab you can:
  - View and change some database properties
  - Back up the LDAP database
  - Import data into LDAP database
  - Perform database reorganization to improve performance
  - Perform database configuration tasks
- Using the Directory/Access control tab you can:
  - Browse the directory tree
  - Work with Distinguished Names (DN)
  - Perform access control operations
- Using the Access Groups tab you can:
  - Create, modify and list access groups
- Using the Access Roles tab you can:
  - Create, modify and list access roles
- Using the Error Logs you can view the log entries.
- Using Logoff you can exit from the LDAP server Web-based interface.

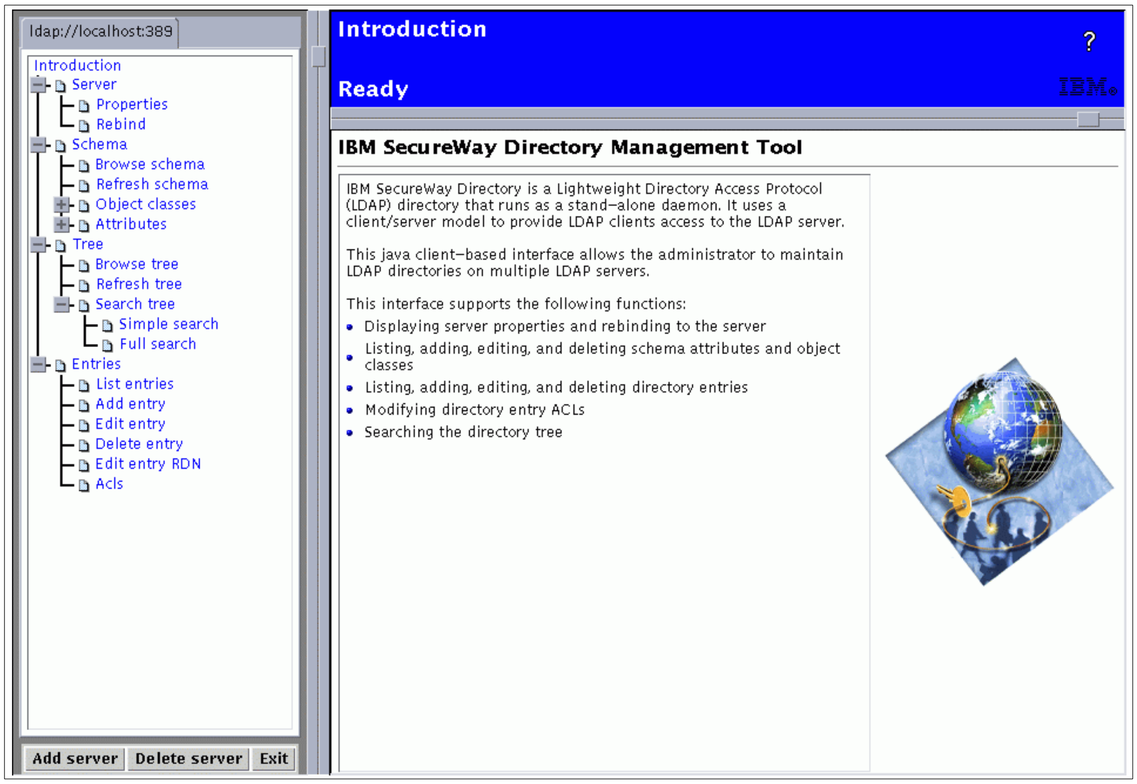
For more information on management of LDAP server please refer to the LDAP online documentation.

9.2.4.2 LDAP Directory Management Tool

The IBM SecureWay Directory Management Tool (DMT) provides a graphical user interface that enables you to manage information stored in directory servers. Use the tool to:

1. Connect to one or more directory servers via SSL or non-SSL connections
2. Display server properties and rebind to the server
3. List, add, edit, and delete schema attributes and object classes
4. List, add, edit, and delete directory entries
5. Modify directory entry ACLs
6. Search the directory tree

Figure 36 on page 160 shows the LDAP DMT.



IBM SecureWay Directory Management Tool

IBM SecureWay Directory is a Lightweight Directory Access Protocol (LDAP) directory that runs as a stand-alone daemon. It uses a client/server model to provide LDAP clients access to the LDAP server.

This java client-based interface allows the administrator to maintain LDAP directories on multiple LDAP servers.

This interface supports the following functions:

- Displaying server properties and rebinding to the server
- Listing, adding, editing, and deleting schema attributes and object classes
- Listing, adding, editing, and deleting directory entries
- Modifying directory entry ACLs
- Searching the directory tree

Figure 36. LDAP Directory Management Tool

In the remainder of this section we will provide detailed step-by-step direction on how to perform certain tasks that an LDAP directory administrator may need to perform.

Connect to Servers

- Log on to a server

  If a directory user's DN and password are not provided in the DMT configuration file, the tool connects as an anonymous user once it is started. Although an anonymous user can browse the directory tree and schema to perform directory updates, in most instances you need to log on as a directory user. To modify the directory server schema you must log on as the server administrator. To log on as a different user:

  - Click **Server -> Rebind**.
  - Provide the user's DN and password

- Press **Enter**.

The Directory Management Tool uses a configuration file located in /usr/ldap/etc/dmt.conf which is read when you start the Directory Management Tool, this file should look similar to this:

```
#browser=
server1.url=ldap://localhost:389
#server1.security.bindDN=
#server1.security.password=
#server1.security.ssl.keyclass=
#server1.security.ssl.keyclass.password=
```

In this file you can specify the LDAP server's URL and port, user DN and password and keyclass file name and password for SSL connections.

Note

We strongly recommend, for security reasons, that you do not provide DN user name and password in the dmt.conf file.

- Connect to an additional server

  If a local host is provided in the URL of the configuration file, DMT will connect to the LDAP directory server on the local machine where DMT is running. To add a connection to another directory server:

  Click the **Add server** button.

  Provide the server name, port, user DN and password.

  Press **Enter**.

Note

Do not specify the URL prefix (for example: `ldap://`) in the server name.

  You can choose to connect to the server via SSL by first selecting **Use SSL** and then providing the keyclass file name and password.

- Show Server Properties

  You can view the server properties by clicking on **Server**->**Properties**. The current bind DN, subschema entry, supported LDAP protocol versions, and the naming contexts that the server holds are displayed. For more information see the DSE documentation on your root server.

Work with Schema

- Browse Directory Schema

  The IBM SecureWay Version 3.1.1 Directory provides dynamically extensible schema support. A system administrator can define new attributes and object classes to enhance the default schema. The directory schema can be browsed and updated with DMT. You must log on as a directory administrator to update the schema. See the Admin GUI helps for information on schema.

  To browse the directory schema:

  - Click **Schema** -> **Browse schema**
  - Click the **+** sign to display the following selections:
    - attributetypes
      To view all defined attributes, or alternatively click on **Schema** -> **Attributes** -> **View attributes**.
    - objectclasses
      To view all defined object classes, or alternatively click on **Schema** -> **Object classes** -> **View object classes**
    - syntaxes
      To view all supported syntaxes.
    - matchingRules
      To view all supported matching rules.

- Add an object class

  To add an object class:

  - Click **Schema** -> **Object classes** -> **Add object class**.
  - Provide the object class name, description, and a unique string of object identifier (the OID).
  - Select the superior object class from which to inherit attributes.
  - Determine the object class type. (The default is **structural**.)
  - Click the **Required attributes** tab and then select the MUST have attributes from the attribute list in the left window. Click **Add** to move the selection to the right window. You can also select the attributes in the right window and then click **Remove** to deselect them. Click on **OK**.
  - Click the **Optional attributes** tab and then select the MAY have attributes.
  - Click **OK**.

- Edit an object class

  This operation is similar to Add An Object Class except that a pull-down menu is provided for the selection of the object class to be edited. To edit an existing object class:

  - Click **Schema** -> **Object classes** -> **Edit object classes**.
  - Select an object class to be edited from the pull-down menu.
  - Provide the description.
  - Click the **Required attributes** tab and select the MUST have attributes. Click **OK**.
  - Click the **Optional attributes** tab and then select the MAY have attributes.
  - Click **OK**.

- Delete object classes

  To delete one or more object classes:

  - Click **Schema** -> **Object classes** -> **Delete object classes**.
  - Select the object class or classes to be deleted.
  - Click **Delete**.
  - Click **OK** to confirm the deletion.

- Add an attribute

  To add an attribute:

  - Click **Schema** -> **Attributes** -> **Add attributes**.
  - Provide an attribute name, description, and an OID.
  - Select a syntax for this attribute from the list.
  - Determine whether this is a multi-valued attribute.
  - Select the matching rules used.
  - Click **OK**.

Note

Advanced users can click on the IBM extensions tab to change the DB2 table name, the DB2 column name, the security class, and the indexing.

- Edit an attribute

This operation is similar to Add An Attribute except that a pull-down menu is provided for the selection of the attribute to be edited. To edit an existing attribute:

- Click **Schema** -> **Attributes** -> **Edit attributes**.
- Select an attribute from the list.
- Make necessary changes to the entries in the General tab.
- Click OK.

Note

Advanced users can click the IBM Extensions tab to change the DB2 table name, the DB2 column name, the security class, and the indexing.

- Delete attributes

  To delete one or more object classes:

  - Click **Schema** -> **Attributes** -> **Delete attributes**.
  - Select the attribute or attributes to be deleted.
  - Click **Delete**.
  - Click **OK** to confirm the deletion.

Work with directory tree

- Browse directory tree

  You can browse the directory tree by using the Browse tree option.

  When you browse the directory tree the directory contents are displayed according to the directory hierarchies. To open part of the tree, expand the entries. The entries that are in the next level down are displayed. To browse the directory tree:

  - Click **Tree** -> **Browse tree**.
  - To expand the tree one level, click a **+** sign.

  The tool bar at the top of the window allows for an operation on a selected entry in the tree to be initiated. The operations include: **Add entry**, **Edit**, **Delete**, **Search**, **Expand**, **ACL settings**, and **Edit RDN**. Use **Edit** to view an entry. When an entry (a node in the tree) is selected click the **Expand** button to expand the entire subtree below the entry.

  Double-click an entry to edit it.

- Search directory tree

To allow convenient access to entries of special object classes such as user and group, DMT provides a simple search option in addition to a full search option. While the full search option allows you to provide complete specifications of all parameters to a directory search operation, the simple search requires only a minimal input for searching through a set of entries that belong to a selected object class.

To perform a **Simple search**:

- Click **Tree** -> **Search tree** -> **Simple search**.
- Select the type of entry to search.
- Determine the filter for the search result.
- Click **OK**.

To perform a **Full search:**

- Click **Tree** -> **Search tree** -> **Full search**.
- Input the search constraint:
  - Search base DN (the default is **all suffixes**)
  - Scope (the default is **subtree**)
  - Size limit (the default is **unlimited**)
  - Time limit (the default is **unlimited**)
  - Alias dereferencing (the default is **no**)
  - Referral chasing (the default is **yes**)
- Click the **Search filter** tab on the top of the display.
- Input the search filter. If necessary, use the **AND** or **OR** connectors.
- Click the **Search return set** tab.
- Select the attributes to be returned or the full entry.
- Click **OK**.

Work with directory entries

- Add a new entry

  To add an entry to the directory tree:

  - Click **Entries** -> **Add entry**. You can also add an entry if you click on **Browse tree**, select the parent entry, and then click **Add** on the toolbar.
  - Provide the parent DN and the Relative Distinguished Name (RDN) for the new entry. The RDN must be entered as an `attribute=value` pair.

- Choose the object type (object class) from the list or **other** for more options. If **other** is selected you can specify either a structural object class or an auxiliary object class or both.
- Click **OK**.
- Another window displays the attributes associated with the selected object class. Highlighted fields are required fields. Enter the attribute values for the entry. Use the Edit icon to add multiple values.

When the action is initiated from the tree browsing window, the parent entry can be selected from the directory tree and the parent DN is entered automatically.

Note

The flyover that appears when the cursor is positioned over the attribute name, or the text field describes the syntax of that attribute.

• Edit an entry (or view an entry)

To view an entry:

- Click **Entries** -> **Edit entry**. You can also select an entry if you click **Browse tree** and then double-click on the entry.
- Provide the entry DN to edit or view.
- Click **OK**.
- Another window displays the attributes associated with the selected object class. Highlighted fields are required fields. Enter the attribute values for the entry. Use the Edit icon to add multiple values.

Like **Add an entry**, this operation can be launched from the browsing tree window. The entry can be edited from the tree by double-clicking it.

Note

The flyover that appears when the cursor is positioned over the attribute name, or the text field describes the syntax of that attribute.

• Delete an entry

To delete an entry from the directory tree:

- Click **Entries** -> **Delete entry**, or from the browsing window click the entry to be deleted.

- Click **Delete**.
- Click **OK** to confirm the deletion.

- Edit an entry RDN

  To edit an entry RDN:

  - Click **Entries** -> **Edit entry RDN**. You can also select an entry if you click **Browse tree** and then click the entry.
  - Enter the current DN and provide the new RDN for the entry. (You can change the new RDN; the current DN is not editable.)
  - Click **OK**.

- ACL settings

  To modify the ACL for an entry:

  - Click **Entries**- > **ACLs**. Enter the DN and then click **OK**. You can also select the entry if you click **Browse tree** and then click the **ACL** button on the top of the window.
  - On the ACLs tab:
    - Either edit the existing list or create a new subject ACL list for a new subject.
    - Determine whether descendent entries are to inherit the ACL lists.
    - Mark the **Remove** check boxes for those subject ACL list, that are to be removed.
    - Mark/Unmark the check boxes for the rights to add/remove.
    - Mark/Unmark the check boxes for the rights to read/write/search/compare the attributes of three security classes.
    - To add a new subject ACL list, enter the subject DN, select the subject type, and then click **ADD**. A new list is added and ready for changes.
  - Click the **Owners** tab.
    - Determine whether descendent entries are to inherit the owner list.
    - Mark the **Remove** check boxes to remove owners.
    - To add a new owner, enter the owner DN, select the owner type, and then click **ADD**.
  - Click **Change**.

  See the *I*BM SecureWay documentation for additional information about ACLs.

Troubleshooting

The following error might occur the first time you edit a suffix or add an entry to a suffix, for example:

```
An error occurred getting attributes for entry c=us: noSuchObject.
```

This means that the suffix contains no data.

To add data to a suffix:

- In the navigation menu click **Entries > Add entry**.
- Leave the Parent DN blank and specify the suffix as the entry (for example `c=us`).
- Select the object type (for example `Country`) and click **OK**.
- Fill in the desired attribute values and click **Create**.

You should now be able to edit the suffix as well as add entries.

You can find the bind DN in either of two ways from the menu area:

- You can locate it on the **Rebind to Server** panel. Click **Server** -> **Rebind** to display the panel. The rebind DN will be displayed in the **User DN** field. (If the bind DN is anonymous, the **Anonymous** radio button is checked.)
- Click **Server** -> **Properties**. In the table, under Server attributes, find the BIND DN property.

9.2.5 Web server management

The primary function of a Web server is to provide a mechanism to receive client requests, direct these requests to an application server and present the application server response back to the client.

Depending on your choice of Web server, you may need to take into consideration different administrative and management issues that are specific to that Web server. Detailed coverage of all available Web servers, and administration and management tasks, is beyond the scope of this book. In this section we will provide some guidelines on the IBM HTTP Server.

9.2.5.1 IBM Administration Server

By leading you through complex configurations, the Administration Server greatly simplifies the once-manual task of configuring IBM HTTP Server. Once you select a server to configure, the Administration Server prompts you for configuration values, which are written to a configuration file when you click **Submit**.

In order to access the IBM Administration Server on AIX you must first start the IBM Administration Server. While you are logged on as root issue the following commands:

```
cd /usr/HTTPServer/bin
./adminctl start
```

This will start the IBM Administration Server. Tto access the administration server, open a browser and go to your IBM HTTP Server page by providing your local host URL, such as `http://rs600035.itso.ral.ibm.com/` and from the main page select **Configure Server**. At this stage you will be prompted for the administration user ID and password. This user ID and password were created during installation of the IBM HTTP Server.

The Administration Server is a browser-based application and requires one of the following browsers:

- Microsoft Internet Explorer 5.0 or later.
- Netscape Navigator 4.08 or later.
- Netscape Communicator 4.51 or later.

We highly recommend a screen resolution of 1024 x 768 with small (12 pt or less) fonts; significant deviation from these recommendations may cause behavior and/or layout problems. Do not override document font settings. Usability of the forms is best when the browser window is maximized. (**Note**: In Netscape Navigator, page layout may be disrupted if the browser window is resized. To correct the problem, simply reload the page or re-select the task.)

We recommend that after you gain access to the Administration Server, you spend some time reviewing the conventions and key points specified under the Getting Started folder. You can access this information by selecting **Getting Started** from the navigation panel at the left side of your browser window.

Tasks in the navigation panel

The navigation panel on the left side of your browser provides you with number of configurable and administrative options. The follwing is a high level definition of what each option does:

- Basic Settings: In this folder, specify settings that affect the general operation of the server. The first three pages, Core Settings, Advanced Properties, and Server Options, control core features of the server.
- Configuration Structure: When you configure the target server using the Administration Server, you apply settings to a subset of the server's resources, which we refer to as a scope. The Scope field at the top of

each page represents the subset of resources that are affected by the settings on that page. For example, to grant access to a particular directory, you might specify that directory in the Scope field on the Individual Access page.

- Indexing: The server can display a list of files available in a directory. Enable this function (called *Fancy Indexing*) by using the pages in this folder. You can control how the display is formatted and whether icons are used for different file types.
- Authentication Files: Here you manage users and groups. Instead of manually creating user authentication files and group authentication files in a text editor, you can create and edit those files here. After you have created these authentication files, use them to grant access to particular resources, under the Access Permissions folder. A user authentication file specified for a given scope contains the user IDs and passwords for all persons that can be granted permission to access that scope. Access can be granted to everyone in the user authentication file, or a subset of those users. The users listed in a user authentication file can be assembled into named groups defined in a group authentication file. Within a given scope, a group authentication file must only contain group members listed in the user authentication file for that scope. If access permission is granted to all in the user authentication file, then granting access to certain groups has no effect. Related tasks:
  - From the Individual Access page, grant access to specific resources for users in a user authentication file.
  - From the Group Access page, grant access to specific resources for groups in a group authentication file.
- Access Permissions: In this folder, you can allow and deny access to a set of resources in the server based on various factors about the client. Some of the pages in this folder refer to user authentication and group authentication files. To change the contents of these authentication files, go to the Authentication Files folder. After you have set up the authentication files, use them to grant access to particular resources here.
- Security: In this folder, you can specify settings related to secure connections. The name of your keyfile, the cipher specifications to use, and the type of client authentication in use are examples of settings you control in this folder.
- Logs: The server maintains log files to help you monitor the requests that it fulfills and the errors it encounters. Use the settings on the pages in this folder to adjust what information the server logs.

- Mappings: The tasks in this folder enable you to control how the server responds to requests after the requests have been sent by the client machine. You can adjust the messages that a client receives if there is a problem fulfilling the request; you can change the address where the server looks for a file in the server's file system; you can allow for address changes based on the user name of the client request; and you can redirect requests to different URLs.
- Scripts: Tasks in this folder enable the server's CGI scripts and server-side includes (SSIs) to inherit environment variables from the shell that invoked the server process. These settings are especially useful for migrating scripts from a CERN Web server to the IBM HTTP Server.
- Performance: Use the pages in this folder to tune factors that affect the server's performance. On the Server Settings page, many of the fields refer to core server functions. To use the Fast Response Cache Accelerator, specify settings on the Set Cache page. To enable the IBM HTTP Server to be monitored through SNMP, fill in the fields on the SNMP page.
- MIME: You can specify that the server handle resources differently, based on the MIME types of the resources. The MIME type of a document relates to its content and is returned to the browser or used in content-negotiation within the server. A handler can be set for a document. The handler determines how the document is processed within the server.
- View Configuration: For those who prefer to edit the configuration file directly, we have provided a text editor you can use without leaving the Administration Server. You can edit any scope in the scope hierarchy, and create scopes that will be reflected in the Administration Server pages. All server directives (settings) supported by Administration Server can be set and viewed in both the individual task pages or the Edit Configuration page. You can also use the Edit Configuration page to manually edit directives that are not supported in the Administration Server pages.

For details on these tasks, please refer to the online documentation available at the IBM Administration Server page.

9.2.6 DB2 UDB management

DB2 UDB is the main data repository for the WebSphere Commerce Suite, Marketplace Edition for AIX. Information is constantly written and read from the DB2 UDB database as the e-Marketplace members interact with the e-Marketplace site and perform their daily activities. Having a well-tuned and managed database has a profound effect on the overall performance of the e-Marketplace application.

DB2 UDB provides a set of administration tools that can assist you in managing the DB2 server(s). Yyou can administer database servers locally or from remote clients. These tools provide a graphical user interface for administration.

9.2.6.1 Administration tools

The tools for administering DB2 are part of the Administration Client, a selectable component with each of the DB2 Universal Database products. The Administration Client is also available on a set of CD-ROMs that include the Administration Clients for all the operating systems on which DB2 is available.

They allow you to install and use the Administration Client on any workstation. It does not matter whether your database servers are local or remote, or what operating system the database servers are running on. The tools enable you to perform the same functions from a graphical user interface as you could from the command line processor. These functions include the entering of DB2 commands, SQL statements, or system commands. With the tools, however, you do not have to remember complex statements or commands and you get additional assistance.

The following tools are available from the Control Center toolbar:

- The **Control Center**. The Control Center is the main DB2 graphical tool for administering your database. From the Control Center, you get a clear overview of all the systems and database objects that are cataloged locally.
- The **Satellite Administration Center**. The Satellite Administration Center allows you to administer DB2 satellite servers.
- The **Command Center**. The Command Center enables you to issue DB2 database commands, SQL statements, and operating system commands, recall previous commands, and scroll through access plans for SQL queries.
- The **Script Center**. The Script Center allows you to create, run, and schedule operating system level commands and DB2 command scripts.
- The **Alert Center**. The Alert Center notifies you when thresholds that you have set have been exceeded or when a node in a multinode environment is no longer responding.
- The **Journal**. The Journal allows you to view the status of jobs and to view the recovery history log and messages log.

- The **Information Center**. The Information Center gives you quick access to the information in the DB2 product manuals and sample programs and provides access to other sources of DB2 information on the Web.
- The **License Center**. The License Center displays the status of your license as well as allows you to configure your system for proper license monitoring.

For some functions that you can perform with the GUI tools, you are given the option of using a SmartGuide. SmartGuides are invoked from the pop-up menus in the Control Center. They provide a greater level of help by prompting you step-by-step on how to fill in the information necessary for the task you are doing and even making calculations and recommendations based on information you supply. SmartGuides are very useful if you are a new database administrator or someone who only administers a database occasionally.

In DB2 Universal Database, the following SmartGuides exist:

- Backup Database. This asks you basic questions about the data in the database, the availability of the database, and recoverability requirements. It then suggests a backup plan, creates the job script, and schedules it. To invoke the Backup Database SmartGuide, select the icon representing the database you want to back up, click mouse button 2, and select **Backup** -> **Database using SmartGuide**.
- Create Database. This SmartGuide allows you to create a database, assign storage, and select basic performance options. To invoke the Create Database SmartGuide, select the **Databases** icon in the Object Tree pane, click mouse button 2, and select **Create** -> **Database using SmartGuide**.
- Create Table. This SmartGuide helps you to design columns using predefined column templates, create a primary key for the table, and assign the table to one or more table spaces. To invoke the SmartGuide, select the Tables icon, click mouse button 2, and select **Create** -> **Table using SmartGuide**.
- Create Table space. This SmartGuide lets you create a new table space and set basic storage performance options. To invoke it, select the **Table Space** icon, click mouse button 2, and select **Create** -> **Table space using SmartGuide**.
- Index SmartGuide. Use the Index SmartGuide to determine which indexes to create or drop for a given set of SQL statements. The recommendations are based on the workload that you specify. To invoke the Index

SmartGuide, select the **Indexes** folder, click mouse button 2, and select **Create** -> **Index using SmartGuide**.

- Performance Configuration. This SmartGuide helps you tune databases by requesting information about the database, its data, and the purpose of the system. It then recommends new configuration parameters for the database and instance and automatically applies them if you wish. To invoke this SmartGuide, select the icon for a database, click mouse button 2, and select **Configure using SmartGuide**.
- Restore Database. This SmartGuide walks you through the process of recovering a database. To invoke the SmartGuide, select the icon for a database, click mouse button 2, and select **Restore** -> **Database using SmartGuide**.
- Configure Multisite Update SmartGuide. This SmartGuide lets you configure databases to enable applications to update multiple sites simultaneously when it is important that the data at all the sites must be consistent. To invoke this SmartGuide, select an instance, click mouse button 2, and select **Multisite Update** -> **Configure using SmartGuide**.

Besides the graphical tools that you can invoke from the Control Center toolbar, there are some additional GUI tools that are not invoked directly from the Control Center toolbar.

- Performance Monitor. Performance Monitor is a tool to monitor DB2 objects such as instances, databases, tables, table spaces, and connections. You use this tool to detect performance problems and tune databases for optimum performance. The Performance Monitor is invoked as a selection on the pop-up menus in the Control Center.
- Event Monitor. Event monitor is a tool that lets you analyze resource usage by recording the state of the database at the time specific events occur. An Event Monitor is created by typing `db2emcrt` from a DB2 command line.
- Event Analyzer. Event Analyzer is a tool that allows you to analyze the data collected by the Event Monitor. An Event Analyzer is invoked by typing `db2evmon` from a DB2 command line.
- Visual explain function. The visual explain function lets you view the access plan for SQL statements as a graph so that you can tune your SQL queries for better performance. Prior to Version 6, you used the Visual Explain tool to view the access plans. In Version 6, visual explain is no longer a separate tool; however, the function is available on pop-up menus from various database objects in the Control Center, and also from the Command Center.

In addition to these tools, another useful tool for database administration that is not part of the Control Center is the Client Configuration Assistant. The Client Configuration Assistant is a SmartGuide with the primary function of setting up communications from remote clients to servers.

All these tools are described in greater detail in the online Administration Guide documentation.

9.3 Security guidelines

Once your application is running in production mode, you can expect a lot of user access to the system. If we lived in a perfect world where all users were law abiding, then we would not have to worry about security. Unfortunately, this is not the case and your system will constantly face security threats, both external and internal.

The site you have developed is only one component of the total system configuration. The security strength of your system is only as strong as its weakest link. Thus, it is necessary to ensure that the other components in the system are configured securely.

With this in mind, end-to-end security will consist of physical, operating system, network and application security. SeeTable 9 for an end-to-end security component list.

Table 9. End-to-end security components

Security Type	Description
Physical	Control access to the hardware equipment hosting your application.
OS	Security at the operating system level.
Network	Secure connectivity flow between external, DMZ and internal networks.
Application	Configure WebSphere security.

9.3.1 Physical systems security

Physical systems security is the foundation of the end-to-end security building blocks. Access to the hardware has to be controlled and monitored proactively. Anyone gaining unauthorized physical access to your servers could halt your server, steal valuable information from your storage, plant viruses, install harmful software, etc. All of these activities are disruptive to your operations and will cause damage to your system.

If the hardware is not secured properly it will void the other security measures you take.

9.3.2 Operating systems security

After securing your physical systems, you will have to work on securing the operating system. As the OS grows richer in function and features, new bugs are discovered or are waiting to be exploited (for example, a bug that allows a user with non-privileged access to perform privileged operations.)

At the operating system level, we recommend the following practices for administering your system:

1. Keep yourself updated on new security glitches.

   Unfortunately, there are public Web sites that provide detailed information about newly discovered glitches. Fortunately, there is also a wealth of public information available that provides temporary or permanent remedies for these glitches. As an administrator, you need to be warned quickly about these loopholes and to take immediate actions to rectify the problem. As a service to their customers, most OS vendors provide updated information related to security hacks. A good source of information is the CERT Web site (`http://www.cert.org`). You can subscribe to their mailing lists to receive regular updates and news flashes by e-mail.

2. Access privileged accounts.

   Your OS security policy will have to consider who has access to the privileged accounts. You will have to determine the roles and level of responsibility each person has. For example, you may want to separate the role of an OS administrator from the application administrator.

3. Enforce good password policies.

   Many security hacks are the result of simple passwords. You will have to enforce good password policies and practices for all accounts in your system, whether they are privileged or non-privileged. Besides having this policy, educate your users on their role in the overall system security. Another policy that you could implement, at the OS level, is forcing the passwords to expire or force them to change after some predefined period. Also you should not allow reuse of passwords.

4. Enable logging and auditing.

   Remember to turn on OS system logging and auditing. In the event of a system break-in, hopefully you will have some trails to start off your investigation.

9.3.3 Network security

Once you have the physical hardware and the operating system secured, you need to turn your attention to security between interconnected systems.

Network security is the act of protecting resources residing in your internal network and demilitarized zone (DMZ) from the external network. You want to restrict and prevent unauthorized user access to your internal systems. At the same time, you do not want to make it difficult for legitimate users to access your systems.

The key technologies available to achieve this network protection include firewalls, intrusion detection monitors, anti-virus detection and the way you implement them. The SecureWay suite of products provide you with a comprehensive security solution that will meet most requirements. Information about SecureWay products can be found at

`http://www.ibm.com/software/secureway/`

9.3.3.1 WebSphere in a firewall environment

How do you restrict and control network access? You can use firewalls in between two networks. When properly configured, the firewall acts as a check point and will force all traffic of a specific protocol to and from the Internet to flow through it. By doing so, it can then scan the traffic and determine whether to allow or disallow the packets based on a set of rules.

When designing your firewalls take the following into consideration:

1. Make sure that there is no direct communication channel between the applications on the intranet and the external Internet. In our example, all external user requests and application responses flow through the Web server residing in the DMZ. If necessary, WebSphere Commerce Server will forward the request to the DB2 UDB in the internal network.
2. Keep it short and simple. Reuse pre-configured rules that exist. Define new rules if necessary. Always remember to include a rule that excludes everything else.
3. You should not allow information pertaining to the internal network to reach the Internet. For example, you would not want the IP addresses of your internal systems to be made available to external users. Hiding this information will reduce the risks of external security hacks, you can use Network Address Translation to accomplish that.
4. At a minimum, you will need a firewall between the external network and your DMZ, and a firewall between the DMZ and the intranet. Introducing a

DMZ configuration creates an additional security barrier, which a network infiltrator would have to overcome.

***** Note*****

When you implement this DMZ configuration, it is possible to implement two or three firewalls in the same physical machine with two or three network adapter cards. This is usually only done for cost saving purposes and is not a preferred solution.

To implement a recommended topology such as the one shown in Figure 14 on page 43, you would typically place the WebSphere Commerce Server in a DMZ. This will separate the Web Application from your corporate data stored in your DB2 UDB which resides within your internal network behind a domain firewall implemented on a separate physical machine.

9.3.3.2 Internet and intranet security considerations

The intranet environment may be a LAN-based departmental network or could span geographic regions via a WAN-based Virtual Private Network (VPN). With this distinction in mind, we can see that the WAN-based intranet environment has similar characteristics to the Internet based systems. The physical network used in the VPN network is usually outside your management control. Thus, you should continue to focus on network security.

For a LAN-based intranet that is segmented along departmental domains, you could still implement a firewall between the various departments. A good example would be to separate the production network environment from the development network environment.

9.3.4 Web application security

The user requests will flow through your firewall to your application. The final security checkpoint would be application security, which will decide who can invoke specific application function.

WebSphere provides an integrated security model to configure security for your Web resources. This can be centrally configured through the WebSphere Application Server Administrative Console.

Let us begin by describing the various WebSphere security components listed in the Table 10.

Table 10. WebSphere security components

Security Component	Location
Security plug-in	Supported Web server
Security collaborator	WebSphere Application Server
Security server	Security application

9.3.4.1 Security plug-in

When Web clients try to access Web resources deployed by the WebSphere Application Server, the security plug-in component will be invoked. The security plug-in will then send the client requests to the security collaborator for security decisions. This security plug-in is installed and configured with the Web server during software installation.

The following table lists the respective Web servers that WebSphere Application Server currently supports.

Table 11. Supported WebSphere Web servers

	AIX	SOLARIS	NT
Apache Server V1.3.6	X	X	X
Netscape Enterprise Server V3.51 and V3.61	X	X	X
Lotus Domino Application Server R5	X	X	X
Domino Go Webserver R4.6.2.5 and R4.6.2.6	X	X	X
IBM HTTP Server V1.3.6	X	X	X

9.3.4.2 Security collaborator

The security collaborator is attached to the WebSphere Application Server. It makes security decisions on remote method calls on servlets or EJBs by performing:

- Authorization checks
- Pre- and post-security trace logging
- Delegation policy enforcement

9.3.4.3 Security server

The security server is found in the security application. The security application resides in every WebSphere Administration Server. It provides the means to configure security policies for both Web resources and EJBs. In a particular WebSphere Application Server domain, there is a shared repository on the database server (DB2, Ooracle), which contains all the security configuration and policy information. Any WebSphere Application Server in the same WebSphere domain can access this shared repository.

Both the security plug-in and security collaborator will call the security server for authentication/authorization services. It provides:

- Centralized control over security policies (permissions, delegation)
- Central security services (authentication, authorization)

The security server is a trusted third party for security policy and control. Web servers and WebSphere Application Servers call on the security server to provide authentication, authorization and delegation services.

9.3.4.4 Secure the WebSphere Commerce Suite system

Additionally you can take the following recommendations to secure the WebSphere Commerce Suite system:

1. **Protect configuration files**.
   Configuration files contain the host name, database name, encrypted password, and merchant key. If the Web server allows access to WebSphere Commerce Suite configuration files, they can be visible through Web browsers and become a security hazard. This information is so sensitive that WebSphere Commerce Suite provides a method to encrypt it. Please refer to IBM HTTP Server configuration manual for information on how to accomplish that.
2. **Disable samples and documentation**.
   To increase the security of your production site, remove any databases used for testing or educational purposes.
3. **Specify security level for commands**.
   Security levels define whether SSL security and login authentication are required to run a particular Commerce Suite command. Access control policies stored in LDAP are also used to secure command execution authorization. Use your Web-based interface (ncadmin) to manage Command security and Marketplace Commands. The command security level can only be changed by the e-Marketplace administrator. Choose the appropriate option below:
   - Assign security levels for commands.
   - Change the security assignment.
   - Remove the security assignment.

9.3.5 WebSphere security model and policy

We have described the key infrastructure components in WebSphere security. Next, we would like to describe the WebSphere security model and policy. Understanding this allows you to make informed decisions in configuring and managing WebSphere security. For example, you can decide which of the authentication methods are supported given a particular user registry.

Specifically, we will describe the authentication, authorization and delegation models and policies within WebSphere.

9.3.5.1 Authentication

Authentication is the process of proving a user is who he says he is. In WebSphere, authentication between a user and the WebSphere Application Server can be specified in terms of:

- **User registry**

  This is where the user and group information will be stored.

- **Authentication mechanism**

  After the user has provided the required data, the authentication mechanism will validate it against an associated user registry. Two types of authentication mechanisms are supported:

  - Lightweight Third Party Authentication (LTPA)
  - Native operating system

- **Challenge mechanism**

  The challenge mechanism specifies how a server will challenge and retrieve authentication data from the user. It can be of the form:

  a. **None** - Security runtime does not challenge user for authentication data.

  b. **Basic** - A user is challenged for ID and password.

  c. **Certificate** - Mutual authentication over SSL.

  d. **Custom** - The ability to specify a custom HTML page to retrieve a user’s ID and password.

Note

The components of the authentication are dependent on one another. For example, the authentication mechanism is defined based on the user registry and this choice drives the challenge mechanism.

In Table 12, we illustrate the relationship between the user registry and the authentication mechanism. If user ID and password are supplied, then authentication is delegated to a user registry. If digital certificates are used, then the certificate credentials are mapped to an associated user registry entry.

Table 12. Mapping between authentication mechanism and user registry

	UNIX	**Windows NT**	**LDAP**
Native OS (user ID, password)	The supplied password is encrypted using the OS's crypt facility. This is then compared against the system's password repository.	Authentication is delegated to the NT Security Access Manager via systems call.	N/A
LTPA (user ID, password)	N/A	N/A	An LDAP bind is performed using the DN (Distinguished Name) and the password.
LTPA (digital certificate)	N/A	N/A	Based on the trust in the Web server, certificated are validated through successful establishment of a mutual SSL connection. A credential mapping is then performed based on the information contained in the certificate.

9.3.6 HTTP single sign-on (SSO)

When you enable HTTP single sign-on, user authentication credentials are preserved across multiple applications in the same domain, for example:

- Cooperating but disparate Web servers
- Cooperating applications like the IBM On Demand Server and Windows NT suites.

With SSO, your application will then avoid repeated requests of user security credentials. However, if your application wants to use the SSO feature, then it must use an LTPA registry (LDAP for example).

9.4 Backup and recovery guidelines

Backup may seem a mundane and repetitive task you perform routinely, but it is absolutely necessary. Its importance to you is never emphasized enough and typically you will only realize it during a disaster. Imagine losing valuable transactional data due to a hard disk failure and you do not have a backup.

We suggest you consider the following factors when considering a backup solution:

1. Data to backup

   You should consider the solution’s support for the various data you need to backup. The data includes operating systems, application data, transaction logs, configuration files, application databases, and WebSphere Application Server repository databases.

2. Available backup window time

   In most situations, there is a limited window of time to complete the backup. Thus, you will have to consider the performance of the backup solution. Consider the performance of the solution as a whole, not the individual pieces.

3. Required system recovery time

   Not only should the backup be fast, but the recovery process should be equally fast. Consider how the backup solution is able to provide fast recovery.

4. Support for enterprise backup

   The solution should be scalable to perform backup of new systems that you may install, as a result of growth and upgrades. You may also want to use the backup solution for other existing applications.

5. Integration with system management tools

   It is very useful if the backup solution can be integrated with existing system management tools and thus provides a central administration capability.

6. Support for emerging technology

   The software should be able to support emerging storage area network (SAN) based storage solutions. As your information needs grow, these storage solutions will provide large-capacity and high-performance data access.

9.4.1 Using Tivoli Storage Manager (TSM)

Based on the above factors in selecting a comprehensive backup solution, we recommend the IBM Tivoli Storage Manager (TSM). It is an integrated storage management solution that will meet the needs of any company, from small Internet startups to large enterprises. TSM provides the following features:

- Full support of client platforms
- High performance backup and recovery process
- Wide support for IBM and non-IBM tape/optical technology
- Scalable solution to meet growing storage demands
- Support for SAN-based solution
- Integrated with the Tivoli suite of systems management products

Let's begin by looking at how you can configure a Tivoli Storage Manager solution. We recommend that each and every system in your configuration be backed up. The frequency of backup will vary, depending on the type of information and the frequency of changes.

Figure 37 shows the recommended Tivoli components for deployment in the internal network nodes.

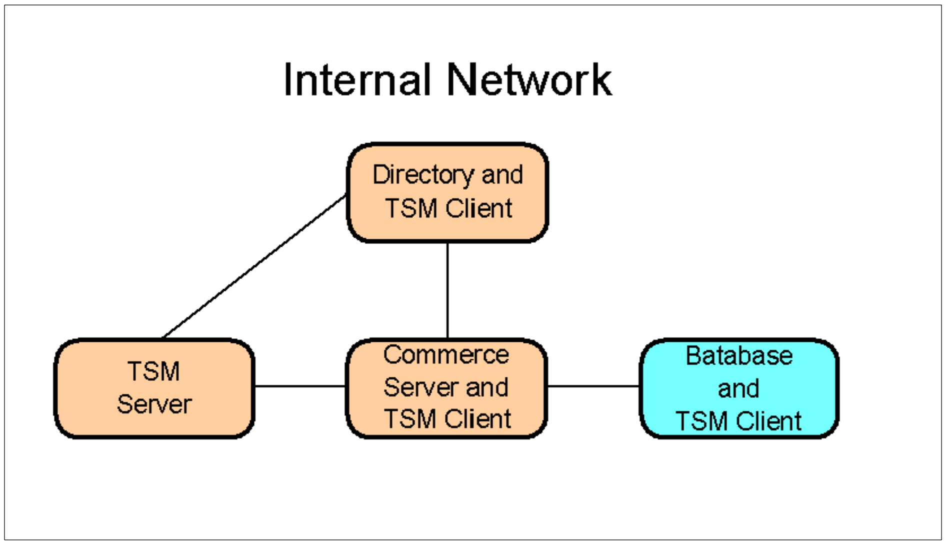


Figure 37. TSM server and client setup in the internal network

Referring to Figure 37, let's assume you install one TSM server in the scenario. This server should be located in the internal network with no external Internet access to it.

Once the TSM server setup is completed successfully, install TSM clients at every system that you would like to back up. For example, we have installed TSM clients at the directory server, Commerce Server and the shared file system server. Test the connectivity between the TSM clients and the TSM server by doing a user-initiated backup.

For the servers in the DMZ (see Figure 38 on page 186), you can also install a TSM client. To facilitate communication between the TSM client in the DMZ and the TSM server in the intranet, you will have to open up one port in the firewall. This port can be preconfigured in both the TSM client and server.

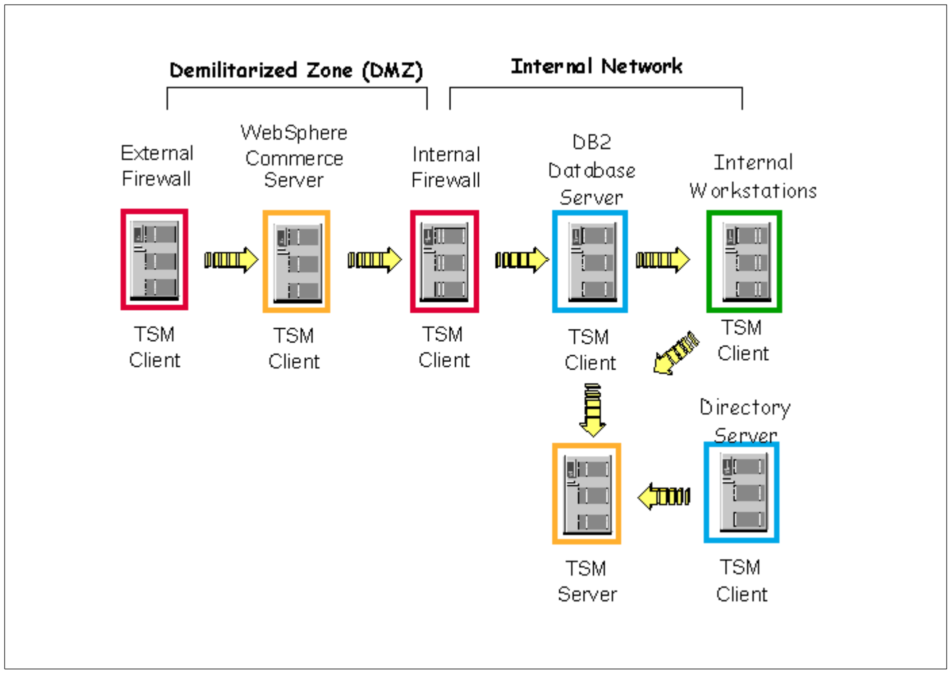


Figure 38. TSM server and client setup in the DMZ and internal network

If this additional firewall port is a security issue in your environment, then there are two viable options to consider:

1. Install Tivoli Data Protection for Workgroups (TDPfW) for Windows NT systems.

   TDPfW provides a stand-alone disaster recovery for Windows NT machines. It can back up entire Windows NT machines or volumes, and if a disaster occurs, TDPfW can restore the complete machine, including the boot volume, disk partitions, security, operating systems, and user files from a locally attached SCSI tape drive.

   This is very useful for small LAN environments (for example, the DMZ shown in Figure 39) where you have to manage only a few servers. However, this product currently is supported on Windows NT only.

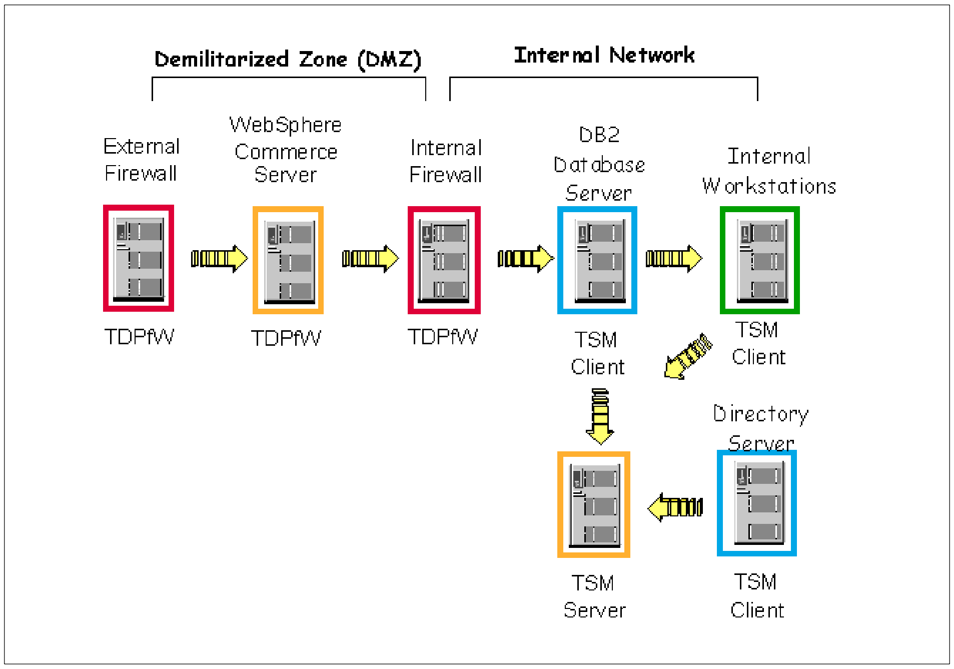


Figure 39. TSM server in the internal network with TDPfW in DMZ

2. Installing a TSM server in your DMZ

   To facilitate a more automated solution, you could install another TSM server in the DMZ. Keep in mind, that as more servers are added to the DMZ, the overhead involved in administering each TDPfW server increases.

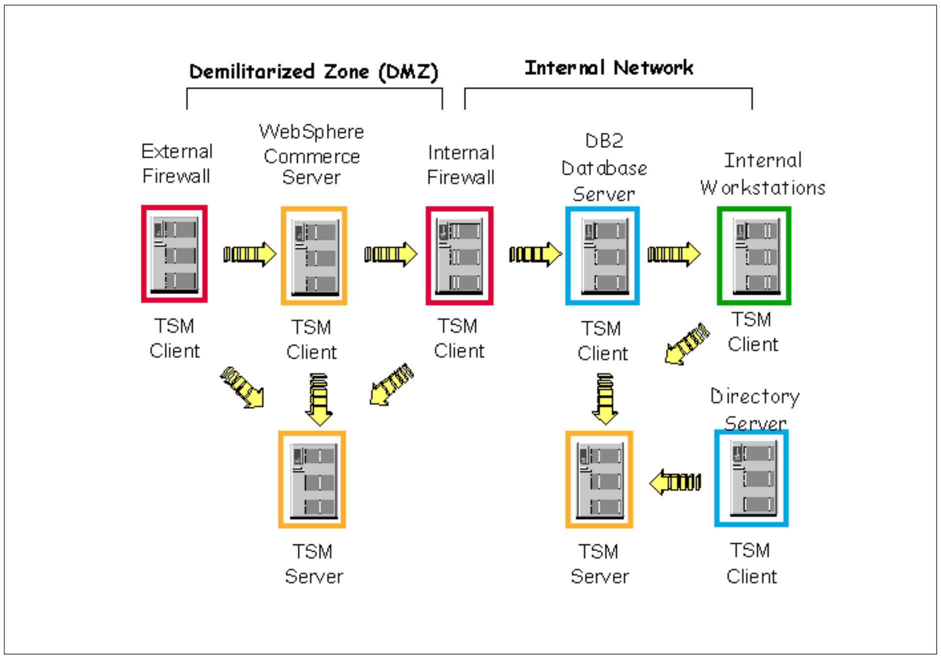


Figure 40. One TSM server in the DMZ and each internal network

9.4.2 Application backup and recovery

We have discussed general architecture for developing a backup solution. In this section, we will highlight product-specific areas to look out for.

9.4.2.1 Operating system

In a total system recovery scenario, the operating system (OS) will be the first software to be reinstalled. You should refer to your OS instruction manual for re-install steps. Our checklist below highlights practical tips for OS backup management:

- Do you have and know where the OS media is located?
- Do you have new updates to the OS?
- Do you have an OS backup after initial system setup?
- Do you do regular OS backup?
- Do you do ad-hoc OS backup before major installation and after major configuration changes?

9.4.2.2 WebSphere database

When you install and configure the WebSphere Application Server, WebSphere Commerce Suite, SecureWay Directory and WebSphere Site

Analyzer, you will notice that they use DB2 as their database. In this section, we will only briefly illustrate how a DB2 backup can be performed. For a comprehensive review of DB2 backup and recovery, please refer to the DB2 administrative guides and references.

In DB2, you can perform either an online or offline backup. When you do an offline backup, the database needs to be shut down. The command for an offline backup is:

```
db2> backup db <instance name> offline=yes
```

If shutting down the database is not an option, then you will have to perform an online database backup:

```
db2> backup db <instance name> online=yes
```

When performing an online database backup, you will have to take into consideration how the database logs will be managed, as they need to be used in a recovery process. Losing the logs will complicate the recovery process. When you use TSM, you will enjoy an automated database log backup solution. This can be achieved by:

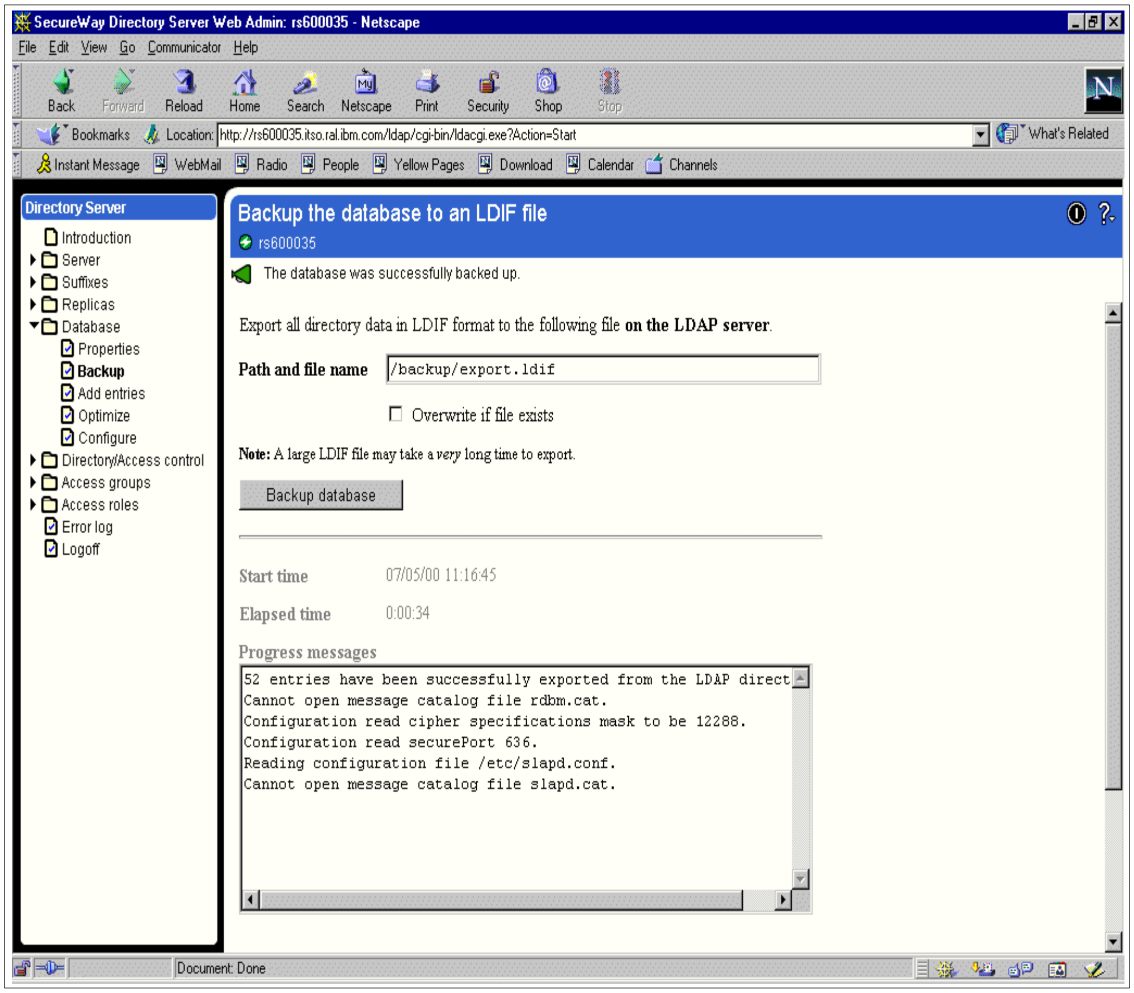
- Configuring the DB2 user exit program to utilize TSM.
- Turning on the user exit and log retain parameters in DB2 database manager.

9.4.2.3 LDAP database backup

With the IBM SecureWay Directory, you can back up the LDAP database from the GUI interface, the command line, or via native DB2 commands.

Using the GUI (see Figure 41 on page 190):

1. Click **Database** in the Navigation frame.
2. Click **Backup** in the Working with back-end database in the work area.
3. Type the fully qualified name of the file to be created in the text entry field. This file will be in LDIF format.
4. Click the **Backup database** button to start backing up the database.


SecureWay Directory Server Web Admin: rs600035 - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Shop Stop
Bookmarks Location: http://rs600035.itso.ral.ibm.com/ldap/cgi-bin/ldacgi.exe?Action=Start
What's Related
Instant Message WebMail Radio People Yellow Pages Download Calendar Channels
Directory Server
Introduction
Server
Suffixes
Replicas
Database
Properties
Backup
Add entries
Optimize
Configure
Directory/Access control
Access groups
Access roles
Error log
Logoff
Backup the database to an LDIF file
rs600035
Backup database
Document: Done

The database was successfully backed up.

Export all directory data in LDIF format to the following file **on the LDAP server**.

Path and file name /backup/export.ldif

☐ Overwrite if file exists

Note: A large LDIF file may take a *very* long time to export.

Start time 07/05/00 11:16:45

Elapsed time 0:00:34

Progress messages

```
52 entries have been successfully exported from the LDAP direct
Cannot open message catalog file rdbm.cat.
Configuration read cipher specifications mask to be 12288.
Configuration read securePort 636.
Reading configuration file /etc/slapd.conf.
Cannot open message catalog file slapd.cat.
```

Figure 41. LDAP database backup GUI

Using the command line, the db2ldif tool can be used to dump entries from a directory to a text file in LDIF format. The syntax is:

```
db2ldif -o <output file> [-s <subtree>]
```

Note

LDIF format does not honor any ACL settings and all data is available in readable text.

To use the DB2 database backup, see 9.4.2.2, "WebSphere database" on page 188.

9.4.3 Guidelines for backup and recovery

Independent of your backup software choice, there are some guidelines you can follow:

1. Monitor the backup process.

   Ensure that the backup process is successful. If the backup process fails, understand why it fails and take necessary actions to remedy. For example, is the backup failing due to dirty drives or faulty media? Perhaps, there is a problem with the hardware? Could it also be a loss of connectivity between your backup server and the client machines?

2. Properly manage the backup media.

   After taking a backup of the system, ensure you have a proper and systematic tracking system for your backup media. In the event of a disaster, can you can easily identify and retrieve these backup media for recovery?

3. Test your backup.

   Do not assume that if the backup process is successful, you have a valid backup. Test your backup! This will give you additional confidence that your backup can really help you recover from a disaster.

4. Document, test and maintain a recovery plan.

   No matter what size your system configuration may be, always have a documented recovery plan. This will prove useful during a disaster situation. The plan should describe step-by-step the process you will take to recover. Test the plan to ensure that it works! For the document to be useful at all times, you will have to maintain and update it when configuration changes occur.

Part 3. Business-to-Business Patterns: e-Marketplace example

© Copyright IBM Corp. 2000

Chapter 10. Marketplace Edition overview

The aims of this section are to provide you with a high-level understanding of the WebSphere Commerce Suite, Marketplace Edition architecture and design. We describe the objectives that the Marketplace Edition is designed to meet and discuss the architecture of the system and it components. More detailed information about the components of the Marketplace Edition can be found in the subsequent chapters of Part 3, “Business-to-Business Patterns: e-Marketplace example” on page 193.

We describe the objectives behind the design and development of the Marketplace Edition, provide an overview of the system architecture, and discuss the main sub-systems that make up the full Marketplace Edition.

10.1 Marketplace Edition objectives

WebSphere Commerce Suite, Marketplace Edition for AIX is based on WebSphere Commerce Suite 4.1. However, some significant modifications have been made to support the functions required of a B2B e-Marketplace trading hub. In making these modifications the goal of the Marketplace Edition is to create a hub that enables B2B commerce among multiple buyers and suppliers. The Marketplace Edition achieves this by providing an unified view of the products and services traded within the e-Marketplace and by providing a variety of mechanisms to facilitate trade of these products.

At the heart of the Marketplace Edition is an aggregated catalog that provides different views of the products available for trade.

The Marketplace Edition will allow organizations which are market members to define roles and simple approval workflows based on these roles, in order to control and customize their interaction with the e-Marketplace.

Price determination within the e-Marketplace will allow:

- Sellers to offer goods at a fixed price
- Sellers to hold an auction for goods within the catalog
- Buyers to submit a Request for Quote (RFQ)
- Sellers to respond to a RFQ
- Sellers to use an exchange mechanism to post available products, quantities and asked prices

© Copyright IBM Corp. 2000

- Buyers to use an exchange mechanism to post desired products, quantities and bid prices
- The hub to match exchange positions of buyers and sellers

Marketplace Edition will also offer reports for the hub operators and for buyers and sellers.

10.2 The Marketplace Edition players

Figure 42 on page 196 shows the various players involved in an e-Marketplace and it is the objective of WebSphere Commerce Suite, Marketplace Edition for AIX to provide facilities for all these various roles.

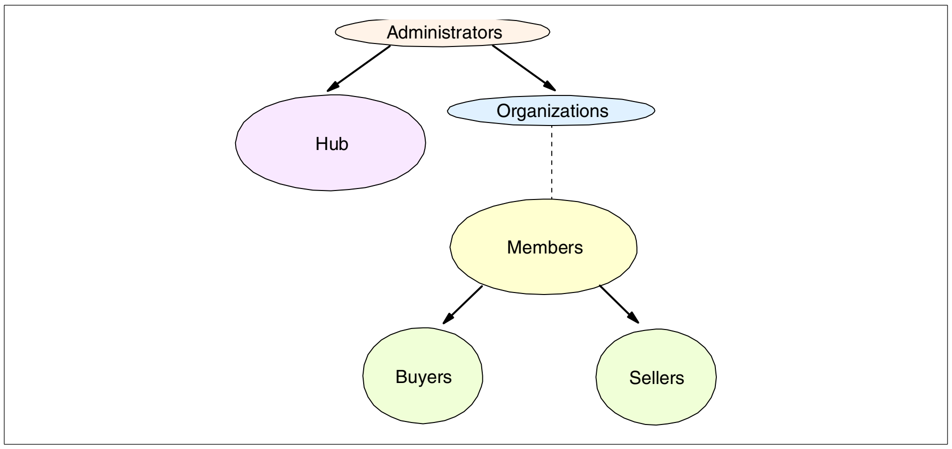


Figure 42. WebSphere Commerce Suite, Marketplace Edition - players

The players associated with the Marketplace Edition can be grouped into two categories:

- Administrators

  There are two types of administrator within Marketplace Edition. The first type is the Hub Administrator who is analogous to the Site Administrator in WebSphere Commerce Suite 4.1 and administers tasks such as roles, catalog taxonomies, induction of new organizations joining the hub and other day-to-day administration functions.

  The second type of administrator is the Organization Administrator who is primarily responsible for the administration of organizational members and their roles.

- Members

Members are the buyers and suppliers associated with organizations registered with the e-Marketplace.

10.3 The Marketplace Edition trading process

The trading process implemented in the Marketplace Edition extends the online buying options available in the WebSphere Commerce Suite 4.1, which currently offers a shopping cart mechanism and support for auctioning. In Marketplace Edition, we can now purchase products through additional methods of RFQ and exchange. Also, the price we pay for a product can be determined by a *contracted price* which may be arrived at as the result of an RFQ.

Figure 43 on page 197 provides a high-level summary of how buyers and sellers can interact with Marketplace Edition.

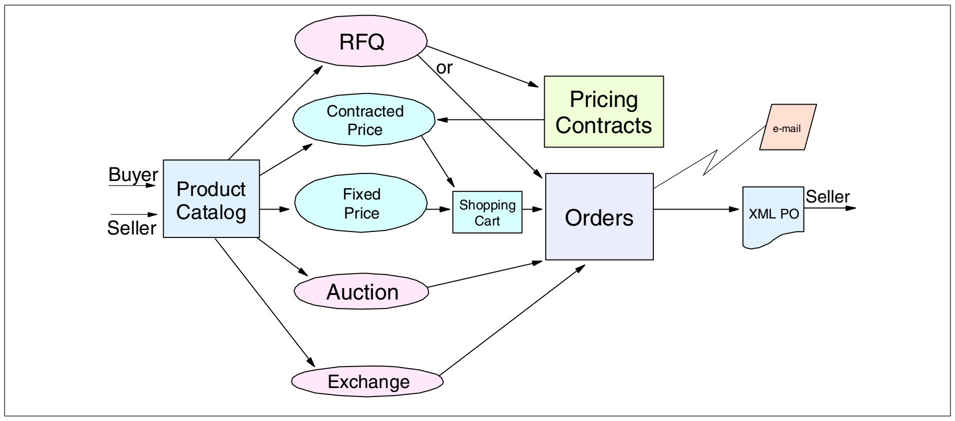


Figure 43. Overview of buying and selling in Marketplace Edition

When a buyer selects a product from the catalog, it can be purchased through the conventional shopping cart mechanism or it can be purchased via *negotiation*. The result of the negotiation process is an order or if an RFQ is performed, the result is a new contracted price for future orders.

For order management, Marketplace Edition uses the base order management capabilities of WebSphere Commerce Suite 4.1. For the conventional shopping cart ordering, the existing order system is used. The auction, RFQ and exchange systems interface to the existing WCS order system. When an order is placed, the supplier(s) are notified via e-mail. The supplier(s) can then look at the order or download it in XML format. Once the supplier has accepted the order, notification is sent to the buyer.

10.4 MarketPlace subsystems

The architecture of the WebSphere Commerce Suite, Marketplace Edition for AIX is made up of a number of subsystems and infrastructure components. It is based on and uses the WebSphere Commerce Suite. Figure 44 shows the main subsystems that are part of the Marketplace Edition.

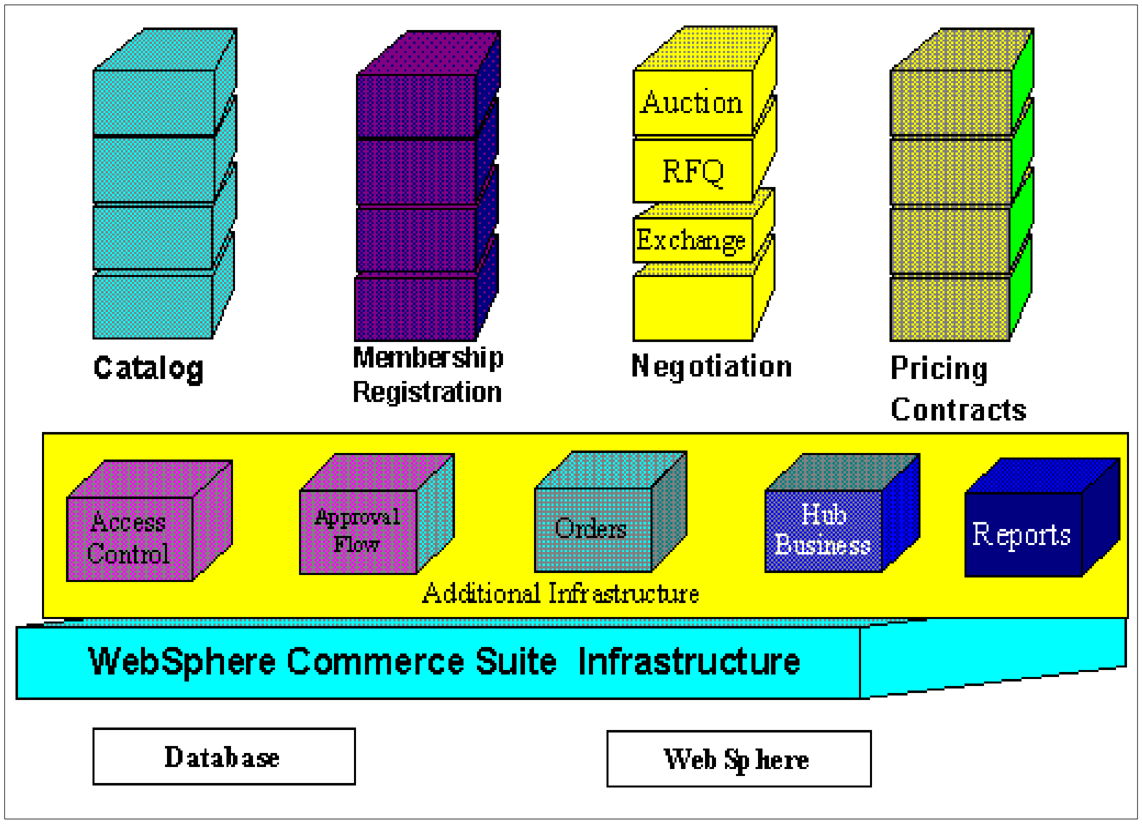


Figure 44. Marketplace Edition subsystems

10.4.1 Catalog subsystem

The goal of the Marketplace Edition design for catalogs is to provide a unified view of the products traded in the e-Marketplace to both buyers and sellers. The catalog must capture both supply and demand positions expressed by the sellers and buyers in the e-Marketplace. To this end we must logically separate the notion of things (that is the description of products) and their trading positions as expressed by the various parties in the e-Marketplace.

The WebSphere Commerce Suite 4.1 catalog is the basis for the Marketplace Edition catalog. The WebSphere Commerce Suite catalog is comprised of database tables for categorizing and describing products as well as import and maintenance processes. Structurally, the Dynasty catalog is largely similar to the WCS catalog. The catalog usage and maintenance processes are quite different due to the B2B nature of the product.

The Marketplace Edition catalog subsystem provides the following features:

- Unified view of the products traded in the marketplace
- Flexible product categories
- Buyer and supplier specific views
- Buyer administration approval of catalog content
- Centralized catalog for different negotiation methods allows multiple methods of trading simultaneously
- Pricing by WebSphere Commerce Suite mechanism, negotiation, or contracts
- Data loading by:
  - Mass import via XML
  - Marketplace Edition aware version of Catalog Architect
  - Spreadsheet import
  - Web browser
- Logging of catalog events

Chapter 14, “Example - catalog subsystem” on page 269 provides a fuller discussion of the catalog subsystem in the Marketplace Edition and provides examples how the subsystem works in practice.

10.4.2 Membership subsystem

This feature provides functionality for B2B membership and captures the information required by the rest of the Marketplace Edition subsystems including:

- Organizational structures of a member
- Roles
- Authentication
- Contact details
- Address

- Commerce preferences

The facilities provided by the membership and registration subsystem as implemented in WebSphere Commerce Suite, Marketplace Edition for AIX include:

- Aassociates people with organizations and record organizational hierarchy
- Defines roles and attributes for members and organizations
- Provides additional information about members and organizations required at registration
- Market administrators are the site admin
- Organization administrators are store admin
- Logs membership events.

Chapter 13, “Example - membership and access control” on page 241 provides a fuller discussion of the membership subsystem in the Marketplace Edition and examples of how the subsystem works in practice.

10.4.3 Negotiation subsystem

The negotiation subsystem consists of the following:

- Auctions - which is based on the WebSphere Commerce Suite V4.1 Auction system with minor accommodations for the Marketplace Edition table extensions
- RFQ - which is a totally new system with no equivalent in the WebSphere Commerce Suite
- Exchanges - which extends the buying and selling process to provide such features as Orderbook, TradingPost and Matching Components

All of the above events will be logged to be able to provide a complete audit trail of any and all negotiation processes. This is necessary for dispute resolutions.

The supported negotiations are:

1. Auctions

   This is the WebSphere Commerce Suite 4.1 functionality that provides auction interactions, bid control, pricing, closing rules and policy variation. Tools available to buyers are:

   - Auction gallery

- Proxy bid
- E-mail, bulletin board notification

Tools available to sellers are:

- Auction setup
- E-mail, bulletin board notification

2. RFQ

The RFQ process allows both buyer and seller organizations to develop a request for either the sale or the purchase of goods. The Marketplace Edition will then match the buyer request to a seller opportunity or a seller opportunity to a buyer request.

If a buyer is searching a catalogue for goods or items and they cannot find the exact item or the price that they wish to pay, then they initiate a process known as a Request for Quote (RFQ). The RFQ can be launched publicly or to selected bidders who can respond to the buyer in order to fulfill the request. In composing the details of the RFQ, the buyer can elect to allow certain fields within the description of the object to be able to be modified such that interested sellers can make changes to the offer with which they respond. E-mail will be generated to targeted sellers, alerting them to the fact that a new RFQ is available for them to quote on.

Buying organizations will be able to evaluate the responses and select winning sellers. The RFQs can be used to negotiate a long-term contract, in which the prices, quantities, and dates agreed to between the selected sellers and unique buying organization can be recorded and enforced in subsequent purchases.

Items generated for the RFQ are added to the RFQ Interest List, these items can be added from an existing catalog or they can be generated from scratch if no such cataloged item currently exists.

The buyer may either specify the RFQ to be open to all e-Marketplace members or may target the RFQ to a specified list of recipients.

Recipients can view a list of RFQs targeted to them, select a particular RFQ to view, and if desired, create a response to the RFQ. Recipients can also download the RFQ in XML format.

The buyer closes the RFQ, evaluates responses, and selects a set of winners. When the winners are selected, the buyer has two choices:

- Immediately issue orders for the item(s) (one for each seller chosen)
- Establish a contract with the seller(s) based upon the RFQ response

The RFQ process will allow the buyers to:

- Create RFQs
- Using catalog product descriptions
- Using newly created product descriptions
- Target RFQs
- Publish RFQs
- View responses and select winners
- Copy RFQ
- Modify close or retract RFQ
- Search and view RFQs and responses on predefined attributes
- Create an order or contract from winning responses

The RFQ process will allow the suppliers to:

- Search and view the RFQs targeted to the supplier, or RFQs that are public
- Create responses
- Submit response
- Modify and retract responses
- Search and view responses and results of responses

3. Exchange/matching

Many business transactions entail the procurement of a product or service available from multiple sources. Quite often, the buyer's purchasing decision is made on the basis of which supplier can provide the product at an acceptable price within a specified time frame. The exchange negotiation model provides buyers and sellers with the functionality to transact business by matching a buyer's price and delivery requirements with the available offerings for that product.

The buyer's bids and seller's offers are placed in an order book, which is accessible to all authorized hub members. Once the bid or offer is placed, the matching of bids and offers is done by a predefined matching procedure. When the matching procedure finds a matched buyer-seller pair, both parties are notified and the order is processed by the Marketplace Edition order processing system. In summary, the exchange model provides buyers and sellers with a dynamic pricing environment inherent in a bid and offer process where:

- Buyers and sellers to express their bid/ask positions against the same product type

  - An order book used to persistently capture bid/ask positions
  - A matching component is used to match bids and offers ("clearing") in real-time, both continuously and periodically

Chapter 16, “Example - negotiation subsystem” on page 367 provides a fuller discussion of the negotiation subsystem in the Marketplace Edition and examples of how the subsystem works in practice.

10.4.4 Pricing contracts subsystem

The goal of the Marketplace Edition design for contracts is to provide a mechanism to store agreed upon prices between sellers and buyers. Such agreements may arise from other interactions in the e-Marketplace, such as negotiations or RFQ. The contract facilities provided by the Marketplace Edition include:

- Right to buy contracts
- Obligation to buy contracts
- Price or discount for all purchases or for specific items
- Manually created or created from the RFQ process
- Queried by catalog based buying
- Notified by order management
- Logging of contract events

Figure 45 on page 204 show how these contract facilities interact with the other parts of the Marketplace Edition.

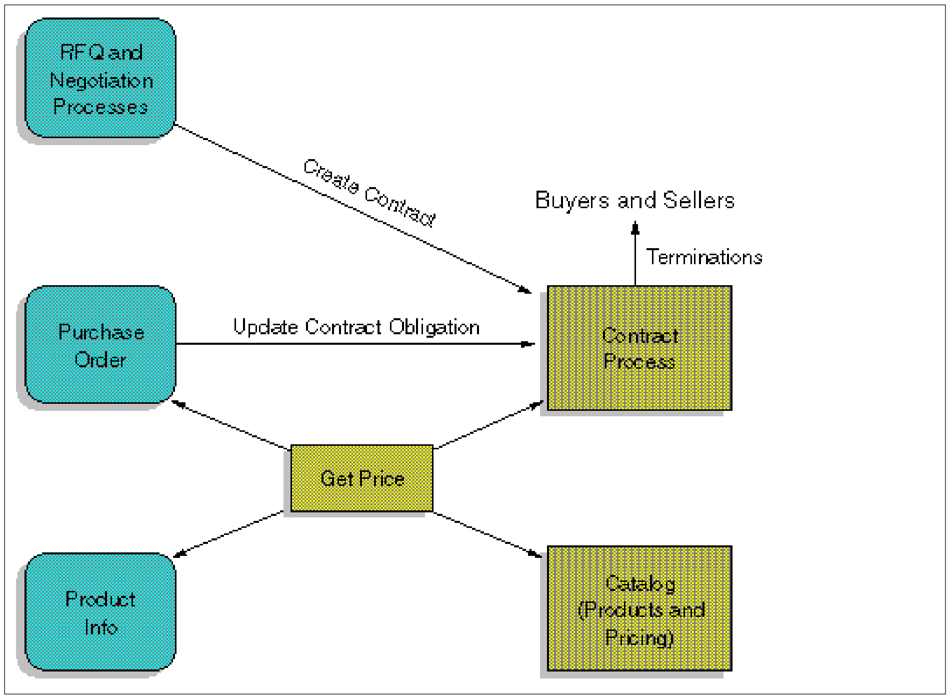


Figure 45. Interaction of the contract process with other parts of an e-Marketplace

The buyer and the seller negotiate a price or a discount on the published price for a particular product. One instance of this is through the RFQ process. This new price now applies when the buyer orders that product. The pricing in the Marketplace Edition must pick up this price.

The following details are included in the contract:

- Seller
- Buyer
- Product
- Price
- Buyer organization, seller organization
- Creator and co-creator
- Min. monetary amount: obligation to buy contracts
- Max monetary amount: right to buy contracts
- BeginTime: The date and time at which the contract begins

- End time: The date and time at which the contract terminates (if not terminated earlier).
- Coverage: All purchases (“applicable to all products”) or specific items (“special pricing condition”). If you select **special pricing condition** in Coverage, click **Manage Item** to add items from Current Contract Items.
- Currency
- Attachment
- Turnaround time:
- Products or categories
- Contract price
- Minimum volume or quantity: The buyer is required to place orders whose quantities add up to at least the minimum quantity. If the MiniQuantity value is not specified, there is no minimum quantity order requirement.
- Maximum volume or quantity: The seller is required to accept the agreed price until the buyer's orders have quantities summing to the MaxQuantity value. If the MaxQuantity value is not specified, there is no maximum quantity order limitation.

The administrators of the e-Marketplace can list contracts including current quantity ordered against the contract and the remaining obligation. Any combination of a buyer, a seller, and any number of products can be specified.

The buyers can list contracts for the requesting buyer including current quantity ordered against the contract and the remaining obligation. A seller and/or any number of products can be specified.

The sellers can list contracts for the requesting seller including current quantity ordered against the contract and the remaining obligation. A buyer and/or any number of products can be specified.The seller can terminate a contract if no EndTime and no MaxQuantity values are specified.

A contract can be terminated in one of the following ways:

- If an end time is specified, the contract ceases when the end time is reached.
- If no EndTime and MaxQuantity values are specified, the seller may explicitly cancel at any time.
- The e-Marketplace administrator can cancel at any time.

- An agreement between both the buyer and the seller can cancel the contract.
- If MaxQuantity is specified, the contract ends if, at any time, the sum of the quantities of orders placed by the buyer under the contract reach the MaxQuantity.

Note: If both an EndTime and a MaxQuantity are specified, then the contract is terminated when either one of the two is reached.

Upon the contract termination (which can occur by either cancellation or time/quantity expiration), both the buyer and seller need to be notified by the notification means specified in their user profile.

Chapter 15, “Example - pricing and contract subsystem” on page 341 gives more details of the way contracts are implemented in WebSphere Commerce Suite, Marketplace Edition for AIX.

10.4.5 Additional infrastructure

This section deals with the extra facilities proved by the Marketplace Edition to support the infrastructure of its e-Marketplace.

10.4.5.1 Hub business

The hub business facility is more or less equivalent to the financial side of the Marketplace Edition. Reports are obtained and processed through this subsystem.

At the core of any e-Marketplace is the ability of all the hub participants to obtain current information on activity within the hub. The *role* of the user wishing to generate a report determines the reports the user can access.

Hub-level reports dealing with revenue and membership will be available in XML format to facilitate access both on and off-line. Organization-level reports providing summary listings and statistics for the catalog, membership, RFQ, exchange and contract subsystems will also be available.

For more details of the features provided by the hub business facilityplease see Chapter 17, “Example - additional e-Marketplace infrastructure” on page 441.

10.4.5.2 Reports

Each subsystem has defined a limited set of available reports that can be generated on demand.

Member organization reports can be generated for catalog statistics, RFQs, exchange, and contracts.

Further details of the reports produced by Marketplace Edition is in Chapter 17, “Example - additional e-Marketplace infrastructure” on page 441.

10.4.5.3 Orders

As the name suggests, the orders subsystem manages the order process including:

- Managing the orders table
- Notifying sellers of pending purchase orders(PO)
- Allowing sellers to download an XML version of the PO
- Updating an order when a seller receives a PO and notifying the buyer
- Managing rejected POs

An order is placed by a member of the e-Marketplace and in the Marketplace Edition implementation of orders it can be associated with only one seller organization. An order consists of a number of linei tems which may be created from contract, fixed price, RFQ, auction, or exchange offerings.

Chapter 17, “Example - additional e-Marketplace infrastructure” on page 441 discuss the order facilities of Marketplace Edition in more detail.

10.4.5.4 Access control

The access control infrastructure provides the Marketplace Edition with the ability to:

- Define user groups
- Specify user attributes that define implicit user groups
- Define product groups
- Sselect attributes that define implicit product groups
- Ddefine access policies that map user groups to other groups.

The Marketplace Edition provides a number of standard roles and associated access policies to manage default access to the e-Marketplace. An access policy in its simplest form is a relationship of a group of users with a command, a group of resources and a role relative to the resources. A command is also known as an action and it may be a basic function such as view, modify, create, or delete, a market level function such as quote, buy, or requestQuote.

These roles include:

- Administrator

  Hub level access to all tables and commands
- Organization Administrator

  Able to access tables and commands for her own organization as allowed by the administrator
- Buyer

  Access allowed to own tables and the buyer commands allowed by the organization
- Supplier

  Access allowed to own tables and the seller commands allowed by the organization
- Broker

  Access allowed to exchange tables and exchange commands

The Marketplace Edition provides tools to allow definition and maintenance of roles and access control policies.

Chapter 13, “Example - membership and access control” on page 241 gives more information about the access control facilities of WebSphere Commerce Suite, Marketplace Edition for AIX.

10.4.5.5 Approval flow

The Marketplace Edition implements a single-level approval flow facility so that selected commands in the e-Marketplace can be subjected to an approval process. This allows administrators to specify which action require approval and to chose who the approvers should be. The Marketplace Edition approval facility will notify approvers that an action is waiting for their approval. Multiple approvals are not currently implemented by the Marketplace Edition.

Approvers may accept or reject the pending approval. If they accept the command awaiting approval will be executed. If the approver rejects the action the command will not be executed. E-mail is sent to notify interested parties whether the command was accepted or rejected.

Chapter 17, “Example - additional e-Marketplace infrastructure” on page 441 provides more details on the approval flow facilities provided by the WebSphere Commerce Suite, Marketplace Edition for AIX.

10.5 Marketplace Edition programming model

The WebSphere Commerce Suite, Marketplace Edition for AIX is built on a base of the WebSphere Commerce Suite 4.1 so it still uses the legacy C++ environment from the WebSphere Commerce Suite, but new Marketplace Edition functionality is implemented in a Java programming model, as this is the strategic direction for WebSphere Commerce Suite products.

Figure 46 shows a high level view of the architecture of the WebSphere Commerce Suite, Marketplace Edition. Auctions, catalog buying and order management are sourced from the legacy C++ model code with few changes, whereas new Java code is utilized for contracts, exchanges, RFQ and reporting. Catalog, catalog buying, and auctions also have some Java code in order to provide new functionality or to interact with the C++ facilities.

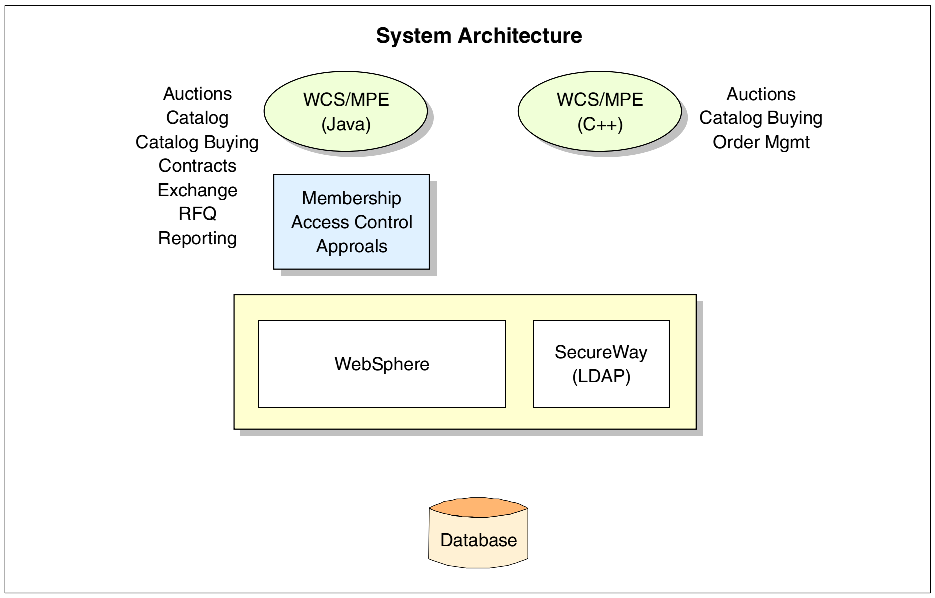


Figure 46. WebSphere Commerce Suite, Marketplace Edition for AIX system architecture

The Marketplace Edition Java programming model has as its base components:

- Interaction controllers

  Interaction controllers control interaction with a specific client type. Since the current implementation of Marketplace Edition is designed for browser-based clients all current interaction controllers are Java Servlets.The job of interaction controllers is to receive client requests and process any parameters from the client. The interaction controllers do not themselves process the request, as this is delegated to the correct command or commands. Once the request processing is complete the interaction controller passes result beans back to the client. In the current implementation of the Marketplace Edition, the result beans are part of a JSP user interface client

- Commands

  Commands are a unit of business logic such as GetPrice or AddToCart.They have an interface and an implementation.Commands are created by a factory that instantiates an implementation of the interface. The command interface provides setter methods to pass data to the command, and execute method to initiate the command action and setter methods to retrieve result from the command.

- Persistent objects

  Persistent objects represent an encapsulation of a row in a table and allow the management of persistent data by mapping beans to tables.

Figure 47 on page 211 shows a view of the Marketplace Edition Java programming model.

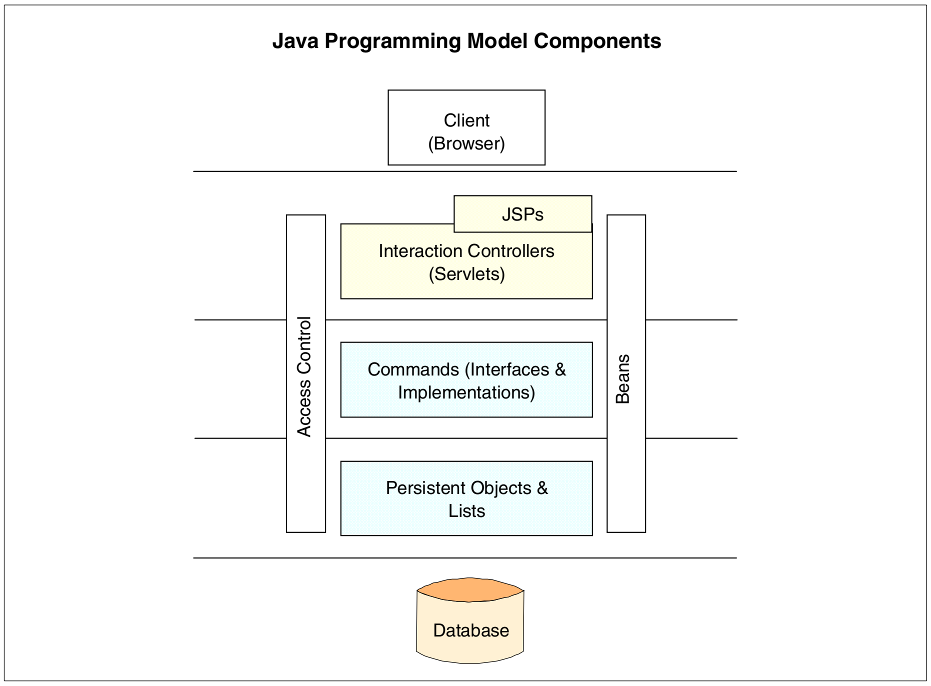

Java Programming Model Components
Client (Browser)
JSPs
Interaction Controllers (Servlets)
Access Control
Commands (Interfaces & Implementations)
Beans
Persistent Objects & Lists
Database

Figure 47. Java programming model

More details about the Java programming model used in the Marketplace Edition can be found in Chapter 7, “Application design guidelines” on page 89.

10.6 Example application

In the example chapters in the remainder of Part 3, “Business-to-Business Patterns: e-Marketplace example” on page 193 we illustrate the implementation of the Marketplace Edition using the standard example application that was shipped with the pre-release code used in writing this redbook. This example application illustrates the features and functions available to an e-Marketplace that uses the WebSphere Commerce Suite, Marketplace Edition.

It uses a fictional example of an e-Marketplace centered around a worldwide shipbuilding marketplace. In Appendix A, “Marketplace Edition installation guide” on page 455 ,we give instructions for installing this example application.

Chapter 11. Example - runtime environment

This chapter discusses the configuration we used in the implementation and testing of the WebSphere Commerce Suite, Marketplace Edition for this redbook.Because the WebSphere Commerce Suite, Marketplace Edition was still being developed when we were developing this redbook, we were not able to implement the Marketplace Edition on more that one system. Therefore for our testing we installed the IBM HTTP Web server, DB2, LDAP WebSphere Commerce Suite, Marketplace Edition, WebSphere Application Server, Advanced Edition, Net.Data and WebSphere Commerce Suite all on one system. This is not the preferred way of implementing a production system. There are several ways that a system can be implemented that will work in a production environment. The following lists several ways of doing a production implementation:

1. Everything on one system except the data

   This would be a good implementation if the system had very low traffic or if the system was going to be used only for a select number of consumers. The reason why this system would not be used in a high value e-Marketplace is because the e-Marketplace could very well become to much for the system to handle and cause the system to be come unstable. Also in a double system configuration there is no redundancy, so if there was a system failure your users would not be able to access your e-Marketplace.

   Figure 48 shows a schematic of this double system implementation.

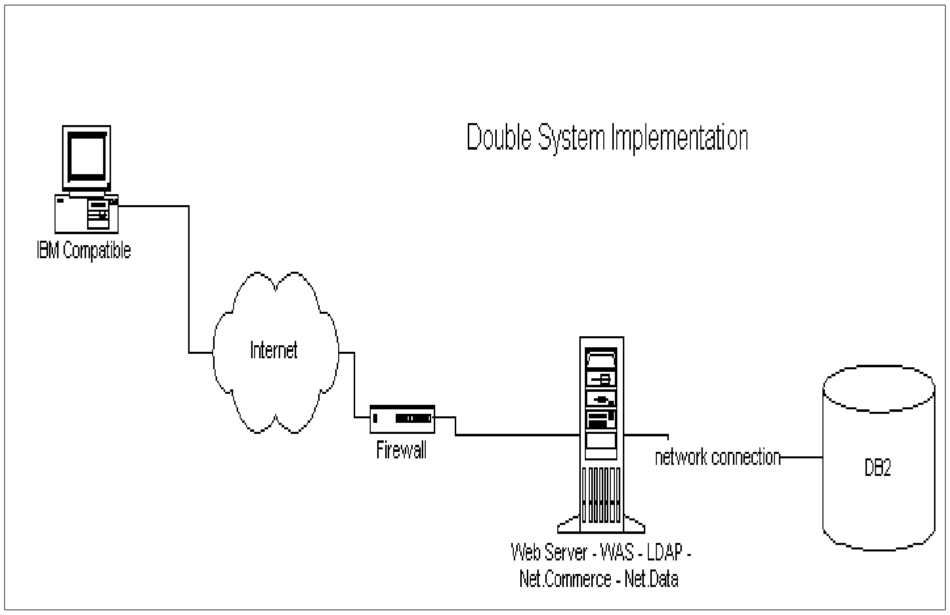

Double System Implementation
IBM Compatible
Internet
Firewall
network connection
DB2
Web Server - WAS - LDAP - Net.Commerce - Net.Data

Figure 48. Double system implementation

2. Separate Web server and database server.

   This triple system implementation is more robust than the first one. In this implementation you would separate the Web server from the application server and keep the database on a separate machine. This would allow for more processing to be handled on the application server, and the data processing would be handled completely by a more robust database server. Figure 49 shows how this can be implemented.

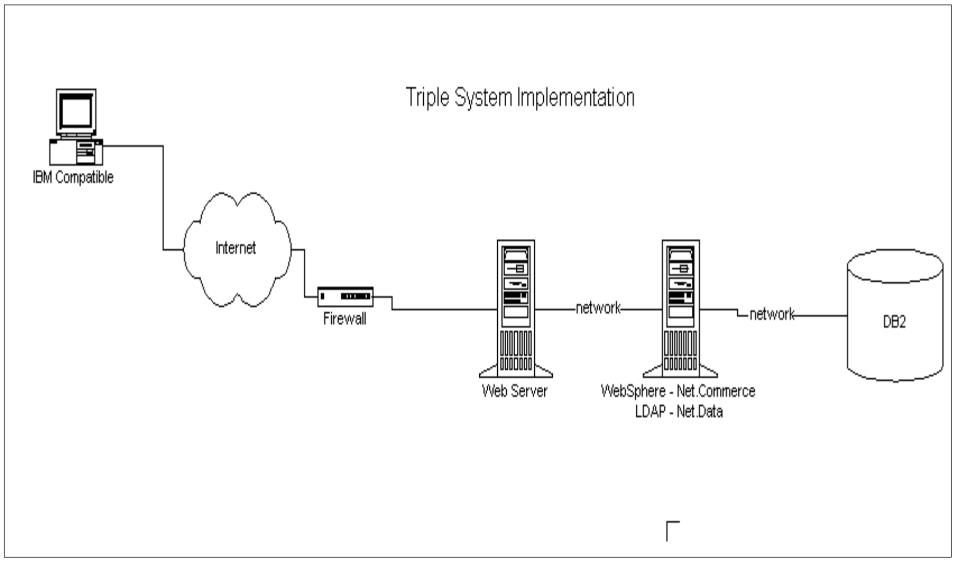


Figure 49. Triple system implementation

3. Build a farm out of each system\

   The last implementation would be a completely robust and load-balanced implementation. This can be implemented by building a Web server farm for the Web servers, building an application server farm for the application servers, and building a database farm for the database servers. This is by far the most robust implementation and the most reliable implementation. One thing to keep in mind is before you implement the WebSphere Commerce Suite, Marketplace Edition for AIX, make sure that you size all of the systems for the traffic that will be using the application. Figure 50 on page 216 shows the farm implementation.

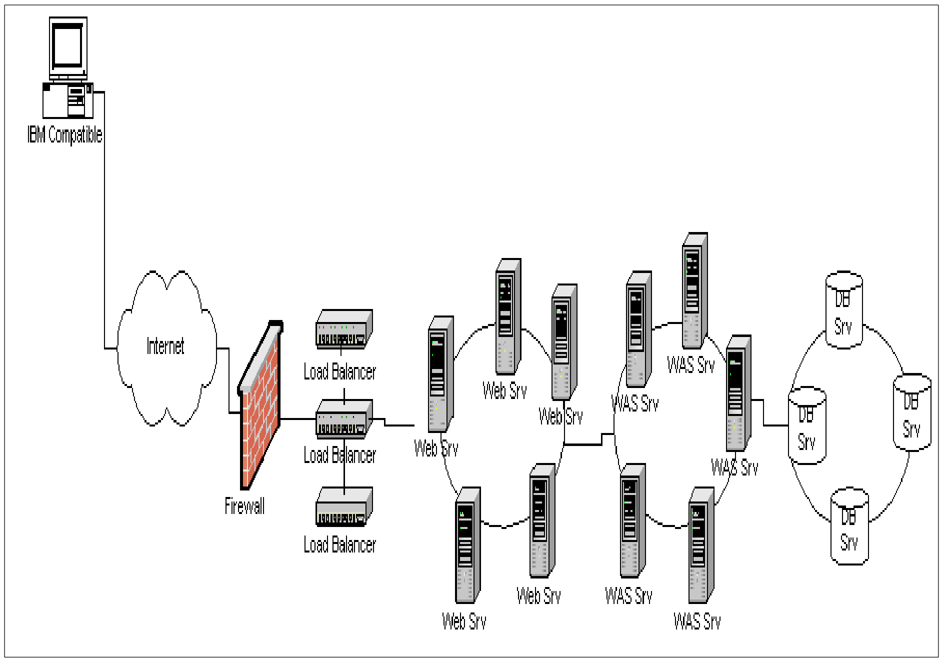


Figure 50. Farm implementation

11.1 Hardware implementation

The hardware configuration that we used in the writing of this redbook was an IBM RS6000 7043-150D 375 MHZ 604e with 196 MB of SDRAM, but this was solely for our testing purposes and is not recommended for production implementations.

11.1.1 Minimum hardware requirements for implementation

The minimum hardware requirements for implementation are based on the minimum hardware requirements for the WebSphere Commerce Suite. A dedicated RISC System/6000 or an IBM Power System Series family of machines (RS/6000 Model C20 or higher) is recommended with the following:

- A 167 MHz processor
- A minimum of 128 MB (256 of random access memory (RAM) recommended
- A minimum of 900 MB of free disk space
- A CD-ROM drive

- A Graphics capable monitor.

To access the Commerce Suite Administrator, a workstation capable of running Windows NT, Windows 98, or Windows 95, with a graphics-capable monitor with at least 256 colors, and a mouse or other pointing device, is needed.

A local area network (LAN) adapter that is supported by the TCP/IP protocol would also be needed.

11.2 Hardware recommendations

The following is the recommended hardware for the different production options:

- 1-tier system (database and Web Server on the same machine)

  For the Web and database server:

  - 4-way F50 (166 MHz PowerPC 604e
  - 2 GD RAM
  - 1 drive for the operating system and the applications
  - 1 drive for paging
  - 8 drives for the database
  - 1 drive for the database logs
  - 1 drive for the Web Server and the Commerce Suite logs.
- 2 - tier system (single Web server connected to remote database)

  For the Web Server:

  - 4-way F40 (233 MHz PowerPC 604e
  - 512 MB RAM
  - 1 drive for the operating system and the applications
  - 1 drive for the paging
  - 1 drive for the Web server and for the Commerce Suite Logs.

  For the database server:

  - 4-way F50 (166 MHz PowerPC 604e)
  - 2 GD RAM
  - 1 drive for the operating system and the applications
  - 1 drive for the paging

- 8 drives for the database
- 1 drive for the database logs

• 3 - tier system (multiple, load balanced Web servers connected to a remote database server)

For each Web server:

- n times 2-way F40 (233 MHz PowerPC 604e)
- 512 MB RAM
- 1 drive for the operating system and applications
- 1 drive for paging
- 1 drive for the Web Server and for the Commerce Suite log.

For the database server:

- 4 -way F50 (166 MHz PowerPC 604e)
- 2 GB RAM
- 1 drive for the operating system and the applications
- 1 drive for the paging
- 8 drives for the database
- 1 drive for the database logs.

11.3 Prerequisite software

Ensure that you have AIX 4.3.2 or 4.3.3 on your WebSphere Commerce Suite, Marketplace Edition for AIX machine. The following fixpacks must be implemented:

- Apply PTF U453695 for CSet++ miscellaneous runtime fixes.
- Ensure that the C runtime xlC.rte is at least at Version 3.6.4.1.
- Netscape Communicator 4.61 (a copy is provided on the WebSphere Commerce Suite CD, in the \Netscape directory)

11.4 Web server configuration

Theminimum requirement for the IBM HTTP Server that is based on Apache HTTP Server is a system that is running AIX 4.2.1 with xlc.rte 3.1.4.8 or higher.

11.5 WebSphere Commerce Suite configuration

Chapter 1 in the IBM WebSphere Commerce Suite Pro Edition for AIX v 4.1 Installation Guide provides a list of prerequisite hardware and guidelines on recommended hardware for production systems.

11.6 JDK implementation

The IBM JDK1.1.8 is required for the WebSphere Commerce Suite and for the WebSphere Commerce Suite, Marketplace Edition for AIX.

Chapter 12. Example - development environment

While no development tasks were conducted in the production of this Redbook, this chapter provides details on the major development tools that would be utilized in a typical Marketplace Edition development environment.

Essentially, the tool set is similar to that provided by the WebSphere Commerce Suite 4.1. Because the majority of software development under the Marketplace Edition is centered on Java-based technologies, IBM provides an extremely solid foundation for this development using VisualAge for Java and the WebSphere Commerce Studio.

In addition to covering the major development tools used to develop Marketplace Edition sites, we also provide instruction on how to import the sample e-Marketplace, “Worldwide Shipping Marketplace” into WebSphere Commerce Studio so that you can work with, modify and subsequently redeploy elements of the application for your own educational purposes. THis process is valuable if you want to gain a better understanding of the structure and elements of a Marketplace Edition application.

12.1 Development environment overview

The development environment required to build, deploy and maintain a Marketplace Edition site is very similar to that used in WebSphere Commerce Suite 4.1 development projects.

In WCS 4.1, a typical development environment included tools which could be classified as follows:

- General development tools
  - WebSphere Studio
  - Page Designer and HotMedia
  - VisualAge for Java
  - Platform specific C++ compiler
  - Text editors for Net.Data macros
- Product and shopping related tools
  - Product Advisor
  - Shopping Metaphor Builder
  - Catalog Architect

© Copyright IBM Corp. 2000

Most of these tools can still be utilized in the Marketplace Edition environment.

12.2 Development platforms

As with most Internet developments, many of the components of applications developed for the WebSphere Commerce Suite, Marketplace Edition for AIX can be developed from a variety of platforms including Windows, AIX and Linux (as well as others, such as OS/2). This is due to the cross-platform capabilities of many of the IBM WebSphere development tools, which is making the choice of development platform truly open.

Typically development is conducted from the following environments:

- Windows

  Most of your Marketplace Edition development can be done from a Windows-based workstation using the WebSphere Commerce Studio tool set and VisualAge for Java. You can then deploy the Java code, HTML and JSP resources to the target platform using WebSphere Studio, which provides both file system-based publishing and FTP-based publishing services.

- AIX

  For compilation of any C++ code, you can use the AIX-based C/C++ tool sets such as C Set++ for AIX to compile and link libraries such as those created for overridable functions.

- Linux

  IBM is committed to porting developer tools for Linux and already provides Linux versions of DB2, VisualAge for Java, HomePage Builder, WebSphere Application Server, and IBM HTTP Server. You may choose to use Linux for a development of application components using any one of these tools.

What is important is that you are not bound to any single development platform and are free to exploit the features of the platform most suited to you.

12.3 Development tools

This section provides a high-level view of the tools available to assist in the development and deployment of your Marketplace Edition solutions. Although some of the components operate only under Windows NT, such as WebSphere Commerce Studio, many of them support other development platforms. Becoming more prevalent in the developer community is development from the Linux platform and, as described in the previous section, it is a platform supported strongly by many IBM products.

There are a number of good references for WebSphere Studio and VisualAge for Java development, including *Servlet and JSP Programming with IBM WebSphere Studio and VisualAge for Java*, SG24-5755.

A number of HTML and PDF format documents are also provided in WCS 4.1 and we expect that full documentation of the Marketplace Edition tools will be provided upon product launch.

12.3.1 WebSphere Commerce Studio

The WebSphere Commerce Studio provides a workbench from which you can manage, edit and deploy many of the assets in your e-Marketplace.

Typically, you use the Commerce Studio to manage:

- JSPs and Java code used in your site.
- Graphics files and HTML files to be deployed to your site.
- Macro files (Net.Data .D2w files).
- Any other scripting or media files that form part of your production or staging sites.

The Commerce Studio provides many features for both the single developer and team development including:

- Full integration to VisualAge for Java using the Toolset API.
- Support for a shared project directory supporting team development.
- Additional development tools such as WebSphere Page Designer and HotMedia.
- A number of wizards to assist in JavaBean creation, SQL statement generation and automated creation of JSP pages based on these SQL statements.
- Full support for publishing the assets of the site via file system copy or FTP copy. The developer can configure publishing servers and target folders to fully control the publishing of site assets to staging or production servers.

Refer to 12.5, “Importing the sample site into WebSphere Studio” on page 229 to learn how we used WebSphere Studio 3.02 to import, inspect and modify files used in the Marketplace Edition sample application.

12.3.2 Page Designer

WebSphere Page Designer is a fully-featured HTML and JSP page designer used to create static and dynamic content for the Web site. Page Designer has full support for JavaBeans and supports JSP Versions 0.91 and 1.0 (this setting is configured from the WebSphere Studio).

12.3.3 HotMedia

The HotMedia software is included in the WebSphere Studio to assist designers in creating multimedia components for use in their e-Marketplaces. It is based on Java applet technology and does not require a browser plug-in to display components delivered to the user's browser. Some of its features include the ability to create:

- Graphics that rotate, pan and zoom.
- Streaming audio and video.
- Various transitional effects for graphic files.

12.3.4 VisualAge for Java

VisualAge for Java is the IBM Java development environment and is considered to be one of the best available. VisualAge provides a fully featured development, debugging and test environment in which to develop your own interaction controllers, command beans, data beans and view beans to create your e-Marketplace applications.

One feature of VisualAge for Java that is very useful in developing Java code in the WebSphere environment is the WebSphere Test Environment (WTE). In the WTE, VisualAge creates an environment that simulates a production WebSphere server, allowing you to test your code as if you had published it to a WebSphere server. The primary benefit of using this environment is that you can use the VisualAge for Java Integrated Debugger to debug your applications locally on your development machine.

While not tested, it is possible to configure the WTE to debug your Marketplace Edition applications.

12.3.5 Catalog and shopping tools

As expected, you can use the HTML tools provided in the Marketplace Edition Administration Console to manage the information stored in the aggregated catalog of the e-Marketplace. These browser-based tools are hosted by the hub and offer supplier-side organizations the ability to update their own products, which will subsequently be displayed in the aggregate catalog. Also available is the ability for each organization to provide the product and catalog data in XML formatted scripts.

Accordingly, it is advisable to have a good XML editor to assist you in creating and debugging these scripts.

12.3.6 Catalog Architect

Catalog Architect was available in WCS 4.1 and offers us the ability to modify catalog and product data from a fully featured GUI environment. This product has been revised for the Marketplace Edition to support the modifications made in areas such as the database schema.

12.3.7 Database tools

If your e-Marketplace requires you to extend the default Marketplace Edition database schema, you may require GUI tools to assist you. If you using DB2 for AIX, the Control Center and Command Center tools provide assistance in this area. You may also require tuning and capacity planning tools in addition to these base tools.

12.3.8 Changes between WCS 4.1 and Marketplace Edition

While most tools used in development of WCS 4.1 projects will work under the Marketplace Edition, there are a number of tools available for use under WCS 4.1 that are either not supported by he Marketplace Edition or must be modified to support it. In addition, the Marketplace Edition provides a number of new capabilities.

- Product Advisor

  This will no longer be supported under the Marketplace Edition.

- Store creator wizard

  At the time of writing, there was no evidence that a similar tool will be available to generate a base-level e-Marketplace.

- Catalog Architect

  A revised version of Catalog Architect will be available that will support catalog administration by the hub administrator and also by organizations (to a limited extent).

- Microsoft Excel spreadsheet

  New support has been added to import catalog data from a Microsoft Excel spreadsheet.

12.3.9 C++ compiler

As in WCS 4.1, you may be required to develop some C++ code if you are implementing logic at the command, task and overridable function layers. Because this code is machine-specific, development must be conducted on the target platform, in this case AIX 4.3.3.

The tools that can be used to develop and compile AIX libraries are:

- C Set++ for AIX Version 3.1.4.
- IBM C and C++ compilers Version 3.6.4.

For more information on customization using the C++ programming model refer to the online documentation provided with the Marketplace Edition or the existing WCS 4.1 documentation:

WebSphere Commerce Suite, Pro Edition Customization Version 4.1.

12.3.10 Development tool interactions

Figure 51 shows the interactions and relationships between the various development tools used in a Marketplace Edition development project.

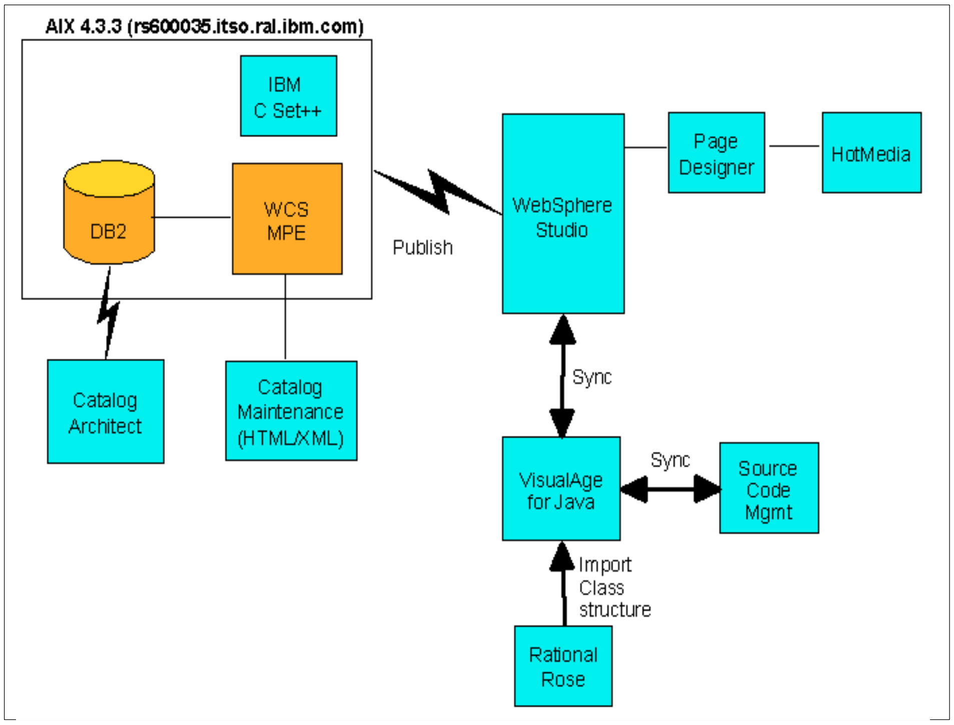


Figure 51. Interactions between common development tools

As a developer, you will need to become familiar with the location of files used by the Marketplace Edition. The base directory where Marketplace Edition files reside is:

```
/usr/lpp/CommerceSuite
```

From this directory a number of subdirectories exist where the core files used by the Marketplace Edition reside, and where modified and new files will be published to from the WebSphere Studio. The core directories are shown in Table 13; however, there are many other directories that may be the target directory for files in your Studio projects.

Table 13. Significant directories for a publish operation from WebSphere Studio

File types	Directory
Java code	`/classes/com/ibm/commerce/emp/<DIR>`
HTML/JSP	`/html/en_US/emp/<DIR>`

File types	Directory

Images	`/html/en_US/emp/standard/Images`
	`/usr/lpp/CommerceSuite/html/en_US/base`
	
	(Note: there are directories other than these)

Access Policies	`/emp_schema/singlebyte/accesscontrol`

12.4 Roles

The typical, but not exhaustive, list of roles commonly used in the development of e-Marketplaces is shown in Table 14.

Table 14. Typical roles seen in large-scale e-Marketplace development

Role Name	Description
Page Designer/Site Flow	Uses WebSphere Studio and Page Designer to build static HTML pages and dynamic JSP pages.
Content Contributors	A role grouping that provides additional services such as graphic design, content writing, and initial setup of the Marketplace base catalog.
WebMaster/Marketplace Administrator	Provides administrative services for the site such as implementing Site Analyzer and Performance Pack in addition to other site administration tasks.
Site Analyst	Reviews data from Site Analyzer and the MPE reporting subsystem to gain understanding of usage patterns with the aim of identifying areas of improvement.
Analyst	Gathers user requirements, prepares use cases and supplies the business rules on which the e-Marketplace will be founded. Also analyzes the requirements of suppliers and buyers participating in the e-Marketplace.
Programmer	Uses WebSphere Studio, VisualAge for Java and C++ tools to develop the business logic for the site.

Role Name	Description
Site Assembler	Uses WebSphere Studio to deploy the site to staging and production environments. Uses Site Analyzer to verify linkages and locate structure problems.
Integration Specialist	Responsible for the analysis and implementation of the integration of buyers and suppliers. Implements message transport services such as those provided by MQ Series.
Infrastructure Specialist	Responsible for the hardware, clustering, failover, performance and security of the e-Marketplace.
Architect	Ensures the successful coupling of the pieces in the solution, from infrastructure to applications.

12.5 Importing the sample site into WebSphere Studio

This section describes how to import the Marketplace Edition sample site "Worldwide Shipbuilding Marketplace" into WebSphere Studio. We have provided this example so that you can work with the example JSP pages and other files to gain further insight into Marketplace Edition application structure.

To perform the import, you must have WebSphere Studio 3.02 or WebSphere Commerce Studio installed on your computer and we assume that you have a running installation of the WebSphere Commerce Suite, Marketplace Edition for AIX with the example e-Marketplace installed and operational.

12.5.1 Importing files using the Import Wizard

The following steps describe how to import the files into WebSphere Studio:

1. Start the WebSphere Studio workbench and select the **Create a new project option** button.
2. In the New Project dialog, enter a name for the project such as `ShippingExample`. Ensure the default template is set to **<none>**.
3. Select the top-level node in the tree. Select **File->Import** to start the Import Wizard.

4. Type the base URL of the site to be imported. The URL in our environment is:

   `http://august.itso.ral.ibm.com/emp/standard/DynastySiteHome.html`

   Click **Next**.

5. Ensure that the following options are selected:

   Follow folders & sub-folders recursively (requires FTP access)

   Server scripts and other files used to generate the pages being served to the browser (requires FTP access)

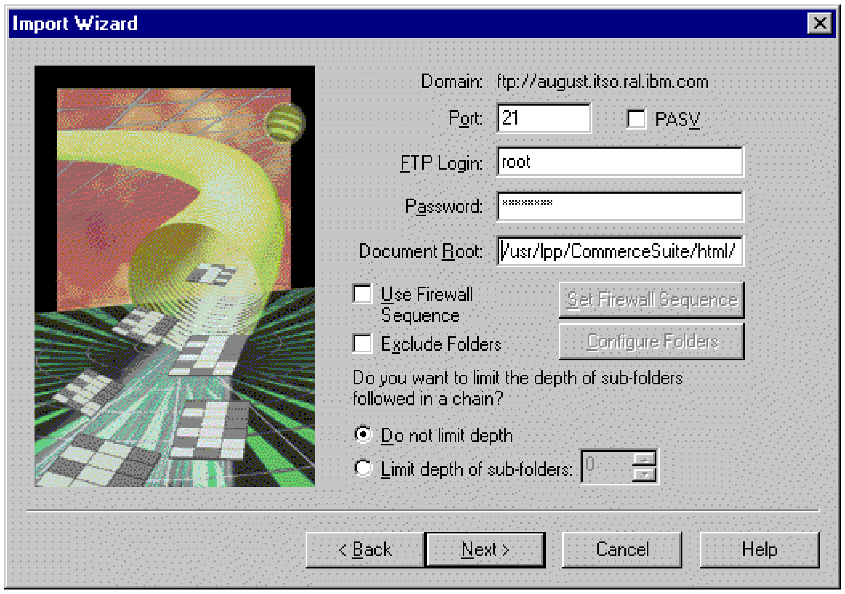
   Click **Next**.

6. Because we're importing files via FTP, enter the FTP port and account information as shown in Figure 52. Also, make other selections appropriate to your environment such as **Use Firewall Sequence**. Click the **Help** button for more information on these options.

   In the Document Root field enter:

   `/usr/lpp/CommerceSuite/html/en_US`

   Ensure that **Do not limit depth** is deselected. Click **Next**.



Domain: ftp://august.itso.ral.ibm.com
Port: 21
[ ] PASV
FTP Login: root
Password: ********
Document Root: /usr/lpp/CommerceSuite/html/
[ ] Use Firewall Sequence
[ ] Exclude Folders
Do you want to limit the depth of sub-folders followed in a chain?
[✓ Do not limit depth]
Limit depth of sub-folders: 0

Figure 52. Import Wizard - specifying FTP and document root options

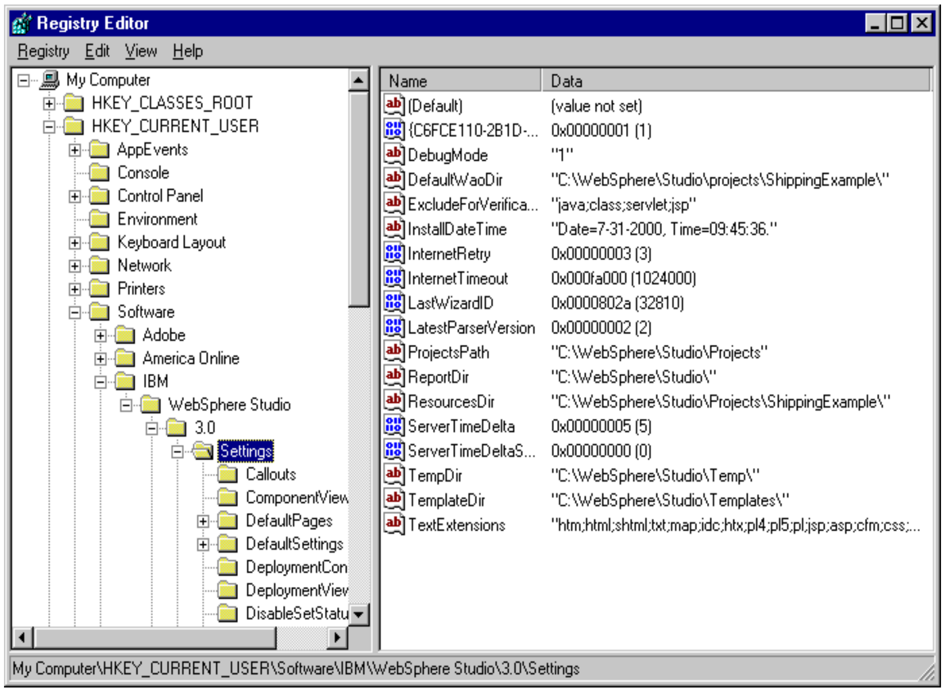
7. In this window, deselect the **Prompt before importing already existing files** option if you have existing files (perhaps from a previous import attempt) and do not want to be notified of this. Click **Next**.
8. In the Summary step of the Wizard, review the options you have selected and click **Finish**. If you want a report to be generated by the import process, select **Generate Import Report**. Click **Finish** to commence the import process.

Special Note

In some circumstances, the WebSphere Studio import may time out before your FTP connection is made to the AIX FTP service. You can remedy this by modifying the time out and retry values used in Studio by editing the Windows NT registry entries:

```
HKEY_CURRENT_USER\Software\IBM\WebSphere Studio\3.0\Settings
"InternetTimeout"=<value>
"InternetRetry"=<value>
```

These entries are DWORD values. The InternetTimeout key is in milliseconds so to set the time out to 32 seconds, specify 32000 (decimal) in this field. This is shown in Figure 53.



Registry Editor

Registry Edit View Help

My Computer\HKEY_CURRENT_USER\Software\IBM\WebSphere Studio\3.0\Settings

Name	Data
(Default)	(value not set)
{C6FCE110-2B1D-...	0x00000001 (1)
DebugMode	"1"
DefaultWaoDir	"C:\WebSphere\Studio\projects\ShippingExample\"
ExcludeForVerifica...	"java;class;servlet;jsp"
InstallDateTime	"Date=7-31-2000, Time=09:45:36."
InternetRetry	0x00000003 (3)
InternetTimeout	0x000fa000 (1024000)
LastWizardID	0x0000802a (32810)
LatestParserVersion	0x00000002 (2)
ProjectsPath	"C:\WebSphere\Studio\Projects"
ReportDir	"C:\WebSphere\Studio\"
ResourcesDir	"C:\WebSphere\Studio\Projects\ShippingExample\"
ServerTimeDelta	0x00000005 (5)
ServerTimeDeltaS...	0x00000000 (0)
TempDir	"C:\WebSphere\Studio\Temp\"
TemplateDir	"C:\WebSphere\Studio\Templates\"
TextExtensions	"htm;html;shtml;txt;map;idc;htx;pl4;pl5;pl;jsp;asp;cfm;css;...

Figure 53. Registry settings to fix time out issues.

When the import is complete, your Studio workbench should resemble the one shown in Figure 54.

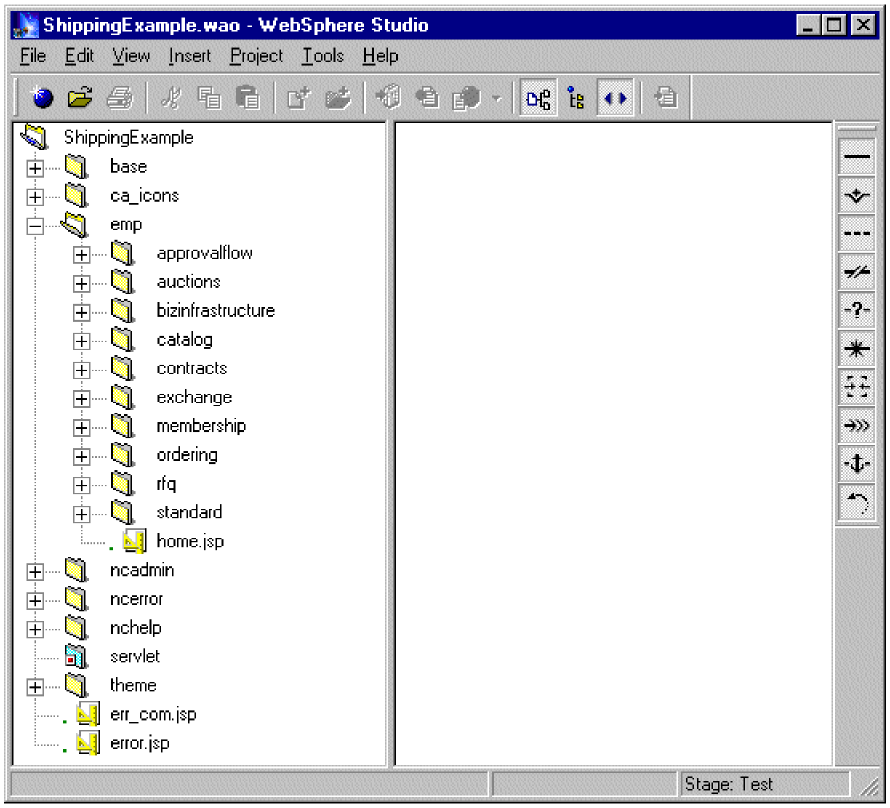


Figure 54. Studio workbench following import of the "Worldwide Shipping Marketplace"

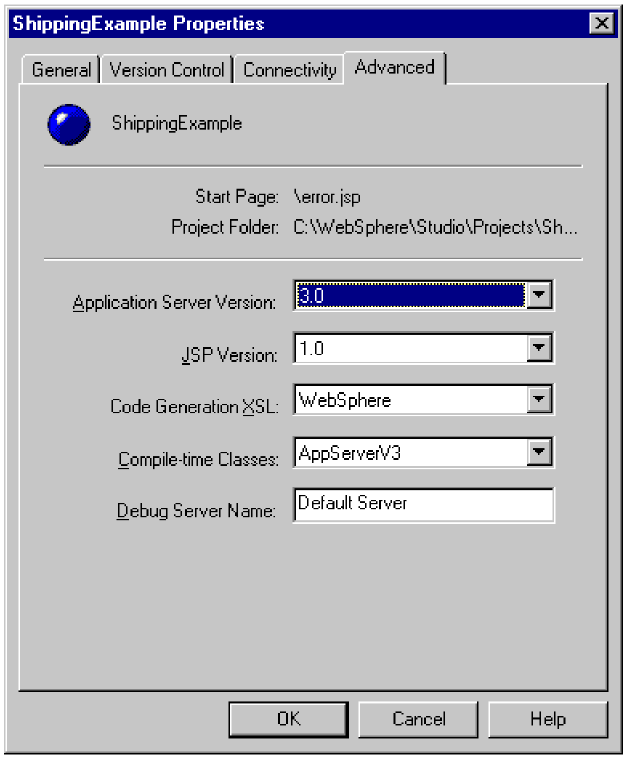
Now that you have the required files imported into WebSphere Studio, you can look at some of them by loading, for example, a JSP file into WebSphere Page Designer. The next section provides details on how to do this.

12.5.2 Making changes

You can modify and inspect some of the files you have imported by using the tools provided in the WebSphere Studio. Because the Marketplace Edition is founded on the WebSphere Application Server Version 3.x, you must ensure that the WebSphere Studio is configured appropriately.

To verify your current Application Server and JSP version support:

1. Select the **Studio project** node (the top-level node in the tree pane).
2. Select **Edit->Properties** to display the dialog shown in *Figure 55.*



ShippingExample

Start Page: \error.jsp
Project Folder: C:\WebSphere\Studio\Projects\Sh...

Application Server Version: 3.0
JSP Version: 1.0
Code Generation XSL: WebSphere
Compile-time Classes: AppServerV3
Debug Server Name: Default Server

Figure 55. Advanced tab of the Project Properties dialog

3. Ensure that the Application Server Version is set to 3.0 and the JSP Version is set to 1.0. Without these settings, you will receive errors when attempting to load JSP pages into WebSphere Page Designer.
4. Click **OK**.

12.5.2.1 HTML and JSP pages

The majority of the imported files are HTML and JSP pages. These can be edited using WebSphere Page Designer, which was installed on your system when you installed WebSphere Studio.

To launch a file into WebSphere Page Designer:

1. Locate the file that you want to view such as category.jsp in the Catalog folder of the tree pane.
2. Double-click the file to open it in WebSphere Page Designer.

In this example, we will add a table cell to the Product Descriptions table. This table already displays properties of the Product bean, namely `getShortDesciption()` and `getLongDescription1()`.

3. Right-click the right most column of the table and select **Specify and Add Cell** from the menu.
4. In the Add Row/Column dialog, select **Column** and **After** and enter 1 in the number text box. Click **OK** to add the column.
5. Straddle the top level heading to span all 3 cells by selecting the top row of the table and selecting **Table->Join Selected Cells**.
6. Switch to the HTML Source view by selecting the tab at the bottom of the main window.
7. In this simple example, we'll simply add the product reference number to the new cell. Click **HTML Source**.
8. Locate the code block shown in Figure 56 and make the modification shown in **bold** to include the product reference number inside the new <TD> tag created as a result of the new cell creation.

```
...
<TABLE BORDER=1 CELLPADDING="4" CELLSPACING="1" BGCOLOR="#E7F6FF">

<TR>
<TH colspan="3" class="TABLEHDR">Product Descriptions
</TH>
</TR>
<% for ( int i=0; i<prodlistSize ; i++ )
{
product= products[i];
String desc = product.getLongDescription1();
if (desc != null ) {
if( desc.length() > 20)
desc = desc.substring(0,20) + "...";
}else{
desc = product.getShortDescription();
}

%>

<TR>
<TD>
<a href="<%= productIC +"prrfnbr=" + product.getProductR
eferenceNumber()  %>"><%= product.getShortDescription() %></a>
</TD>
<TD>
<%= desc + "&nbsp" %>
</TD>
<TD><%= product.getProductReferenceNumber() %></TD>
</TR>
<%
}
%>
<!-- End of for loop -->
</TABLE>
...
```

Figure 56. Making a simple modification to category.jsp.

9. Switch back to **Normal** view. Notice the {J} that now appears in the cell.
10. Save the file. Your page should look like the one shown in Figure 57.

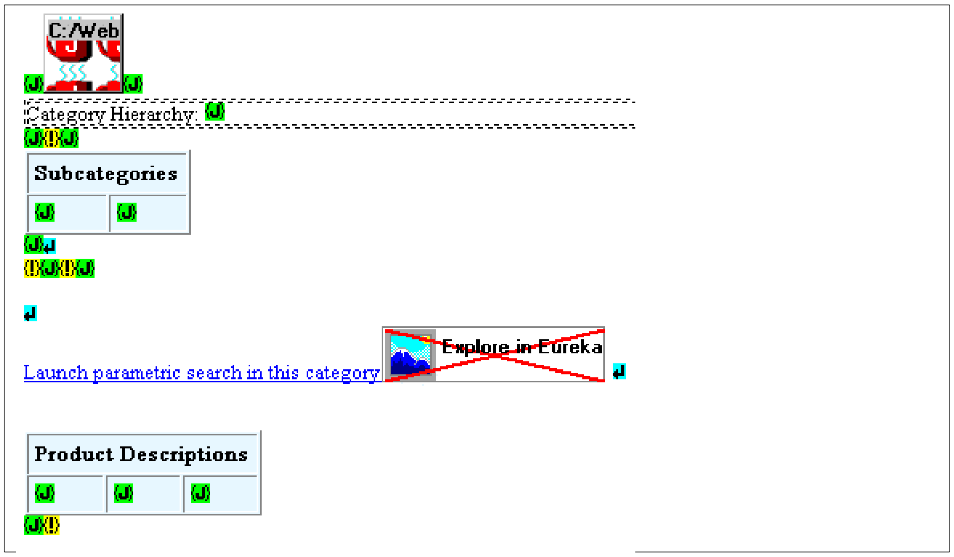


Figure 57. Modified category.jsp

11.Return to the WebSphere Studio workbench.

12.Right-click the edited file (category.jsp) and select **Check-in** from the menu. The file is now ready for publishing.

Java code

The Marketplace Edition ships with .class files so the Java source code is not available for you to view. However, if you develop your own servlets and JavaBeans for use in the e-Marketplace environment, you can include these into your Studio project enabling you to edit and deploy them to the server.

12.5.3 Configuring the publish operation

Having made a change to a file, you will want to publish it back to the site. This is done directly from WebSphere Studio by setting up a *publishing server* and by specifying one or more *publishing targets*.

12.5.3.1 Inserting a new server node

You must specify the server that you want to publish files to and also set the Publishing Stage before you can define publishing targets.

1. Select **View->Publishing**.
2. Select **Project->Publishing Stage->Test**
3. Delete the existing node named localhost.

4. Highlight the **Test** node and right-click to bring up the menu.
5. Select **Insert->Server** to display the Insert Server dialog.
6. Enter the server name such as `august.itso.ral.ibm.com`. Click **OK**.
7. Right-click the new server node to display it's context menu. Click **Properties**. In this dialog, we need to change the publishing type to use FTP and not File system publish.
8. Click the FTP option button and enter the FTP user name and password for your server.
9. Click **Define Publishing Targets** to modify the location that we can to publish the files to. In this case:

   `/usr/lpp/CommerceSuite/html/en_US`

   Click **OK** to close the Publishing Targets dialog.

   Click **OK** to close the server's Properties dialog.

Special Note

We had a problem in our initial publish operation where Studio automatically updated some links in category.jsp to include the full URL to the resource. This caused the resulting page links to be incorrect when the page was displayed. To remedy this situation we manually corrected the URL in Page Designer and republished the file. Also, you may want to deselect the Use Parser option in the file's Properties dialog to prevent Studio from interrogating links in some files known to cause problems in your publishing operations.

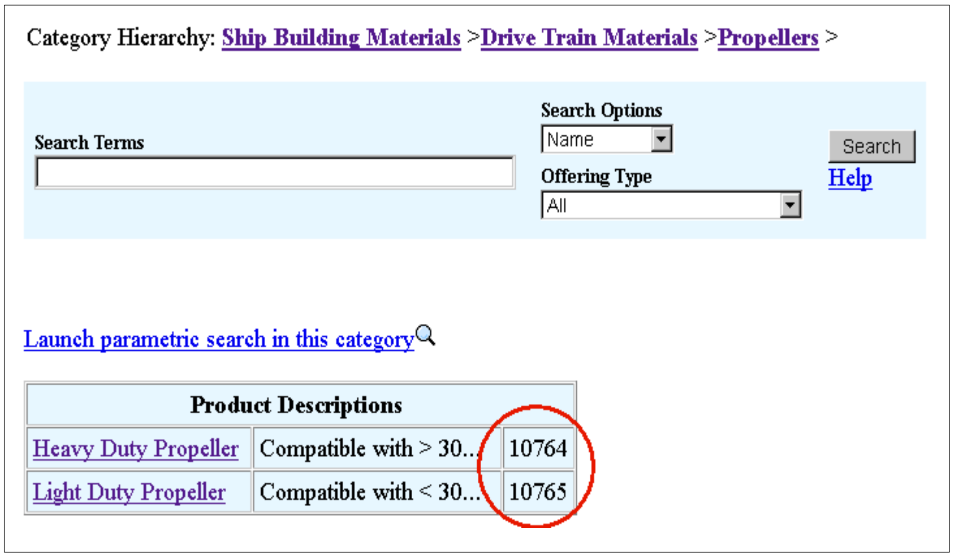
12.5.4 Publishing the modified files

Now that the publishing server is configured, we simply need to select the file we want to publish and publish it.

1. In the Publishing pane, select the **category.jsp** file and right-click to display the file's context menu.
2. Select **Publish this File.**
3. Ensure that ***Relative to document root*** is selected. Click **OK**.
4. Accept any warning about broken child links.Click **OK**.

12.5.5 Verifying the results

You can view the modified page by navigating the product catalog to a subcategory that contains products. You should see the reference number displayed in the right-most cell of the Product Descriptions table as shown in Figure 58.


Category Hierarchy: Ship Building Materials >Drive Train Materials >Propellers >

Product Descriptions		
Heavy Duty Propeller	Compatible with > 30...	10764
Light Duty Propeller	Compatible with < 30...	10765

Figure 58. Modified category.jsp deployed to the example Marketplace Edition site

Chapter 13. Example - membership and access control

The features offered by the Marketplace Edition for membership and access control builds on the concepts of *shopper* and *merchant* provided by WCS 4.1.

In an e-Marketplace, we now have more complex membership and access control facilities. The Marketplace Edition introduces the membership entities of *users*, *organizations* and *markets* and the concept of *user roles*. For each of these entities, we also have *access control* mechanisms that provide security for the various *resources* offered within the e-Marketplace ,such as products, directory entries, commands and interaction controllers. In addition to securing resources, we also have the ability to control the *actions* that members can perform on these resources through the use of *access policies*.

This chapter provides an explanation of these key areas of functionality by explaining the features and design of the membership and access control subsystem. We also provide examples of how to administer each of these features using the Marketplace Edition Administration Console.

13.1 Membership

In the context of e-Marketplaces, membership includes the functions of registration, account maintenance, administration and directory services. The information stored by the membership subsystem is required by virtually all other processes within the e-Marketplace and spans the areas of:

- Contacts and addresses
- Commerce preferences
- Organizational structures
- Roles
- Authentication
- Groups

The e-Marketplace uses this information continually in all market-related functions, such as identifying roles in the buying process, providing notifications to suppliers, and integrating into organizational systems, such as ERP systems or procurement portals.

One of the primary goals of the membership subsystem is to provide a model that enables the appropriate security and authorization procedures to be

© Copyright IBM Corp. 2000

applied to processes, resources and actions. One assurance that the e-Marketplace must give to its participants is the assurance that participants can perform only those tasks they are authorized to.

13.1.1 Participants overview

Non-administrative users in the e-Marketplace can be categorized into the groups of buyer and seller, where each participant is usually part of a larger organizational entity registered with the e-Marketplace. All participants in the e-Marketplace are *users* - the lowest common denominator of the membership subsystem, irrespective of other explicit or implicit user groupings such as grouping by specific user attributes. This is evidenced by the fact that the Marketplace Edition accepts login only by users - not by an *organization.*

An example of e-Marketplace participants is shown in Figure 59. We can observe that this e-Marketplace has three organizations registered in it and each of these organizations has a number of members.

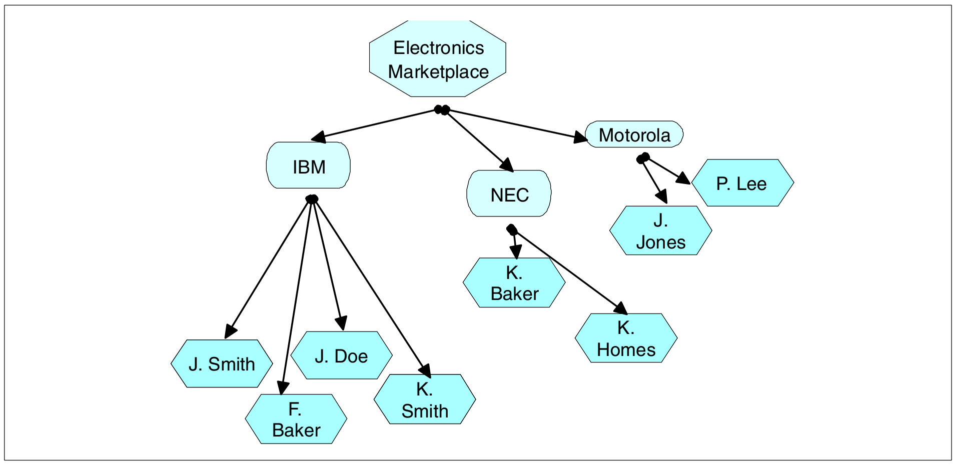


Figure 59. Example e-Marketplace featuring organizations and members

You may be are familiar with the terminology used in WCS 4.1, which categorizes users as *shoppers* or *merchants*. For compatibility, Marketplace Edition provides mappings for *people* to *shoppers* and *organizations/markets* to *merchants*. This mapping is shown in Figure 60.

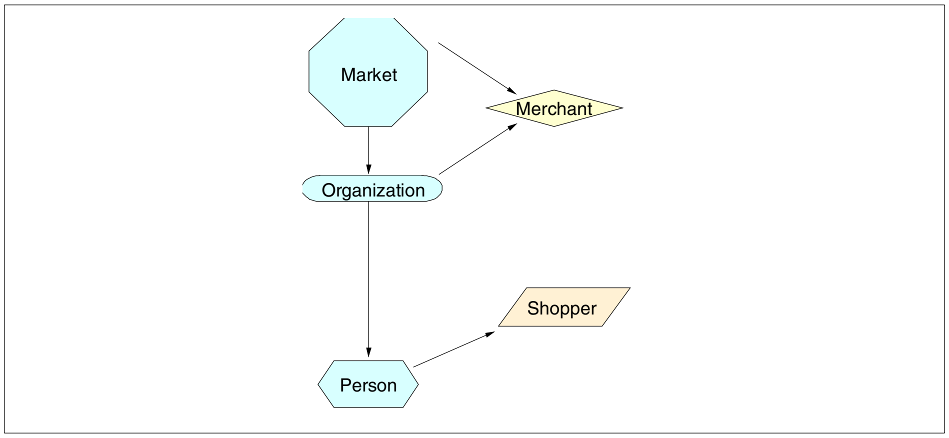


Figure 60. Mapping Marketplace Edition terminology to WCS 4.1

Organizations in the e-Marketplace are comprised of users.

13.1.2 Registration basics

The registration process allows new participants, both users and organizations, to join the e-Marketplace and it is during this process that basic participant information is collected. The actual processes performed for registration can differ depending on the type of functions and services offered by the e-Marketplace. For example, some e-Marketplaces may enforce very stringent registration processes where user-roles are assigned and certain credit checks are performed. Others may offer simple self-registration pages or offer guest access to the e-Marketplace.

The layout of pages associated with the registration process is shown in Figure 61.

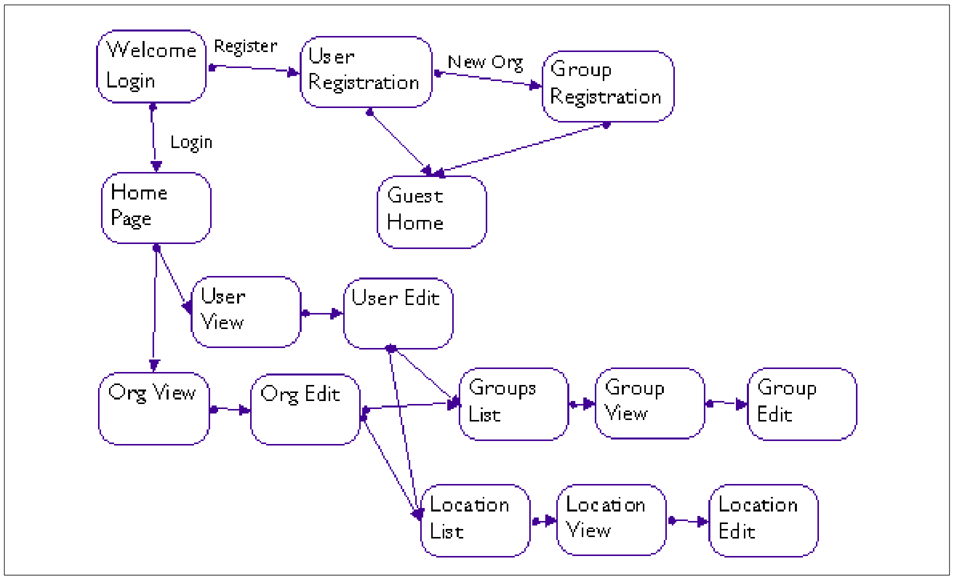


Figure 61. Logical page layout for registration processes

These differing registration requirements highlight three common registration processes:

- Self

  The self-provisioning registration process allows new members to supply the initial account information and to maintain this information themselves. The system is responsible for ensuring that the user ID or organization ID is unique.

- Administrator

  The administrator-provisioning registration process calls on the hub administrator to add new participants to the e-Marketplace and the Organization Administrator to add new users for their organization.

- Guest registration

  Another type of registration that some sites may implement is *guest registration.* This allows users who are not registered with the e-Marketplace to enter and use the e-Marketplace. These users usually have reduced access privileges and are usually required to register with the system prior to making a purchase.

- Registration approval

In any of the above cases, the new registration may require approval before the new participant is admitted to the e-Marketplace. The approval process may involve a human decision or an automated process such as a credit check against a credit subsystem. The Marketplace Edition supports a single-level approval process for registration of new users and organizations.

13.1.3 Pre-registered administrative members

Following installation, the Marketplace Edition has two accounts, *mrkadmin* and *empadmin*. The *mrkadmin* account is a member of the default Market Organization, which is created at installation time and is used to represent the hub business. The *empadmin* account is the general hub administration account.

The default attributes of the two accounts are shown in Table 15. Notice the minor difference in the roles for each.

Table 15. Default hub administration account

Organization Name	Account Name	Password (default)	Roles (default)
Market Organization	mrkadmin	commerce	orgAdministrator marketAdministrator approver
None	empadmin	commerce	orgAdministrator hubAdministrator approver

13.1.4 Registering new users

In this section we describe how to add new users using the Marketplace Edition Administration Console. In this example, the user is added using the *Administrator-provisioning* registration process described in 13.1.2, “Registration basics” on page 243 and we are logged in to the Marketplace Edition with the *empadmin* user ID.

You can navigate to the User Administration pages by starting the Administration Console with a URL such as:

```
http://rs600035.itso.ral.ibm.com/emp/standard/DynastySiteHome.html
```

By selecting the **Administration** link in the Navigation pane, additional links will be displayed, one of them being User Administration. Selecting this link displays the GUI for managing users and allows you to:

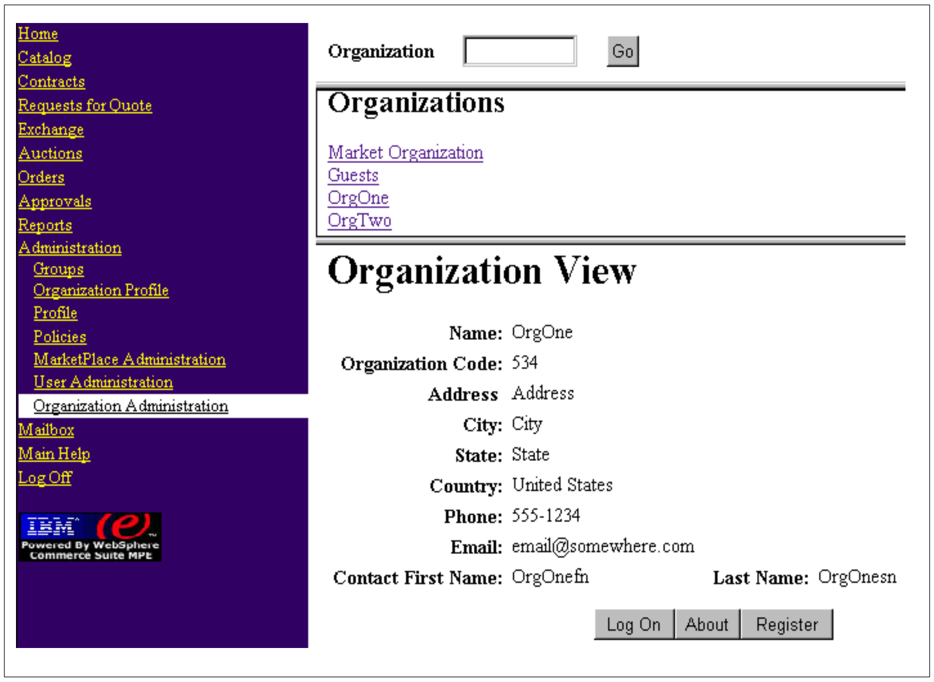
- Display users by organization
- Search for users and organizations
- Display user information
- Edit user information
- Assign locations and groups to the user
- Register new users

Before you add a new user, you must know the organization code of the organization in which the new user will be registered. You should obtain this information by selecting the **Organization Administration** link on the Navigation pane.

To obtain the organization code:

- Click the **Go** button at the top of the page to display all organizations or enter some text in the Organization field to restrict the search operation to some known organizations.
- Click the desired organization from the Organization list.
- The organization code is displayed in the Organization View pane. This is the code that we will require when registering new users.

This Organization View is depicted in Figure 62.



Organizations

Market Organization
Guests
OrgOne
OrgTwo

Organization View

Name:	OrgOne
Organization Code:	534
Address	Address
City:	City
State:	State
Country:	United States
Phone:	555-1234
Email:	email@somewhere.com
Contact First Name:	OrgOnefn
Last Name:	OrgOnesn

Figure 62. Obtaining the organization code

Now that we have the organization code, we can commence adding the new user by clicking the **User Administration** link in the Navigation pane and clicking the **Register** button to display the default registration page.

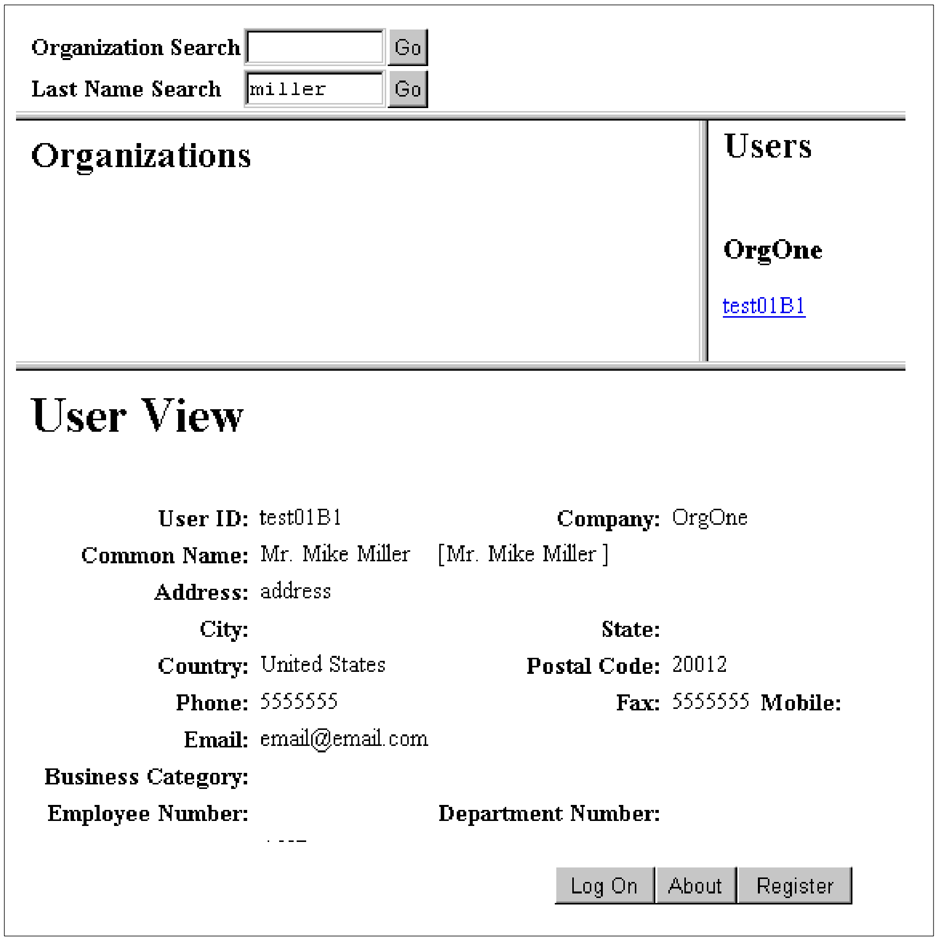
The initial part of the user registration process is straightforward and captures the usual information about the user. Note that some of the fields are mandatory - indicated by bold typeface.

To register a new user:

- Enter a unique *login ID* and default *password* for the user.
- Enter values for the remaining fields.
- In the Organization Membership Code field, enter the organization code you obtained previously.
- Click the **Submit Registration** link to complete the registration.

You should verify that the user was correctly created by selecting the **User Administration** link and typing the surname of the user in the Last Name

Search field. The user should be displayed within the User pane as shown in Figure 63.



Organization Search

Last Name Search: miller

Organizations

Users

OrgOne

test01B1

User View

User ID:	test01B1	**Company:**	OrgOne
Common Name:	Mr. Mike Miller [Mr. Mike Miller]		
Address:	address		
City:		**State:**	
Country:	United States	**Postal Code:**	20012
Phone:	5555555	**Fax:**	5555555 **Mobile:**
Email:	email@email.com		
Business Category:			
Employee Number:		**Department Number:**	

Figure 63. Verifying the new user registration

When the user is registered, you can enter further information about the user from the User Edit page.

Select **Administration->User Administration** from the Navigation pane.

1. Select the organization and user that you wish to edit.
2. From the User View page, click **User Edit**.
3. Enter the additional information for this user.

Note

At this point, the user should be assigned a role based on their intended interactions with the marketplace. This process is discussed in 13.2.10, "Assigning roles" on page 261.

13.1.5 Registering new organizations

The process of registering organizations is similar to the "new user" registration process. In this example, the registration is being performed by the hub administrator.

Select **Administration->Organization Administration or Administration->User Administration** from the Navigation pane.

- Click the **Register** button.
- Locate the link in the text at the top of the page that reads:

  ```
  If your organization is not already registered, click here to request a
  new organization registration.
  ```

This will display the New Organization Registration page where you will register the organization and define an organization administrator for the organization.

- Enter the company name and specify a unique login ID and password.
- Specify the other mandatory and optional information on the form.
- Click the **Submit Registration** button.
- Verify that the new organization was successfully created by selecting **Administration->Organization Administration** and locating the new organization.

This process has defined a new organization within the e-Marketplace and configured an administration account for the organization. This administration account will be used by the organization administrator to add new users for the organization rather than relying on the hub administrator to perform that task.

In the self-provisioning registration process where organizations can register themselves with the e-Marketplace, the hub administrator must approve the new organization before it can become an active participant.

The approvals subsystem is discussed in more detail in Chapter 17, “Example - additional e-Marketplace infrastructure” on page 441.

13.2 Access control

Having defined organizations and users within our e-Marketplace, we can start to consider the access control mechanisms that should be granted for the e-Marketplace members. The primary objective of access control is to govern the actions that users can perform on particular resources. In the Marketplace Edition, this is achieved using access policies that enforce boundaries for the functions that a particular user can perform.

Access control policies in Marketplace Edition control access using a combination of the definitions for:

- Users
- Roles
- Groups
- Actions

This section begins with a discussion of some of these areas before looking at how access policies are implemented.

13.2.1 Users and user groups

Marketplace Edition allows you to group users for access control purposes. Groups always have an owner who can be an individual or an organization. Groups can be explicit or implicit. Explicit groups are created by explicitly adding users to a group. Implicit groups are created by providing group attributes, where users whose profile matches the specified attributes are implicitly made members of the group. This is described in more detail in 13.2.9, “Administering groups” on page 260.

An example of this is to group all those users in a particular *department* (assuming that department was captured in the user profile) because it is likely that the access policies for members of the same department will be identical. By grouping users, access policies can be applied to the entire group reducing the number of policies that need to be created and managed by the administrator.

13.2.2 Resource groups

Resource groups are identical to user groups but provide implicit and explicit grouping of the resources such as command and interaction controllers. The predefined resource groupings in Marketplace Edition are shown in Table 17 on page 256.

13.2.3 Roles

Roles are named groupings of defined actions that can be performed on particular resources. There are a number of basic actions such as view, modify, create, and delete in addition to market-related actions such as quote, buy and requestQuote. Table 16 shows a matrix of actions and roles applicable to the pricing contracts subsystem in Marketplace Edition.

Table 16. Roles and associated privileges for the pricing contracts subsystem

Role	View	Create	Verify	Search	Report	Approve
Buyer	◆			◆		
contractAdmin.	◆	◆	◆	◆		
orgAdministrator	◆			◆	◆	
hubAdministrator	◆			◆	◆	
Approver	◆			◆		◆

User roles

The access control model implemented in the Marketplace Edition has its roots in traditional role-based models but implements a slightly modified terminology. In a pure role-based model, roles represent a function within an organization. Users are assigned roles which inherently grant them the privileges and rights associated with that role as shown in Figure 64, but not necessarily all of the actions that the user can perform.

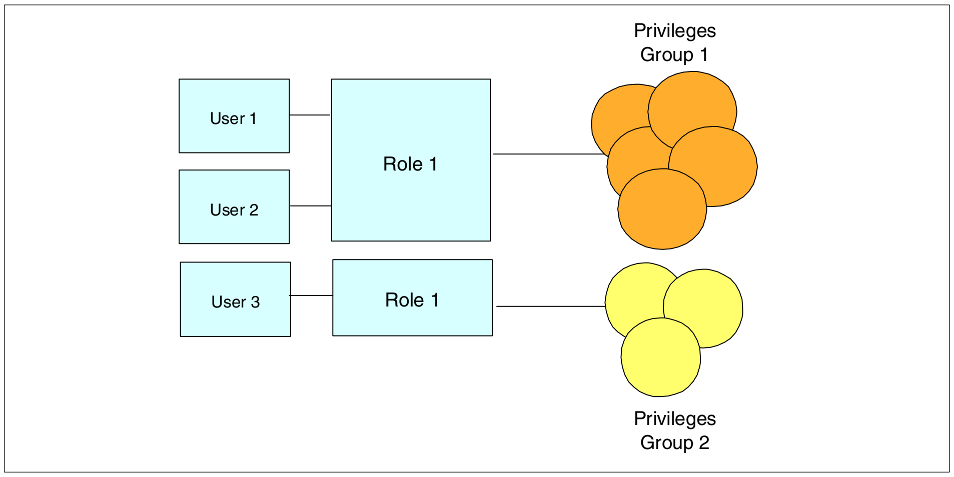


Figure 64. Traditional role-based model for user privileges

However, in e-Marketplaces, it is necessary to further define the term “role” to include the possible actions that a user can perform in the context of a particular resource. Marketplace Edition provides this functionality using the concept of Access Policies. Refer to 13.2.5, “Access policy overview” on page 254 for further information.

Resource roles

Similar to user roles, we can assign roles, or a set of roles, to a resource or group of resources. Each resource may have a user or a group of users that fulfill each role defined for the resource. For example the role of recipient might apply to a document resource, or the role of supplier might exist for a product within the aggregated catalog. Figure 65 shows this relationship.

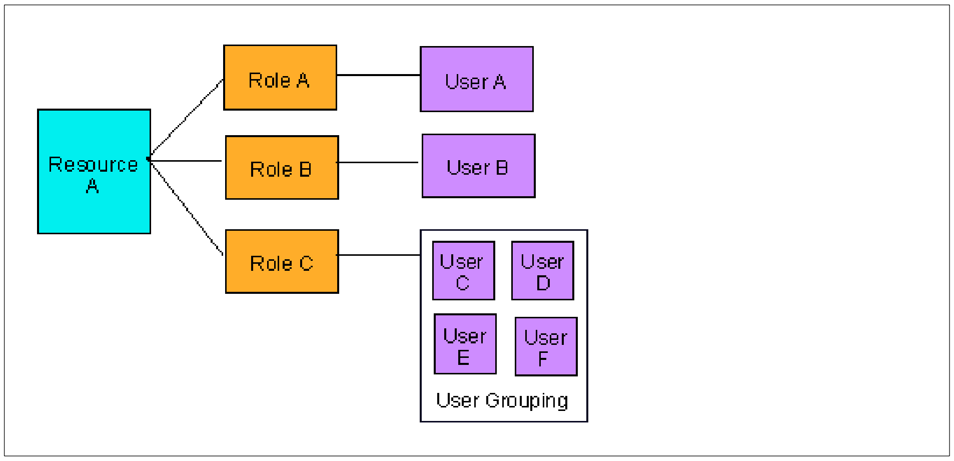


Figure 65. Relationship of resources, roles and users

By assigning roles to resources, we can subsequently govern the actions that can be performed on them.

13.2.4 Actions

Actions represent the actual activities that can be performed on a particular resource in the e-Marketplace. Some actions that may apply to a document resource for example might be “Read” and “Update”. We can also apply actions such as “Buy” or “Send request” to applications in the e-Marketplace which may subsequently kick off additional processes that refer to the target of the action - a product in the case of the “Buy” action. By introducing actions, we effectively control what the user is able to do with a given resource, as shown in Figure 66.

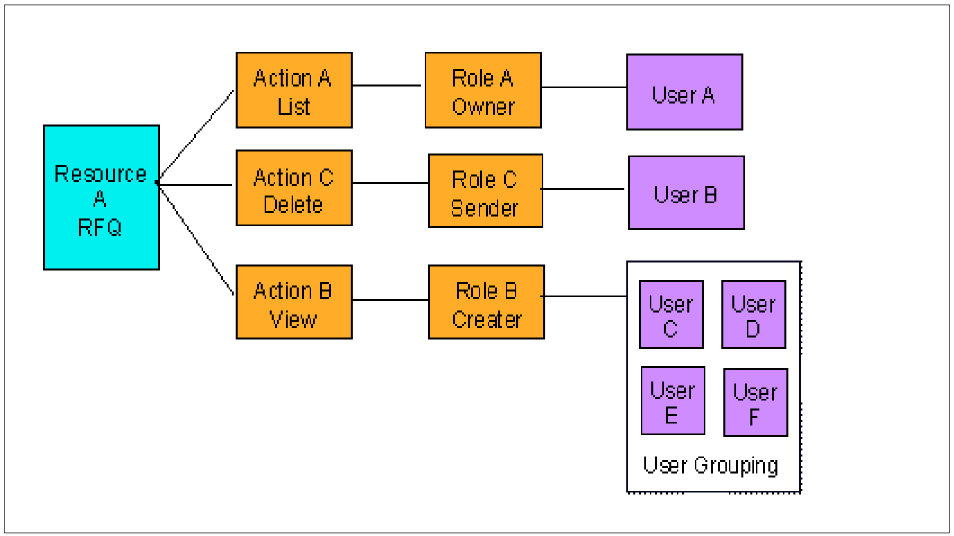


Figure 66. Relationships between resources, actions, roles and users

13.2.5 Access policy overview

Access policies are rules that combine the elements of resources, actions, roles and users together for the purpose of granting user-access to a resource. A textual representation of access policies are:

All users in United States can view all categories

or

User John Smith can buy from category ABC

The access control mechanisms in the Marketplace Edition also support delegation - the granting of authority for a user to act on behalf of another user. This feature is often required, for example, in organizational hierarchies where a manager is required to act on behalf of one of their team members. Alternatively, the manager might hand this responsibility to another subordinate team member. To cater for this situation, the Marketplace Edition provides the "Act For" action to delegate responsibility to another member.

An example of an access policy used during the creation of this redbook is shown in Figure 67. This policy allows all users access to all resources. You might consider using an access policy such as if you are new to Marketplace Edition and simply want to familiarize yourself with the product without access restrictions. In the WebSphere Commerce Suite, Marketplace Edition for AIX, this access policy and all others reside in the directory:

```
/usr/lpp/CommerceSuite/emp_schema/singlebyte/accesscontrol
```

```
<?xml version="1.0" ?>

<Policies>
   <!-- user groups -->
   <UserGroup Name="anyone"  ownerMID="1" >
      <Filter>(objectclass=person)</Filter>
   </UserGroup>

   <!-- resource categories/resource groups -->
   <ResourceCategory Name ="java.lang.Object" ownerMID="1" >
      <Action>*</Action>
   </ResourceCategory>
   <ResourceGroup Name="Object"  ownerMID="1" >
      <Member Name="java.lang.Object" />
   </ResourceGroup>

   <!-- ACPolicy section -->
   <ACPolicy Name="anyoneCanPerformAnyActionOnAnything" ownerMID="1"
UserGroup="anyone" Action="*" ResourceGroup="Object" />

</Policies>
```

Figure 67. freeAccessPolicy.xml

The important elements of this XML script are described below:

- UserGroup

  The UserGroup segment defines the implicit or explicit grouping of our e-Marketplace users to whom this access policy will apply. For example, in the model depicted in Figure 66 on page 254, our user grouping could be based on the common role of creator, to which four users have been assigned.

  In the freeAccessPolicy.xml, the user group specified is “anyone”, which is one of the pre-defined implicit user groups offered by Marketplace Edition. Also note the ownerMID=1 attribute/value pair which specifies the policy owner - in this case the value 1 is the Administration Organization (the e-Marketplace owner). This attribute will appear in all segments throughout the file.

- ResourceCategory

  The ResourceCategory segment defines the resources that will be made accessible by the policy and defines the actions that are valid on each resource. In the freeAccessPolicy.xml file, the ResourceCategory name is java.lang.Object and the actions allowed on this resource are * (all actions). By specifying the root level object of java.lang.Object from which all other commands and interaction controllers are derived, we automatically have access to all Java objects inherited from this object.

- ResourceGroup

  The ResourceGroup segment specifies the objects that you want to allow the UserGroup to perform an action on. In freeAccessPolicy.xml, the group Object has been identified and the Member Name tag set to java.lang.Object.

- ACPolicy

  The final segment, ACPolicy is responsible for naming the access policy and references the other definitions specified in the previous segments in the file. As is appropriate, the freeAccessPolicy.xml is given an appropriate name representing its functionality - “anyoneCanPerformAnyActionOnAnything”.

13.2.6 Access policy resource groupings

There are a number of common predefined resources groupings provided by default. When creating new access policies for an e-Marketplace, you can specify these groupings in addition to any other objects (such as commands or interaction controllers) that you wish to grant access to.

Table 17. Predefined resource groupings

Resource group	**Actions (Commands)**	**Resource relative roles**
User	UserLoginDisplay UserLogin UserRegistrationDisplay UserRegister UserDisplay UserEditDisplay UserUpdate UserPasswordReset UserOrganizationAdministratorSetReset UserHubAdministratorSetReset UserDelete UserAddRole UserRemoveRole UserSearchDisplay UserSearch UserResultsList	owner

Resource group	Actions (Commands)	Resource relative roles
Organization	OrganizationRegistrationDisplay OrganizationRegister OrganizationDisplay OrganizationEditDisplay OrganizationUpdate OrganizationSearchDisplay OrganizationSearch OrganizationResultsDisplay OrganizationReport	owner
User Group	GroupList GroupDisplay GroupEditDisplay GroupCreate GroupAddMembers GroupRemoveMembers GroupAddCondition GroupRemoveConditions GroupDelete	owner member
Location	LocationList LocationDisplay LocationEditDisplay LocationCreate LocationUpdate LocationDelete	owner
AccessPolicy	AccessPolicyList AccessPolicyDisplay AccessPolicyEditDisplay AccessPolicyCreate AccessPolicyUpdate AccessPolicyDelete	owner

13.2.7 Manual access policy walkthrough

In this section, we will walk through the creation of an example access policy.

Figure 68 shows an example of the real-world access policy defaultAcessPolicy.xml, which allows the role of contractAdministrator access to specific contract commands.

1. For this example to work, the role of contractAdministrator must exist in the ldapinit.ldif file located in the directory:

   /usr/lpp/CommerceSuite/emp_schema/singlebyte/membership

```
...
dn: mid=1,dc=emph,dc=ibm,dc=com
mid: 1
objectclass: Member
objectclass: organization
objectclass: top
objectclass: eMerchant
objectclass: eMarketplace
objectclass: Policies
o: Administration Organization
preferredCurrency: USD
empRole: orgAdministrator
empRole: hubAdministrator
empRole: marketAdministrator
empRole: contractAdministrator
empRole: approver
empRole: guest
empRole: buyer
empRole: supplier
aclsource: default
ownersource: default
aclpropagate: TRUE
ownerpropagate: TRUE
entryowner: access-id:CN=ROOT
aclentry: group:CN=ANYBODY:normal:rsc
...
```

Figure 68. The role of contractAdministrator defined in ldapinit.ldif

2. Add a new implicit UserGroup to defaultAccessPolicy.xml:

```
<UserGroup Name="contractAdministrators"  ownerMID="1" >
      <Filter>(empRole=contractAdmin)</Filter>
</UserGroup>
```

3. There should be a ResourceCategory entries in defaultAccessPolicy.xml for all commands that deal with contracts. This tells us the actions that apply to each resource. For example:

```
<ResourceCategory
Name="com.ibm.commerce.emp.contract.icontrollerc.ContractSubmit"
ownerMID="1" >
<Action>Execute</Action>
</ResourceCategory>
```

```
<ResourceCategory
Name="com.ibm.commerce.emp.contract.icontrollerc.ContractVerify"
ownerMID="1" >
<Action>Execute</Action>
</ResourceCategory>
```

4. Create a ResourceGroup of contract commands that includes all the commands that you want only contractAdministrators to execute as follows:

```
<ResourceGroup Name="contractCommands"  ownerMID="1" >
<Member
Name="com.ibm.commerce.emp.contract.icontrollerc.ContractSubmit" />
<Member
Name="com.ibm.commerce.emp.contract.icontrollerc.ContractVerify" />
</ResourceGroup>
```

5. Create the access policy to allow contractAdministrators to execute contractCommands. These entries appear toward the end of the defaultAccessPolicy.xml file in a section denoted by the comment `<!-- START ACPolicy Section -->`:

```
<ACPolicy Name="contractExecute" ownerMID="1"
UserGroup="contractAdministrators" Action="Execute"
ResourceGroup="contractCommands" />
```

13.2.8 Administering access policies graphically

Now that you are familiar with the segments of the XML formatted access policy files, you can better understand the administration process implemented in the Administration Console. Generally, we recommend that you use the console for administration of your access policies.

To administer access policies, select **Administration->Policies** from the Navigation pane.

Creating a new policy

1. Click the **Create Policy** button. You may need to scroll the form to see this button.
2. In the Policy Creation window (no title given at the time of writing), enter the information on which the new policy will be based.
3. Click **Save** when complete.

Edit an existing policy

1. Click the link for the policy you want to edit to display the Policy Edit window.

2. Modify the policy attributes (refer to 13.2.5, “Access policy overview” on page 254 for information regarding the fields).
3. Click *Save* when complete.

13.2.9 Administering groups

Administering groups is performed by selecting **Administration->Groups** from the Navigation pane to display the Groups List page. From here you can view existing groups (and subsequently edit or remove them) and create new groups.

Log into the Marketplace Edition using an Organization Administration account. Select **Administration->Groups** from the Navigation pane.

Creating a new group

1. Click the **Create Group** button.
2. Type a group name such as `AcmeUsersByPostalCode`. In this example, we have three users who all share the same postal code so we’ll create an implicit grouping based on this attribute.
3. Click **Create New Group** to display the Group Edit screen. From here you can explicitly add members or implicitly add members by specifying the common member attributes by which the group will be implicitly created.
4. Scroll to the Attribute list box as shown in Figure 69. Select the **postalCode** attribute and then the **equals** operator. In the value field, type the postal code by which the grouping will be made - in our example, some of our users share the postal code of `27513`. Click the **Add to Attribute List** button.

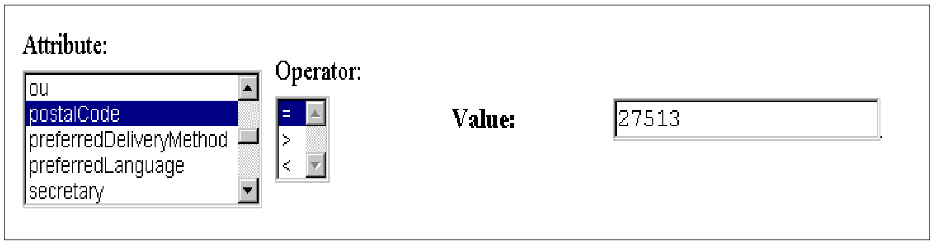
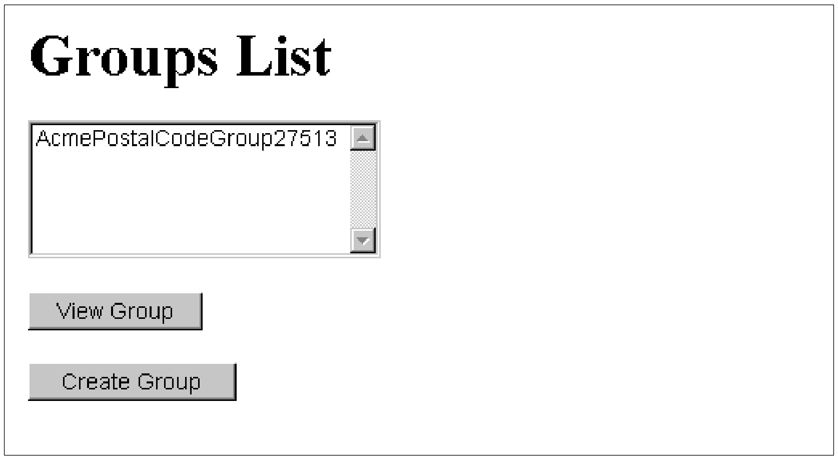

Attribute:
ou
postalCode
preferredDeliveryMethod
preferredLanguage
secretary
Operator:
=
>
<
Value: 27513

Figure 69. Creating an implicit user group by attribute

5. The new attribute will be added to the Attribute List list box.
6. Click **Save Attributes**.

7. Return to the Groups List page by selecting *Administration->Groups* from the Navigation pane. The new group should be displayed in the Groups list as shown in Figure 70.


Groups List

AcmePostalCodeGroup27513

Figure 70. The Groups List

8. To view the group members, highlight the entry and click **View Group**.

Edit an existing group

1. From the Groups List page, select the group and click the **View Group** button.
1. Click **Group Edit** to display the edit page for the group.
2. Make the modifications such as adding and removing members or defining new implicit attributes.
3. Click **Save Attributes**.

Removing a group

1. From the Groups List page, select the group and click the **View Group** button.
2. Click **Group Edit** to display the edit page for the group.
3. Click **Delete Group**.

13.2.10 Assigning roles

The administration console for assigning user roles is reached by selecting the **User Edit** button from within the User View pane. Toward the end of the User Edit screen is the User Roles section where you can assign roles to the selected user. When a user is initially created, they have no pre-assigned roles. It is up to the administrator to assign the roles appropriate for each user. The Marketplace Edition provides the user roles shown in Table 18.

Table 18. Base roles in Marketplace Edition

Role Name	Description
hubAdministrator	Administrator of the entire e-Marketplace.
orgAdministrator	Each organization in the e-Marketplace has an organization administrator who can administrate organization-level tasks.
contractAdministrator	Contract administrators
approver	The approver role allows privileges to approve the actions of other users
guest	Guest roles allows users restricted access to the e-Marketplace.
buyer	Makes the member a buy-side e-Marketplace participant.
supplier	Makes the member a sell-side e-Marketplace participant.

Most non-administrative users will be associated with either the buy-side (shopper) or a sell-side (supplier). The Marketplace Edition gives us two predefined roles to encapsulate base privileges for these types of users. The user privileges associated with these commands can be tracked back to defaultAccessPolicy.xml where there are the appropriate resource groupings for both buyers and suppliers. One such resource grouping is buyerCommands grouping shown, in part, below:

```
<ResourceGroup Name="buyerCommands" ownerMID="1">
<!-- contracts buyer commands -->
<Member
Name="com.ibm.commerce.emp.contract.commands.ContractGetCategoryContractCmd"/>
<!-- rfq buyer commands -->
<Member
Name="com.ibm.commerce.emp.rfq.request.commands.interfaces.CreateRFQCmd"/>
<Member
Name="com.ibm.commerce.emp.rfq.request.commands.interfaces.ModifyRFQCmd"/>
<Member
Name="com.ibm.commerce.emp.rfq.request.commands.interfaces.PublishRFQCmd"/>
<Member
Name="com.ibm.commerce.emp.rfq.request.commands.interfaces.ActivateRFQCmd"/>
...
</ResourceGroup>
```

Figure 71. buyerCommands resource group

This resource group is made available via the access policy name "buyerExecuteBuyerCmds" defined as follows:

```
<ACPolicy Name="buyerExecuteBuyerCmds" ownerMID="1"
UserGroup="buyer"
Action="Execute"
ResourceGroup="buyerCommands" />
```

Figure 72 shows the section of the Edit User screen used to assign roles.

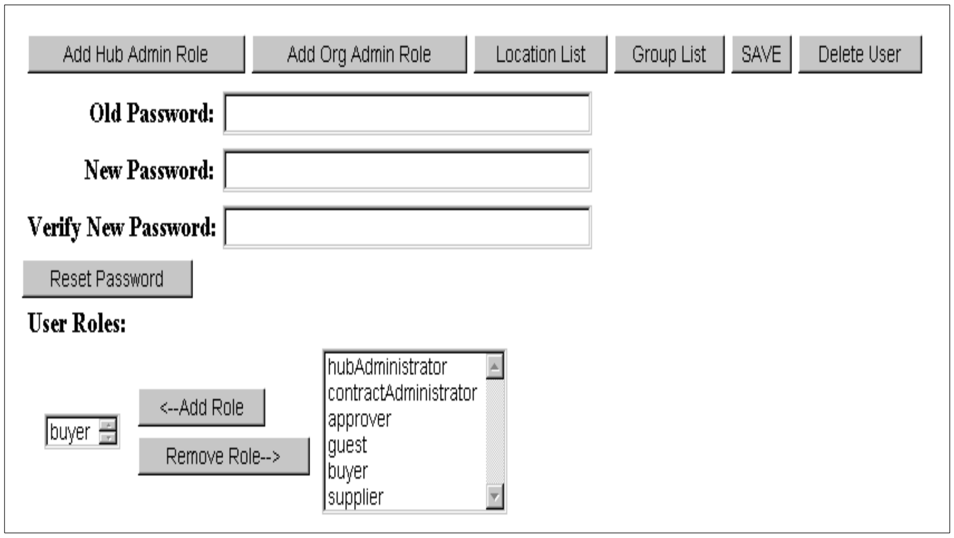


Figure 72. Assigning roles to users

To assign a role to a user:

1. In the User Edit screen, scroll to the bottom of the page.
2. Select the role from the list of available roles, such as **buyer**.
3. Click **Add Role**.

13.3 Customizing the membership and access control subsystem

There are a number of Java commands and interaction controllers centered on the membership and access control subsystem that can be used to customize the behavior of your Marketplace Edition applications. Interaction controllers such as UserSearch, UserAddRole and OrganizationRegister can be called from your own applications so you are not restricted to the default interfaces provided by the Marketplace Edition. This section provides some further details of the elements used in customizing applications around the membership and access control subsystem.

13.3.1 Policy manager

When customizing a Marketplace Edition site, you will often need to determine if a particular user/resource/action combination is permissible in the current context. This is performed via the policy manager which is responsible for managing the access policies held by the site.

The functions of the policy manager are accessed via a Java API. One important method in the policy manager API is the isAllowed() method which has the signature:

```
isAllowed(user, action, resource)
```

This is a simple but powerful method that returns true if the user is allowed to perform the action on the specified resource. The process performed by the policy manager is as follows:

1. To determine access to a resource, the policy manager looks up the resource's owner using its getOwnerDn() method.
2. It retrieves the policies of the owner using the PPPolicyFactory.
3. For each policy returned by the factory, the policy manager invokes the policy.isAllowed() method.

The policy known to the PPPoliciesFactory is loaded from the LDAP server as an XML document and subsequently parsed. Based on the parsed data, the appropriate Java object is created and initialized and added to a hash table for quick retrieval on subsequent calls to the factory.

13.3.2 Interaction controllers

Generally the Marketplace Edition provides interaction controllers for every command in the membership and access control subsystem. In some circumstances, more that one IC will service a single command - where each IC redirects to a different JSP page depending on the access privileges of the user. With this design, the content displayed to the user can be controlled by the access control subsystem.

13.3.3 Commands

Table 19 shows the commands available in the context of the membership and access control subsystem.

Table 19. Commands applicable to the membership and access control subsystem

Membership/Access Control Commands	
UserLoginDisplay	UserDelete
UserLogin	UserAddRole
UserRegistrationDisplay	UserRemoveRole
UserRegister	UserSearchDisplay
UserDisplay	UserSearch

Membership/Access Control Commands	
UserEditDisplay	UserResultsList
UserUpdate	OrganizationRegistrationDisplay
UserPasswordReset	OrganizationRegister
UserOrganizationAdministratorSetReset	OrganizationDisplay
UserHubAdministratorSetReset	OrganizationEditDisplay
OrganizationUpdate	LocationList
OrganizationSearchDisplay	LocationDisplay
OrganizationSearch	LocationEditDisplay
OrganizationResultsDisplay	LocationCreate
OrganizationReport	LocationUpdate
GroupList	LocationDelete
GroupDisplay	AccessPolicyList
GroupEditDisplay	AccessPolicyDisplay
GroupCreate	AccessPolicyEditDisplay
GroupAddMembers	AccessPolicyCreate
GroupRemoveMembers	AccessPolicyUpdate
GroupAddCondition	AccessPolicyDelete
GroupRemoveConditions	
GroupDelete	

13.3.4 Database tables

The database tables related to membership and access control are shown in Table 20:

Table 20. Database tables related to membership and access control

Database Tables	Description
MERCHANT SUBORGREL	Organizations (represented by MERCHANT) can contain other (sub)-organizations. The relationship is expressed by SUBORGREL. **NOTE**: The model does not prevent a sub-organization from being a child of multiple organizations.

Database Tables	Description
ORGATRVAL MEMBERSHIPATR	Organizations can have an arbitrary set of attributes (ORGATRVAL). Attributes may be defined to be inheritable which allows members of an organization to carry the same attributes or override them. The attributes are defined in MEMBERSHIPATR.
SHOPPER ORGMEMBER	Individual members (represented by SHOPPER) belong to organizations (expressed by ORGMEMBER). **NOTE**: The model does not prevent a member from belonging to multiple organizations.
MEMBERATRVAL MEMBERSHIPATR	Members can have an arbitrary set of attribute values (MEMBERATRVAL). These attributes may have been inherited from the organization they belong to or may be specific to the member. The attributes used in MEMBERATRVAL are defined in MEMBERSHIPATR.
ACC_GROUP	The market (and individual organizations) may define roles for members (represented by ACC_GROUP). A role essentially defines the set of actions that a member can perform.
ACC_USRGRP	An organizational member may be associated with one or more roles (ACC_USRGRP).

13.4 Authentication

The WebSphere Commerce Suite, Marketplace Edition for AIX supports the authentication mechanisms supported by WCS 4.1. Registered users who require access to resources other than those allowed using the *guest* account, must be authenticated before they can access restricted resources. The authentication mechanisms implemented depend on the requirements of the e-Marketplace and offer a number of trade-offs in security and ease of use.

The Marketplace Edition provides the usual support for certificates and PKI in addition to mandatory support for LDAP for user authentication.

13.5 LDAP

The membership subsystem is designed around standards-based directory services supporting the LDAP protocol. Because the majority of user information is relatively static and rarely changes, a directory service is very appropriate for persisting this data. It also provides a standard and simple way to retrieve the data using JNDI classes. By default, the directory services in the Marketplace Edition are provided by IBM Secureway Directory.

LDAP Directory entries
In this directory, the attributes of the members and organizations are represented by directory entries, which are made up of a name and a number of attributes. One of the most important attributes in each entry is the object class, which enforces the type of data which can and must be stored at the entry.

Member schema
The directory schema in Marketplace Edition is comprised of object classes and attribute definitions as described above. A number of standard object classes have been defined by standards bodies and include the inetOrgPerson for definition of people and organization and organizationalUnit for definition of organizations. The Marketplace Edition supports these base definitions and, in the case of the inetOrgPerson definition, extends it to the IBM standard - ePerson.

Chapter 14. Example - catalog subsystem

The main objective in any e-Marketplace is to complete a commerce transaction in a timely and accurate fashion. In the WebSphere Commerce Suite, Marketplace Edition for AIX, this objective is fulfilled by the use of the aggregated catalog. The catalog allows the buyers to have a complete view of all the products being traded in the e-Marketplace regardless of the selling organization or the transaction mechanism that supports the offering. The catalog enables the buyers to easily and efficiently initiate the purchasing process. The WebSphere Commerce Suite, Marketplace Edition for AIX catalog also enables the suppliers to inform potential buyers of new products and provide information on existing products in an easy-to-use manner.

In this chapter we will cover the following topics:

1. Catalog subsystem high-level overview
2. Catalog subsystem low-level design
3. Examples of catalog creation and maintenance
4. Supplier interaction: examples of offering creation and maintenance
5. Buyer interaction: examples of catalog based buying
6. Interaction with other subsystems

In this chapter we use several graphics to provide visual representation for some of the topics we are covering. In this redbook we use the "Worldwide Shipbuilding Marketplace" example provided with the default installation of the WebSphere Commerce Suite, Marketplace Edition for AIX.

14.1 Catalog subsystem high level-overview

The catalog subsystem is one of the key components of the WebSphere Commerce Suite, Marketplace Edition for AIX. In this section we provide a high-level overview of the catalog subsystem including some of the unique new features.

Typically any e-Marketplace has three groups that interact with the catalog subsystem: the e-Marketplace administrators, buyers and suppliers. The e-Marketplace administrator creates and maintains the e-Marketplace catalog subsystem. This topic is covered in detail in 14.3.2, "Creation and maintenance of category hierarchy" on page 315. Buyers use the catalog to view the products and offerings and perform transactions. The interaction between buyers and the catalog subsystem is covered in detail in 14.5,

© Copyright IBM Corp. 2000

“Buyer interaction: examples of catalog based buying” on page 333. Suppliers interact with the catalog to create new offerings or manage existing offerings. The interaction between suppliers and the catalog subsystem is covered in detail in 14.4, “Supplier interaction: offering creation and maintenance” on page 327.

The catalog subsystem contains several components that are created by the e-Marketplace administrator to facilitate the task of creating and managing the e-Marketplace catalog. We will start describing how these components work together to facilitate creation of a full catalog.

1. One of the components that an e-Marketplace catalog subsystem requires is a data dictionary. A data dictionary defines all the data types, attribute definitions, units of measurement, and operators that are appropriate for the products being offered in the e-Marketplace. Figure 73 depicts the data dictionary and the logical relationship model.

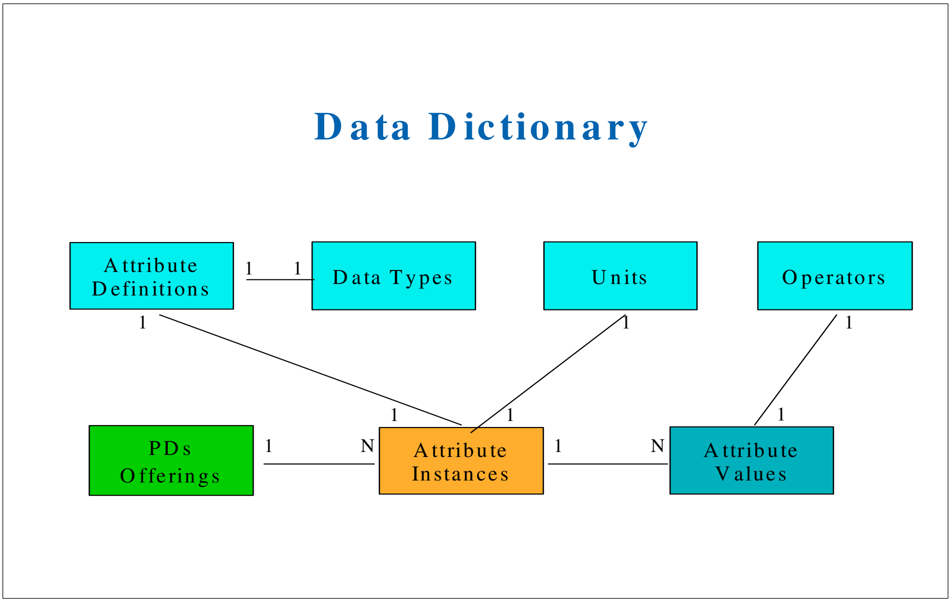


Figure 73. Logical model view of the data dictionary of the catalog subsystem

2. The next component in an e-Marketplace is the catalog hierarchy. The e-Marketplace administrator groups products with similar description together. This grouping can represent a category or a subcategory under a more general category. The e-Marketplace administrator arranges these categories in a category hierarchy that is logically appropriate for the type

of e-Marketplace being created. A catalog hierarchy has three main levels. They are:

- *Root category:* This is the top-most category: all other categories are attached to this. Every catalog must have a root category.
- *Categories:* This is the grouping of all objects in the e-Marketplace that have similar properties. These categories are used to organize the products and the offerings in the e-Marketplace.
- *Product Description Templates:* These are generic descriptions of families of products. Usually they are at the leaf level and they contain attributes and values that are commonly used by the e-Marketplace members to create offering for these products. This method of populating the e-Marketplace catalog imposes a level of consistency between all offerings regardless of the supplier and allows an efficient way of maintaining the e-Marketplace catalog.
- *Offerings:* When an association is made between a trading mechanism and a product description template, an offering has been created. The offerings are placed in the catalog and are made available to buyers.

A product description can be placed at any level. For ease of use, usually all product descriptions are placed at the leaf level of the category hierarchy.

3. The next component of the catalog subsystem is the product taxonomy. The e-Marketplace administrator creates the product taxonomy; by this we mean that product offerings are allowed to be systematically created and arranged in categories according to established criteria. When a supplier creates an offering from a product description, the offering is placed in the proper place in the catalog according to the product taxonomy.

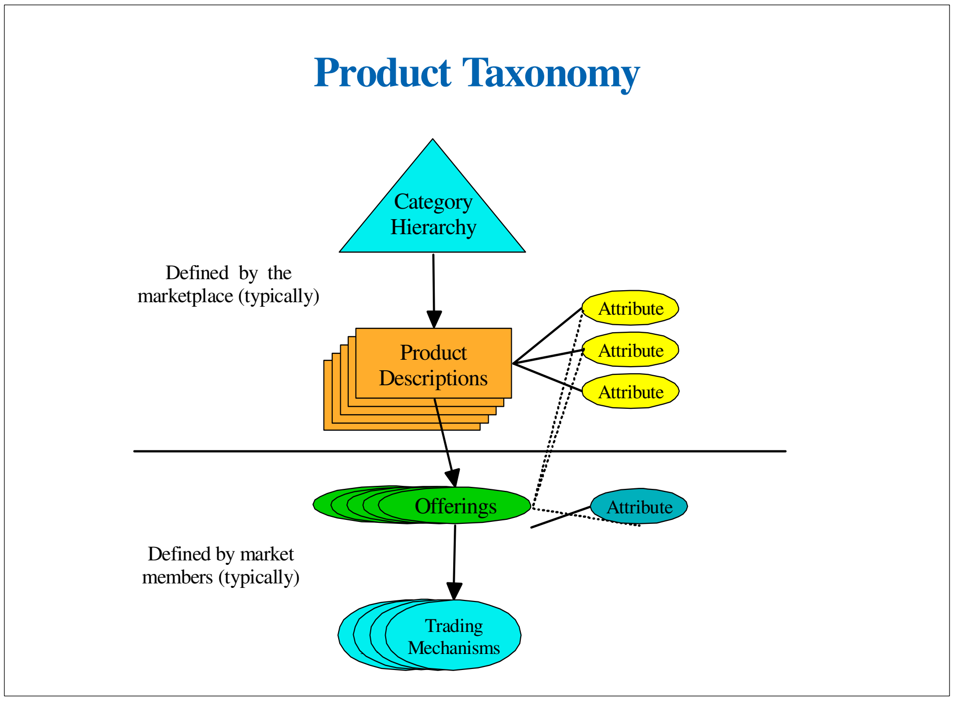


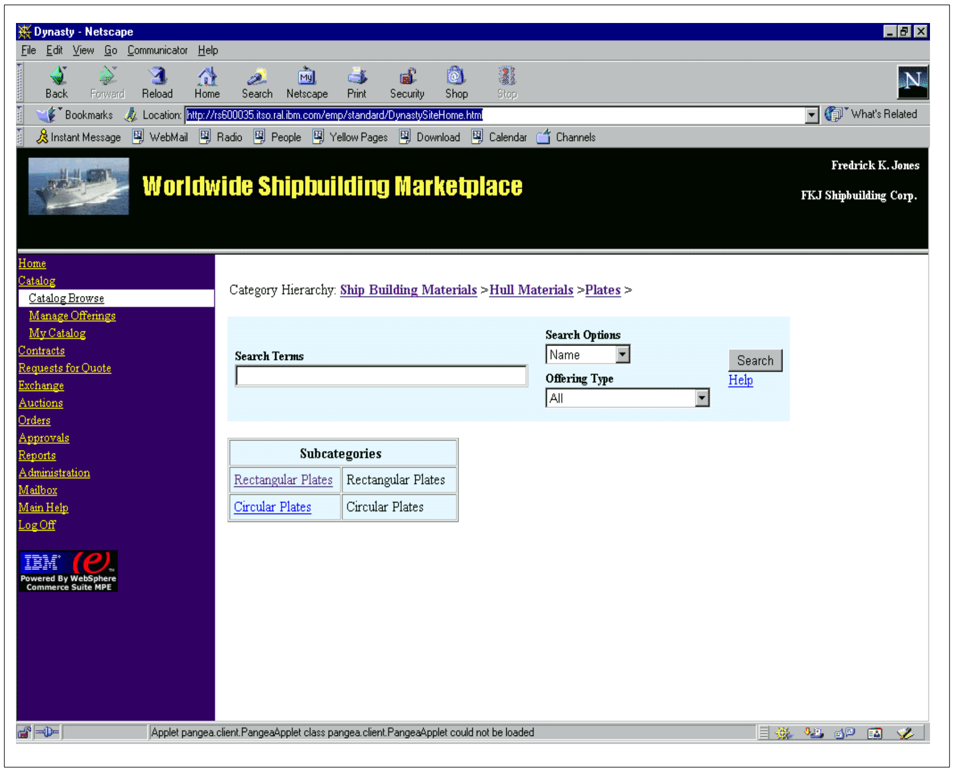
Figure 74. Product taxonomy

The e-Marketplace administrator creates product description templates, which include a set of attributes for a specific product and the product placement in the category hierarchy. Suppliers access the catalog and use the product templates created by the e-Marketplace administrator to create an offering. Suppliers can attach new attributes to the offering from the e-Marketplace data dictionary.

14.1.1 Catalog views

The catalog subsystem presents three views of the catalog:

1. Category browse
2. Manage offerings
3. My catalog


Dynasty - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Shop Stop
Bookmarks Location: http://rs600035.itso.ral.ibm.com/emp/standard/DynastySiteHome.html
What's Related
Instant Message WebMail Radio People Yellow Pages Download Calendar Channels
Worldwide Shipbuilding Marketplace
Fredrick K. Jones
FKJ Shipbuilding Corp.
Home
Catalog
Catalog Browse
Manage Offerings
My Catalog
Contracts
Requests for Quote
Exchange
Auctions
Orders
Approvals
Reports
Administration
Mailbox
Main Help
Log Off
IBM
Powered By WebSphere Commerce Suite MPE
Search Terms
Search Options
Name
Offering Type
All
Search
Help

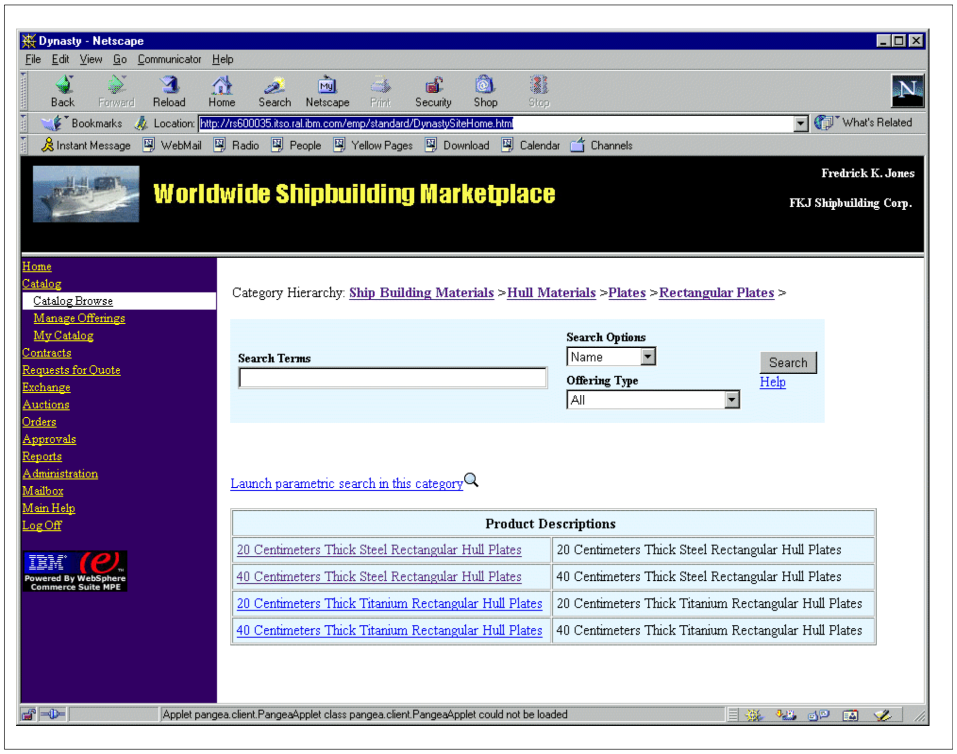
Category Hierarchy: Ship Building Materials >Hull Materials >Plates >

Subcategories	
Rectangular Plates	Rectangular Plates
Circular Plates	Circular Plates

Applet pangea.client.PangeaApplet class pangea.client.PangeaApplet could not be loaded

Figure 75. Catalog browse view

- Catalog browse: This view is typically used by buyers to browse the catalog hierarchy and find the products that they are interested in. Figure 75 shows the subcategories of the catalog browse view. Figure 76 on page 274 shows the product description window of the catalog browse view where buyers can select a specific product by clicking the product description link to see the offering available for that specific product.


Dynasty - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Shop Stop
Bookmarks Location: http://rs600035.itso.ral.ibm.com/emp/standard/DynastySiteHome.html What's Related
Instant Message WebMail Radio People Yellow Pages Download Calendar Channels
Home
Catalog
Catalog Browse
Manage Offerings
My Catalog
Contracts
Requests for Quote
Exchange
Auctions
Orders
Approvals
Reports
Administration
Mailbox
Main Help
Log Off
Search Terms
Search Options
Name
Offering Type
All
Search
Help

Worldwide Shipbuilding Marketplace

Fredrick K. Jones

FKJ Shipbuilding Corp.

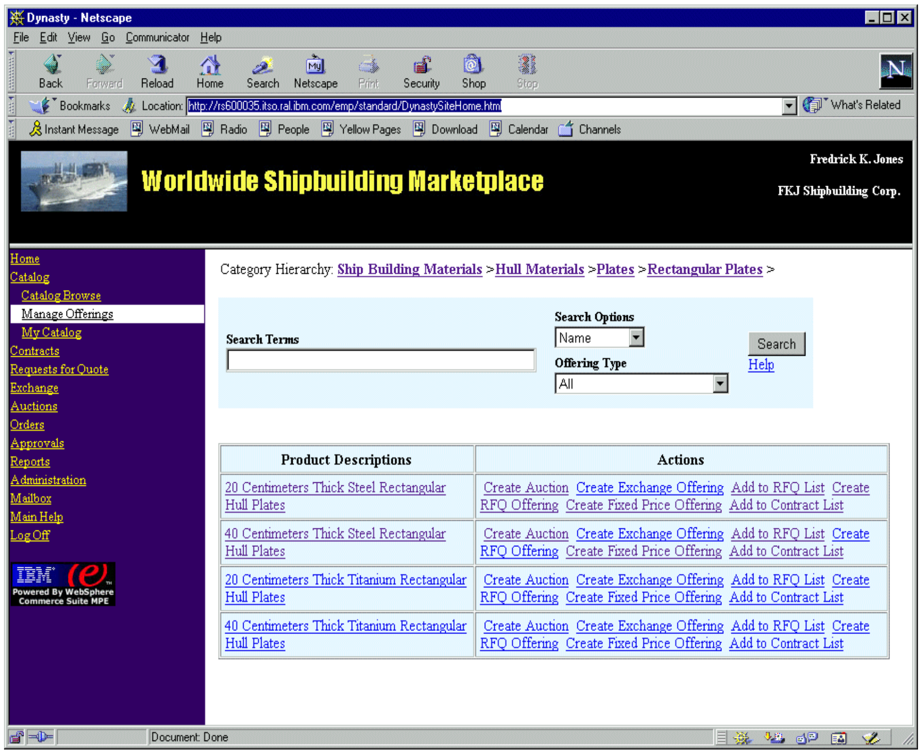
Category Hierarchy: Ship Building Materials > Hull Materials > Plates > Rectangular Plates >

Launch parametric search in this category

Product Descriptions	
20 Centimeters Thick Steel Rectangular Hull Plates	20 Centimeters Thick Steel Rectangular Hull Plates
40 Centimeters Thick Steel Rectangular Hull Plates	40 Centimeters Thick Steel Rectangular Hull Plates
20 Centimeters Thick Titanium Rectangular Hull Plates	20 Centimeters Thick Titanium Rectangular Hull Plates
40 Centimeters Thick Titanium Rectangular Hull Plates	40 Centimeters Thick Titanium Rectangular Hull Plates

Applet pangea.client.PangeaApplet class pangea.client.PangeaApplet could not be loaded

Figure 76. Catalog browse product description view


Dynasty - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Shop Stop
Bookmarks
What's Related
Instant Message WebMail Radio People Yellow Pages Download Calendar Channels
Home
Catalog
Catalog Browse
Manage Offerings
My Catalog
Contracts
Requests for Quote
Exchange
Auctions
Orders
Approvals
Reports
Administration
Mailbox
Main Help
Log Off
Powered By WebSphere Commerce Suite MPE
Search
Help

Location: http://rs600035.itso.ral.ibm.com/emp/standard/DynastySiteHome.html

Worldwide Shipbuilding Marketplace

Fredrick K. Jones
FKJ Shipbuilding Corp.

Category Hierarchy: Ship Building Materials > Hull Materials > Plates > Rectangular Plates >

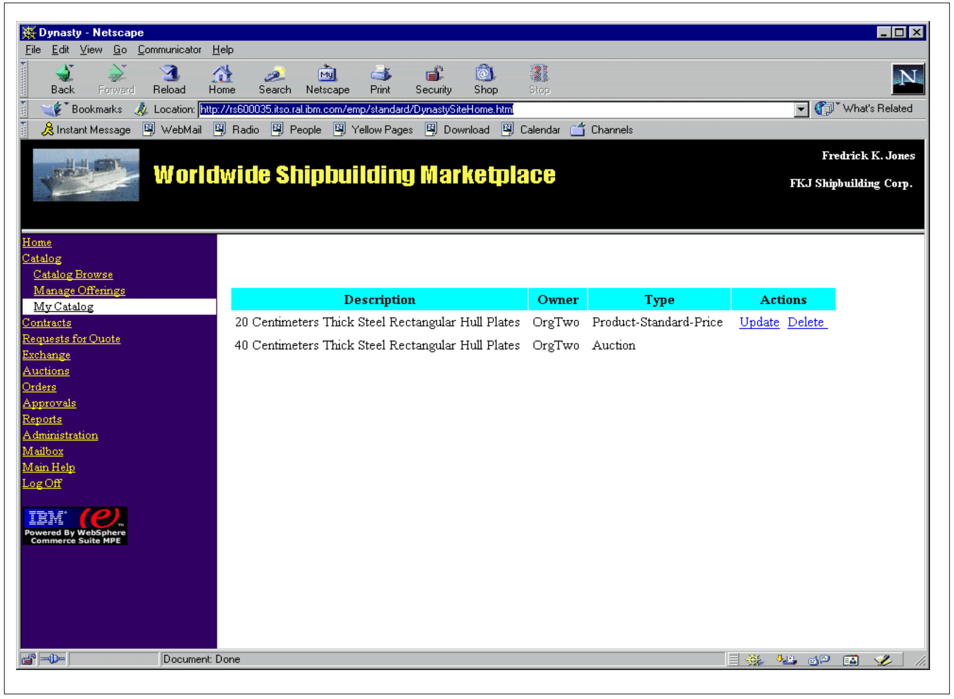
Search Terms:
Search Options: Name
Offering Type: All

Product Descriptions	Actions
20 Centimeters Thick Steel Rectangular Hull Plates	Create Auction Create Exchange Offering Add to RFQ List Create RFQ Offering Create Fixed Price Offering Add to Contract List
40 Centimeters Thick Steel Rectangular Hull Plates	Create Auction Create Exchange Offering Add to RFQ List Create RFQ Offering Create Fixed Price Offering Add to Contract List
20 Centimeters Thick Titanium Rectangular Hull Plates	Create Auction Create Exchange Offering Add to RFQ List Create RFQ Offering Create Fixed Price Offering Add to Contract List
40 Centimeters Thick Titanium Rectangular Hull Plates	Create Auction Create Exchange Offering Add to RFQ List Create RFQ Offering Create Fixed Price Offering Add to Contract List

Document: Done

Figure 77. Manage offerings view

- Manage offerings: This view is used by suppliers to manage their offerings. Suppliers can navigate through the catalog to get to the desired product description window and create an offering for that product. Figure 77 shows the product description level of the manage offerings catalog view and the actions that a supplier can take to create offerings and manage their catalog.


Dynasty - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Shop Stop
Bookmarks Location: http://rs600035.itso.ral.ibm.com/emp/standard/DynastySiteHome.html What's Related
Instant Message WebMail Radio People Yellow Pages Download Calendar Channels
Worldwide Shipbuilding Marketplace
Fredrick K. Jones
FKJ Shipbuilding Corp.
Home
Catalog
Catalog Browse
Manage Offerings
My Catalog
Contracts
Requests for Quote
Exchange
Auctions
Orders
Approvals
Reports
Administration
Mailbox
Main Help
Log Off
Powered By WebSphere Commerce Suite MPE
Document: Done

Description	Owner	Type	Actions
20 Centimeters Thick Steel Rectangular Hull Plates	OrgTwo	Product-Standard-Price	Update Delete
40 Centimeters Thick Steel Rectangular Hull Plates	OrgTwo	Auction	

Figure 78. My catalog view

- My catalog: This view is used by both buyers and suppliers to view the offerings that they have in the e-Marketplace. Figure 78 on page 276 shows the offerings that the currently logged-on user has in the e-Marketplace.

14.1.2 Catalog search

In both Figure 76 on page 274 and Figure 77 on page 275 you can see that the catalog subsystem provides built-in search facilities. The e-Marketplace catalog subsystem provides two search methods:

- *Text search:* Also referred to as a keyword search, is a pre-defined search of certain attributes. Text search is available from both the catalog browse and manage offerings views. Search Term, Search Options and Offering Type are the three fields that accept parameters to perform the search.
  - Search term:
    - To search for a phrase enclose the group of words in double quotes. for example "Drive Train Materials".

    - To search for words that must *all* appear, place a + in front of the words that must appear, for example +Drive +Train +Material.
    - To search for one or more words, simply type the words in the field, for example Drive Train Material
    - To exclude a word in your search result place a - sign in front of the word, for example -Train
  - Search options:
    - To search on a product name, select **Name**.
    - To search on a product description select **Description**.
  - Offering Type: This will limit the search to the selected trading mechanism. The options are:
    - All
    - Exchange
    - Product Standard Price
    - Request For Quote
    - Product Description
    - Auction
- *Attribute-based parametric search:* This type of search mechanism allows searching by using values of the product attributes. This search is available from the product description windows of the catalog browse view where product attributes are available. At the time of writing, this component of the code was not ready and no testing was performed for attribute-based parametric search.

14.2 Catalog subsystem low-level design

In this section we look at the low-level design of the catalog subsystem. This section is useful for developers who will be using the WebSphere Commerce Suite, Marketplace Edition for AIX. In this section we will list the new components of the catalog subsystem and give a brief explanation on their functionality. We do not intend to discuss these components in detail as they are outside the scope of this redbook. For detailed information, we recommend the on line documentation and the JavaDocs that are provided by WebSphere Commerce Suite, Marketplace Edition for AIX.

14.2.1 Design principles

Figure 79 describes the conceptual view of the catalog.

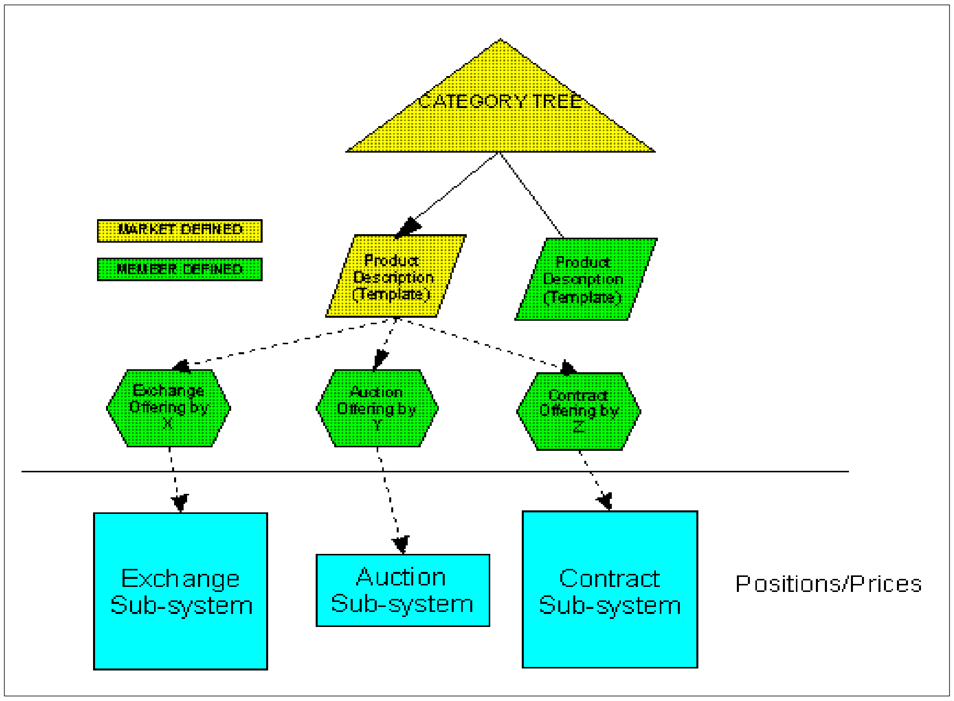


Figure 79. Catalog subsystem Conceptual View

Categories are created and maintained by the e-Marketplace administrator.

Product descriptions define the products in the e-Marketplace by specifying their attributes and their constraints. Offerings are tradeable entities in the catalog that have a trading mechanism and an owner associated with them. Offerings are always entered in the catalog as children of product descriptions and inherit their attribute definitions and constraints. In cases where members do not find a product description and have to create an offering, the attributes of the offering are abstracted into a product description and the offering is made a child of the newly created product description.

The catalog sub-system supports the navigation and searching of the categories, product descriptions, and offerings. Details of subsystem-specific data such as trading positions with respect to a particular offering (for example all bids for an auction offering) are maintained by the subsystem supporting the offering's trading mechanism.

The interaction from the catalog to the subsystems is through subsystem-specific interaction controllers ,which are registered in a subsystem interaction controller registry.

The interaction from the subsystems to the catalog (typically, to add product descriptions or offerings) to the catalog is through commands.

14.2.2 Usage models

The catalog subsystem has three distinct usage models. The first model describes the displaying of the catalog content and navigation through subcategories of the catalog. The second model describes the creation and management of the catalog content and the third model describes the creation of offerings.

14.2.2.1 Displaying catalog content

For any given category that the user is in, the following are displayed:

- Any subcategories in that category
- Any product descriptions in that category

For any given product description that the user selects, the display includes:

- The detailed view of the product description
- A listing of all the offerings that are children of the product description

When using the catalog browse view, the catalog subsystem includes any subcategory or product description listed for the current parent category. At the product description level, the catalog subsystem includes all the offerings available for that product description from all trading subsystems that may have an offering.

When using the manage offering view, the catalog subsystem includes any actions, in the form of interaction controllers, that each trading subsystem may have registered at the category level, such as creating an offering. The same behavior is maintained when a product description is displayed, that is the catalog subsystem includes any trading system actions registered at the product description level. When an offering is listed/displayed, the catalog subsystem includes any actions that each trading subsystem may have registered for the offering level such as viewing the offering positions, creating a bid, and so forth.

When using the My Catalog view, the catalog subsystem includes all offerings that belong to the currently logged on user. Managing interaction controllers at this view allow the user to work with their offerings.

All display of categories, product descriptions, and offerings, regardless of the catalog view option used, are subject to access control restrictions put in place by their respective owners.

14.2.2.2 Creating catalog content

Creation of categories and the category hierarchies is the responsibility of the e-Marketplace administrator. These tasks are performed against the catalog subsystem. Furthermore, the creation of product descriptions and their relationships to the category hierarchy is also through the catalog subsystem. The e-Marketplace Administrator may use one or many of the following interfaces to perform the creation of the catalog content:

- An HTML interface similar to the existing WebSphere Commerce Suite 4.1 administrator interface (ncadmin).
- Through a tool such as Catalog Architect, which generates a defined XML output that is imported into the catalog.
- Through a file interaction controller that allows upload of XML or spreadsheet files with pre-defined schemas into the catalog

14.2.2.3 Creating offerings

Creation of offerings and modifying offering attributes is the domain of the individual trading subsystems based on the type of offering being created. The catalog subsystem is responsible for providing commands, objects, and beans that allow the trading subsystems to manage catalog-specific data for an offering.

An offering can be created either from an existing product description or from scratch.

Create offering from an existing product description

When creating an offering from existing product description, the trading subsystem interaction controller that receives the request instantiates an offering bean and passes it to ProductDescriptionBeanCmd. This command retrieves the product description information into the offering bean and passes the populated bean to the appropriate JSP to displays a form for the user to complete. Through this form, the user enters offering-specific information, new attributes and modifies existing attributes of the product description. The form is then passed to an offering-specific interaction controller, which sets the new parameters into the offering or its attribute beans. It then invokes ManageOffering and ManageOfferingAttributes commands to create the offering. If the trading subsystem creating the offering requires setting member access control for the offering then the

ManageProdMemRel command is invoked to establish the required access control constraints.

Create offering from scratch

When creating an offering from scratch, the trading subsystem interaction controller that receives the request instantiates an offering bean and passes it to the appropriate JSP to display a form for the user to complete. Through this form, the user enters product information, which will be used to create the product description and offering-specific information. The form is then passed to an offering-specific interaction controller to set the new parameters into the offering or its attribute beans. It then invokes the CreateOfferingFromProductDescription command to create the new parent product description and then the offering under this product description. If the trading subsystem creating the new product description needs to add the product description to the category hierarchy of the catalog then the ManageCategoryProductRel command must be invoked. If the trading subsystem creating the offering needs to set member access control for the offering, then the ManageProdMemRel command is invoked to establish the required access control constraints.

Create a compound offering

If the trading subsystem has already added the child offering in the catalog, then the create compound offering will use the CreateOfferingFromProduct, as in the case above, and creates the relationship between the compound offering and the child offerings using the ManageOfferingRel command.

If the subsystem has not yet added the child offering to the catalog, it creates an instance of the CompoundOffering bean, sets its attributes and adds the child offering to the object, using the ManageCompoundOffering command to create the compound offering, the child offerings and the relationship between them. If the trading subsystem creating the new product description needs to add the product description to the category hierarchy of the catalog, then the ManageCategoryProductRel command must be invoked.

14.2.3 Code level components

As mentioned earlier the catalog subsystem is made up of several components. These components are largely built using Java technology. The design pattern for the catalog subsystem follows the Model-View-Controller (MVC) design pattern. In this pattern, users initiate interaction controllers through HTTP requests. Interaction controllers initiate commands that execute business logic and eventually create view pages as a response to the user request. For more information on the Model-View-Controller design pattern, please see *Design Patterns: Elements of Reusable Object-Oriented Software,* by E. Gamma, R. Helm, R. Johnson, J. Vlissides.

In the next few sections we discuss the following catalog subsystem components:

- Interaction controllers
- Commands
- JSPs
- Objects
- Database tables

14.2.3.1 Interaction controllers

The catalog subsystem interaction controllers parse the input parameters and forward requests to the WebSphere Commerce Suite, Marketplace Edition for AIX command manager. The interaction controllers also execute commands that are passed back by the command factory.

The following lists the catalog subsystem interaction controllers for WebSphere Commerce Suite, Marketplace Edition for AIX.

- com.ibm.commerce.emp.catalog.icontrollers.CategoryDisplay

  Initiates the retrieval of a category, its child categories and related product descriptions.
- com.ibm.commerce.emp.catalog.icontrollers.ProductDisplay

  Initiates the retrieval of a product description, its attributes and the child offerings for the product description.
- com.ibm.commerce.emp.catalog.icontrollers.CatalogReport

  Initiates the retrieval of the statistics and report information for a given offering for an organization.
- com.ibm.commerce.emp.catalog.icontrollers.CatalogSearchIC

  Initiates the search commands to searches for products and attributes that match the input search parameter.
- com.ibm.commerce.emp.catalog.icontrollers.CreateSPOffering

  Interacts with standard price offerings commands to manage these offerings. Create, update or delete commands are invoked, based on input parameter for the selected offering.
- com.ibm.commerce.emp.catalog.icontrollers.CreateSPOfferingForm

Alters the standard price offering bean of a given session and creates the fixed price offering form page.

- com.ibm.commerce.emp.catalog.icontrollers.EurikaService

  Initiates the commands for attribute base parametric search.

- com.ibm.commerce.emp.catalog.icontrollers.ManageCatalogIC

  Initiates the retrieval and display of all the offerings that a logged-on member owns.

- com.ibm.commerce.emp.catalog.icontrollers.ManageCategory

  Initiates the retrieval of a category, its child categories and related product descriptions. This interaction controller is used by other subsystems that require catalog management functions.

- com.ibm.commerce.emp.catalog.icontrollers.ManageProduct

  Initiates the retrieval of a product description, the attributes in the product description and the child offerings of the product description. It provides various offering management links.

- com.ibm.commerce.emp.catalog.icontrollers.ManageAccessIC

  Manages the access control entries for a given product.

More information on catalog subsystem interaction controllers is available in the online documentation of WebSphere Commerce Suite, Marketplace Edition for AIX and com.ibm.commerce.emp.catalog.icontrollers JavaDoc.

14.2.3.2 Commands

Commands are the component of the catalog subsystem that carry out the execution of the business logic and ultimately address the requests of the users. Commands interact with almost all other components of the catalog subsystem from interaction controllers to JSPs and beans.

The following are the commands in the catalog subsystem of the WebSphere Commerce Suite, Marketplace Edition for AIX.

- CatalogReportCmd

  Gets an instance of the BCatalogReportList bean that contains a list of BCatalogReport beans. The BCatalogReportList bean stores all the offering for a particular organization and has methods for displaying the content.

- CategoryBeanCmd

  Creates the category bean from the persistent Category object.

- ChildCatalogListBeanCmd

  Retrieves all the immediate categories under a given category.
- CreateOfferingFromProduct

  Creates an offering in the catalog as a child of the identified product description and inherits the attributes of that product description.
- ExtProductAttributeListBeanCmd

  Gets an extended product attribute list bean for a given product description/offering and attribute name.
- GetBaseSpecialPrice

  Retrieves the price of an offering for an organization that has an special price arrangement.
- GetBaseUnitPrice

  Retrieves the base price information for a given offering.
- GetCategoryDisplayTemplate

  Identifies the appropriate template for displaying its sub-categories, and product descriptions.
- GetCompoundOffering

  Retrieves a compound offering, its attributes and any child offerings, related to the compound offering.
- GetParentCompoundOfferings

  Gets a given offering's parent offering in case of a compound offering such as a bundle.
- GetProdMemRelList

  Retrieve the set of member relationships with respect to a product description or offering.
- GetProductDisplayTemplate

  Identifies the appropriate template for a product description and its offerings.
- ManageCategory

  Provides management methods to add, update and delete for category objects.
- ManageCategoryProductRel

  Manages the relationship of a product description with a category by allowing add and delete functionality.

- ManageCategoryRelation

  Manages therelationship between the categories in the catalog.

- ManageCompoundOffering

  Manages a compound offering object by allowing add, modify and delete functionality. A compound offering object holds parental relationships with one or more simple offerings (for example bundle, package, etc).

- ManageOffering

  Manages an offering object by allowing modifies and deletes in the catalog.

- ManageOfferingRelation

  Manages relationships between a parent offering and one or more child offerings.

- ManageProdDescAttributes/ManageOfferingAttributes

  Creates, modifies or deletes offering attributes. Combinations of actions are not allowed per command invocation. The modify operation performs a full overwrite of the selected attributes. All operations are subject to the constraints specified by the attribute definition of the product template if the offering is a child of a product template.

- ManageProdMemRel

  Manages the relationship between a product description or an offering and members.

- ManageProductDescription

  Manages a product description entry in the catalog by allowing add, modify and delete operations on the product description.

- ManageStandardOfferingPrice

  Manages the price of a standard price offering. Performs add, update and delete on the PriceBean objects based on the input operation type.

- OfferingBeanCmd

  Gets the offering bean from the persistent offering object.

- OfferingListBeanCmd

  Gets all offerings, as beans, that match a variety of conditions such as belonging to a product, being of a certain type or belonging to a certain member.

- PriceBeanCmd

Creates an instance of the PriceBean bean from the persistent price object.

- ProductDescriptionBeanCmd

  Creates the product bean from the persistent product object.

- ProductDescriptionListBeanCmd

  Gets all the product descriptions, as beans, that match a variety of conditions, such as belonging to a category or to a certain member.

- SPOfferingBeanCmd

  Retrieves a standard price offering along with its price information.

- StandardPriceApprovableCmd
- CreateStandardPriceOffering

  Creates a standard price offering along with the attributes' price information and access control entries.

- DeleteStandardPriceOffering

  Deletes (marked delete) a standard price offering along with the attributes' price information and access control entries.

- UpdateStandardPriceOffering

  Updates a standard price offering along with the attributes' price information and access control entries.

More information on catalog subsystem commands is available in the online documentation of WebSphere Commerce Suite, Marketplace Edition for AIX and com.ibm.commerce.emp.catalog.commands.interfaces and com.ibm.commerce.emp.catalog.commands JavaDocs.

14.2.3.3 JSPs

JavaServer Pages (JSP) is server-side Java code that facilitates creation and display of dynamic content on Web pages. JSP technology separates the creation of dynamic content from the static HTML content. This feature allows Web page designers to modify the design and the static content of Web pages without having to worry about the dynamic content. All they have to do is just place the dynamic content on the page they are designing. For more information on JSPs and the latest specifications please visit:

`http://java.sun.com/products/jsp/`

The catalog subsystem contains the following JSPs. It is important to note that not all JSPs render a Web page. Some are used to interact with

JavaBeans and extract and pass information to other components of the catalog subsystem.

- AccessControl

  This JSP interacts with the prodmemberSelect JSP to establish product and member relationship.
- CategoryBrowserFrames

  This JSP creates the frames required to perform category navigation and browsing.
- CreateSPOfferingForm

  Displays the form for creation of a standard price offering for a given product.
- CreateSPOfferingFormFrame

  Creates the frames required to display the CreateSPOfferingForm and the associated action buttons for the form.
- GetCategory

  Interacts with BCategory bean to extract child category information. If a category reference number is provided, then the subcategories for that category are extracted. However, if no category reference number is provided ,all child categories for the root category are extracted.
- NewAttribute

  This JSP displays the form to create a new attribute for a product or offering.
- ProductMemRel

  Displays the form that allows user assignment to a specific product when creating a standard offering with restricted access.
- catalog_search

  This JSP displays the catalog search interface available when navigating through the categories of the e-Marketplace catalog. Figure 75 on page 273 shows the catalog search interface at the top of the page.
- category

  When using the catalog browse view ,the category JSP displays the list of product descriptions if there are no more subcategories (see Figure 76 on page 274).
- categoryTOC

When using the catalog browse view, the categoryTOC JSP displays the subcategories in the next lower level (see Figure 75 on page 273).

- managecategory

  This JSP acts similar to category JSP with the difference that managecategory is used when the manage offering view of the catalog is being accessed.

- managecategoryTOC

  This JSP acts similar to categoryTOC JSP with the difference that managecategoryTOC is used when the manage offering view of the catalog is being accessed.

- manageproduct

  This JSP displays the detailed product description when navigating the catalog using the manage offering view. New offerings can be created from this window.

- report

  This JSP creates a summary of statistics of various types of offerings stored in the catalog for a given organization. This JSP is called when the user selects **Reports > Organization** from the navigation frame and then selects **Catalog report** from the list of reports to run.

- reportmember

  This JSP is called when the **Group By Member** button is clicked from the window generated by the report JSP.

- marketReport

  This JSP creates a report of all the offerings by all the members of the e-Marketplace. This report is generated for the users with hub administrator role only.

- mycatalog

  This JSP displays the my catalog view (See Figure 78 on page 276).

- offeringSearchResults

  This JSP displays the result of a search performed based on a specific offering type.

- productDescription

  This JSP displays the detailed product description when navigating the catalog using the catalog browse view.

- productreport

This JSP displays the list of products under a given category, or a message stating that there are no products for the given category.

- refresh

  This JSP is displayed when the register information is updated.

- deleteproduct

  This JSP displays a page indicating that a product has been marked deleted.

- prodmemberSelect

  This JSP interacts with AccessControl JSP to establish a product and member relationship.

For more information on JSPs please see the online documentation.

14.2.3.4 Objects

The WebSphere Commerce Suite, Marketplace Edition for AIX uses two types of objects to accomplish the object-oriented implementation and enable reusability of components. They are:

- Beans
- Persistent Beans

Beans:

Beans are used to instantiate reusable components for commands or are used for interaction with JSPs.

- AccessControlClause

  This bean instantiates an access control clause object and has one method to get the product access clause information.

- BAttributeDefinition

  This bean, with its fields and methods, instantiates and manipulates an attribute definition object.

- BCatalogReport

  This bean instantiates a catalog report object. It provides set and get methods to manipulate offerings and organization information.

- BCatalogReportList

  This bean along with its fields and methods instantiates and manipulates a catalog report list containing catalog report objects for an organization.

- BCategory

This bean instantiates a category object and through its member methods provides set and get functions to the child categories, child products and other subsystem interaction controllers.

- BCategoryProductRel

  This bean instantiates a category product relation object and through its member methods provides set and get functions to interact with the category reference number, product reference number and the sequence number.

- BCategoryRelation

  This bean instantiates a category relation object and through its member methods provides set and get functions to interact with the child and parent category, the category sequencing and the category reference numbers.

- BChildCategoryList

  This bean instantiates a child category lList object. It has only one method to get a list of child categories.

- BCompoundOffering

  This bean instantiate a compound offering object and provides methods to manipulate the object by setting and getting child offerings and their relation type information.

- BExtAttribute

  This bean instantiates an object to extend attributes in the catalog subsystem. Through its member set and get methods, attribute information such as attribute name, attribute options, attribute type, etc. are manipulated.

- BExtAttributeValue

  This bean instantiates an object to extend attribute values in the catalog subsystem. This bean and the BExtAttribute bean work in conjunction with each other.

- BExtProdAttributeValue

  This bean instantiates an object to extend attribute values associated with a specific product description.

- BExtProductAttribute

  This bean instantiates an object to extend attributes for a given product description. This bean provides set and get methods to manipulate the product reference number of the product description to which an attribute is being extended.

- BExtProductAttributeList

  This bean instantiates an object to contain a list of extended product attributes.

- BFormFieldDeterminerBean

  This bean instantiates an object that works with forms and fields in different sections of the application. Through the methods of this bean, field types are determined.

- BMarketReport

  This bean instantiates a market report object, it works with the BCategoryReportList bean to manipulate organizational report at the marketplace level.

- BOffering

  This bean instantiates an offering object, to manipulate offering information such as the offering type etc.

- BOfferingList

  This bean instantiates a list of offerings.

- BOfferingRelation

  This bean instantiates an offering relation object. Through its set and get methods this objects manipulates the offering relation information such as parent reference number, child reference number, relationship type etc.

- BProductAccessControlEntry

  This bean along with BProductAccessControlEntryList instantiate objects to manage the access control entries for products in the catalog subsystem based on the users role, organization they belong to and the type of action.

- BProductAccessControlEntryList

  This bean instantiates a list object to contain all access control entry objects created by the BProductAccessControlEntry bean.

- BProductDescription

  This bean instantiates a product description object. Through its set and get methods this bean provides a large number of functions to manipulate the product description information.

- BProductDescriptionList

  This bean instantiates a list object to contain all product descriptions objects within a given category.

- BStandardPriceOffering

  This bean instantiates an object to facilitate manipulation of standard price offerings.
- BStandardPriceOfferingList

  This bean instantiates a list object to contain all standard price offerings.
- BSubsystemIC

  This bean instantiates an object to facilitate manipulation of subsystem interaction controllers ‘registry entries.
- CatalogConstants

  This bean instantiates an object to initialize the constant parameters of the catalog subsystem. These constant type parameters are used throughout the catalog subsystem.
- ProductFormFieldAdapter

  This bean instantiates an object that controls the product description form fields based on the datatype of the fields.
- SPHelper

  This bean operates as a helper object to the standard price offering object to get form beans for access control and price information.
- SPOfferingConstants

  This bean instantiates an object to initialize constant parameters for the standard price offerings.

For detailed information on these beans please see the online documentation and the JavaDoc on com.ibm.commerce.emp.catalog.beans.

Persistent Beans:
Persistent beans are used to instantiate reusable components that interact with database tables.

- CategoryProductRel

  The persistent object representing an entry in the CGPREL table.
- CategoryRegistery

  A registry object that serves as a cache for category.
- CategoryRelationRegistry

  A registry object that serves as a cache for category relations.
- EMPPrice

The persistent object representing an entry in the PRODPRCS table.

- ExtProdAttributeValue

  The persistent object representing an entry in the EXTPRODATR table.
- ExtProdAttributeValueList

  A persistent list object that deals with a set of ExtProdAttributeValue objects.
- ExtProductAttribute

  The persistent object representing an entry in the EXTPRODATR table.
- Offering

  The persistent object representing an offering entry in the PRODUCT table.
- OfferingRelation

  The persistent object representing an entry in the PRODUCTREL table.
- ProductAccessControlMgr

  The persistent object representing an entry in the PRODMEMREL table.
- ProductDescription

  The persistent object representing product description entry in the PRODUCT table.
- SubsystemIC

  The persistent object representing an entry in the SUBSYSICREG table.
- SubsystemICRegistery

  A registry object that serves as cache for the subsystem interaction controllers.

For detailed information on these beans please see the online documentation and the JavaDoc on com.ibm.commerce.emp.catalog.db**.**

Dictionary Objects:
These objects represent the data dictionary system component of the WebSphere Commerce Suite, Marketplace Edition for AIX.

- AttributeDefinition

  A persistent object representing an entry in the ATTRIBUTEDEF table.
- AttributeDefinitionRegistry

  A registry object that serves as a cache for attribute definitions.

- BOperator

  This bean instantiates an operator object. Through its set and get methods it manipulates the object properties such as operator reference number, operator creator etc.

- BUnit

  This bean instantiates a unit object. Through its set and get methods it manipulates the object properties such as unit reference number, unit scope, unit description etc.

- DataDictionary

  This bean instantiates a DataDictionary object. This object mostly through its get methods extracts operator and unit information, their relationship, and their relationship with attribute definitions in the data dictionary.

- DataDictionaryConstants

  This bean instantiates an object to initialize the constant parameters of the data dictionary.

- GetDataDictionary

  This command retrieves all the data dictionary information.

- ManageAttributeDefinition

  This command manages attribute definitions in the data dictionary.

- OperatorMgr

  The persistent object representing an entry in the OPERATORS table.

- OperatorRegistry

  A registry object that serves as a cache for operators.

- UnitsMgr

  The persistent object representing an entry in the UNITS table.

- UnitsRegistry

  A registry object that serves as a cache for units.

For more information on the data dictionary please refer to the online documentation or the JavaDoc on com.ibm.commerce.emp.dictionary.

14.2.3.5 Database Tables

In this section we are including the modifications and additions to the WebSphere Commerce Suite 4.1 database schema with respect to the catalog subsystem. Some new tables have been added and some existing tables have been modified to facilitate the storage data required for the e-Marketplace catalog subsystem.

Figure 80 depicts the relationship model of the catalog subsystem.

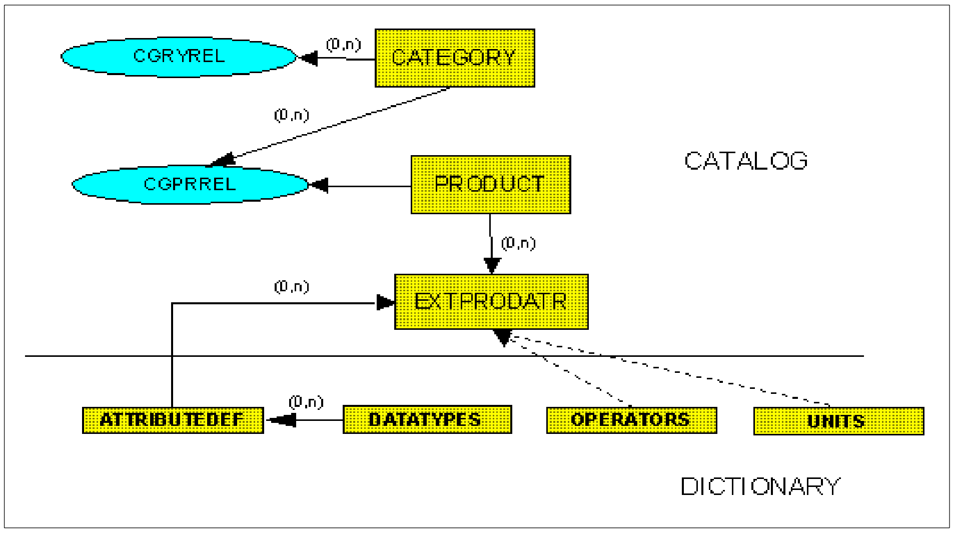


Figure 80. Catalog subsystem database tables ER diagram

Catalog tables

This section lists and describes the tables that are used by the catalog subsystem.

PRODUCT: The PRODUCT table schema has two modifications compared to the WebSphere Commerce Suite 4.1 PRODUCT table. Table 21 shows these changes.

Table 21. PRODUCT table

Column Name	Data Type	Description
PRMENBR	integer	The organization that owns this entry
PRPUB	small int	Should this product be displayed to the public? 0 - No 1 - Yes 2 - Marked for deletion 3 - Targeted (open only to a select set of members) -- set defined in PRODMEMLIST 4 - Draft status (content may not be ready yet)
PRSPECIAL	CHAR(4)	Special information about the product: S - on sale A - Auction offering R - Reverse auction offering (single line item) RB - Reverse auction offering (bundle of line items). Points to a set of offerings with PRSPECIAL=R X - Exchange offering D - Product description -- serves as a template for offerings SP - Standard price -- also governs contracts (if any)

EXTPRODATR (New): The extended product attributes table contains the set of attributes that define products and offerings. The WebSphere Commerce Suite 4.1 product attribute structure is limited. Table 22 presents significant extensions to the product attributes. The WebSphere Commerce Suite, Marketplace Edition for AIX catalog will *NOT* make use of the PRODATR table.

Table 22. EXTPRODATR table

Column Name	Data Type	Description
EAREFNUM	bigint not null	The primary key
EAPRRFNBR	int not null	The product/offering to which this attribute belongs. Foreign key to PRODUCT.PRRFNBR.
EANAME	varchar (128)	Name of the attribute (redundant info -- from ATTRIBUTEDEF).
EAADREFNUM	integer not null	Reference to the attribute definition. Foreign key ATTRIBUTEDEF.ADREFNUM.
EAUSAGE	small int not null	The usage of this attribute: 0: Non-changeable 1: Default value (changeable/negotiable) 2: No value provided (for example certain product template attributes) -- used to associate an attribute with an entity
EAMANDATORY	small int not null	1: mandatory attribute 0: optional attribute (default)
EAVALTYPE	small int not null	Indicates whether the attribute is single-valued, a range, or an enumeration: 1: Single-valued (Default) 2: Range 3: Enumeration
EAUNIT	varchar(32)	The units for the attribute value (for example MHz, lb, GB). Values from UNITS.UNIT.
EASEQNUM	integer	Order in which this field is presented. Unique for a set of attributes belonging to same trading instance (for example RFQ, auction).
EAFIELD1	varchar(254)	Ccustomization field.

EXTATRVALUE (New): Contains values for the attributes associated with products.

Table 23. EXTATRVALUE table

Column Name	Data Type	Description
EVREFNUM	bigint not null	The reference number. Primary key.
EVEAREFNUM	bigint not null	The attribute to which this value belongs. Foreign key to EXTPRODATR.EAREFNUM.
EVOPERATOR	varchar(32) not null	This specifies the relationship of the attribute to the attribute value (for example >, <, =). Values from OPERATORS.OPERATOR.
EVVALUE	varchar(254)	Value of the attribute For FILE/URL data type, this contains the text to displayed that can be clicked to access the file. If null, the file URL is displayed as the text.
EVATTACH	bigint	The file attachment for this attribute (if any). Foreign key reference to ATTACHMENT table.
EVSEQNUM	integer	Sequences the values belonging to a single attribute.
EVFIELD1	varchar(254)	Customization field.

ATTROPTS (New): This table contains options specified for attributes in the ATTRVAL table. Options may include allowed operators, units or even values.

This information is especially useful for specifying templates such as RFQ templates. Table 24 shows the ATTROPTS structure.

Table 24. ATTROPTS table

Column Name	Data Type	Description
AOATREFNUM	bigint not null	Foreign key to ATTRVAL.ATREFNUM.
AOOBJTYPE	char(1) not null	Enumeration to distinguish between Operators, Units, and Values: O: Operators U: Units V: Values
AOVALUE	char (254)	Value corresponding to this particular option. For example, >, <, Kg, lb, 100, etc: Actual value depends on AOOBJTYPE.
AODEFAULT	small int	Indicates if the specified option is default option for that attribute 1: YES, 0: NO

PRODMEMREL (New): This table captures the relationship between products and members. The purpose of this table is to enforce access control for individual products, for example: who is allowed to modify a product or view a product.PRODMEMREL table

Column Name	Data Type	Description
PMPRFNBR	integer (not null)	The product reference number to which this entry belongs. Foreign key to PRODUCT.PRRFNBR.
PMENTRYTYPE	small int (not null)	Indicates whether the entry points to an individual member, a user-defined group, an organization, or to the entire membership (public). This flag serves as a switch on which of the following three columns (if any) is to be used for this entry. At most one of the three columns should be non-null (all three can be null when the entry is public). 1: Member 2: Organization 3: Group
PMMEMBERID	integer	The ID of the member who has a role to play for this product. Refers to SHOPPER.SHRFNBR.

PMORGID	integer	The ID of the organization that has a role to play for this product. Refers to MERCHANT.MERFNBR.
PMGROUPID	integer	The ID of the group that has a role to play for this product. Refers to MEMGROUP.MGREFNUM.
PMROLE	char(4)	The role that this entry plays with respect to this product: A: Approver C: Creator D: Delegatee T: Targeted receiver S: Self-service receiver (found a broadcast RFQ through browse/search and expressed interest) Other flags TBD
PMROLECREATOR	integer	Who created the member's role: could be the same as PMMEMBERID if members added themselves. Refers to SHOPPER.SHRFNBR.
PMADDACTION	char(4)	The action that created this entry. C: Create D: Delegate T: Copy from template Other flags TBD
PMCREATETIME	TIMESTAMP	Time at which the entry was created.

PRODUCTREL (New): This table manages relationships between product descriptions and/or offerings. Relationships are characterized by their type (for example bundles, packages, etc) and their direction (bi-directional vs uni-directional). This concept is from the WebSphere Commerce Suite 4.1. However, the tables defined in the WebSphere Commerce Suite are not used to support these relationships because they are very general purpose and there is large implementation and maintenance overhead associated with using them for an e-Marketplace catalog subsystem implementation.

Table 25. PRODUCTREL table

Column Name	Data Type	Description
PLPARENTRN	integer not null	The parent product/offering of this relationship. The semantics of being a parent are important only in uni-directional relationships (Foreign key to PRODUCT.PRRFNBR).
PLCHILDRN	integer not null	The child product/offering of this relationship. The semantics of being a child are important only in uni-directional relationships (Foreign key to PRODUCT.PRRFNBR).
PLRELTYPE	integer not null	The relationship type: 0: Bundle 1: Package Others as required
PLDIRECTION	integer not null	0: Uni-directional relationship (DEFAULT) 1: Bi-directional relationship
PLMEMRN	integer not null	The member who created/owns this relationship.
PLFIELD1	varchar(128)	Customization field.
PLFIELD2	varchar(128)	Customization field.

SUBSYSICREG (New): This table acts as a registry for the interaction controllers that are provided by each subsystem that needs to be integrated into the catalog. The catalog subsystem uses entries in the registry when displaying product descriptions and offerings to users so that the correct subsystem is invoked.

Table 26. SUBSYSICREG table

Column Name	Data Type	Description
SRREFNUM	int not null	The primary key.
SRCLIENTSYS	char (4) not null	The subsystem that can be the client for this entry: P: Product catalog B: Business subsystem S: Shop-cart
SRSYSTEM	char (4) not null	The subsystem to which this IC applies: A: Auctions R: RFQ C: Contract X: Exchange

SRSCOPE	char(4) not null	Where to use this IC: C: Category P: Product description O: Offering When the IC is applicable at multiple levels, there will be multiple entries for that IC (one for each level). This makes querying for the applicable list easier.
SRURL	varchar(254) not null	The URL of the IC
SRHTTPS	integer not null	Indicates whether the URL to subsystem IC should use HTTP or HTTPS: 0: http (default value) 1: https
SRURLPARAM	varchar(254)	Parameters and values that the subsystems may want associated with the IC URL. This will be of the form param1=value1&param2=value2&param3=value3
SRDESC	varchar(254)	A description of the IC.
SRTITLE	varchar(32)	A string to display along with the URL.
SRIMAGE	varchar(254)	An image file URL that maybe used to display the URL.
SRSEQNUM	small int	The display sequence number
SRFIELD1	integer	Customization field.
SRFIELD2	varchar (254)	Customization field.

Interest list tables:
The interest list tables are used to implement the different types of interest lists (or carts) that will be available for users of the WebSphere Commerce Suite, Marketplace Edition for AIX. Two existing WebSphere Commerce Suite 4.1 tables (IILIST and SHOPPINGS) are used as is.

ILISTTYPE (New): This table adds type information to interest lists and is complementary to IILIST. Also, one of the merchant extension fields in

SHOPPINGS will be used for the WebSphere Commerce Suite, Marketplace Edition for AIX purposes.

Table 27. ILISTTYPE table

Column Name	Data Type	Description
ITLISTRN	int not null	Foreign key to IILIST.LIST_RN
ITTYPE	char (1) not null	The type of the list (the subsystem that will eventually be responsible for the contents): C: Contract R: RFQ S: Standard Price
ITSTATUS	char (1) not null	The status of the list: A: Active D: Marked for deletion S: Submitted?
ITEXTRN	varchar(128)	An extended reference number that a subsystem might want to associate with the list.
ITFIELD1	varchar(128)	Customization field.

Data Dictionary tables:
This section lists and describes the tables that are used to create and maintain the data dictionary of the catalog subsystem.

ATTRIBUTEDEF (New): This table defines the set of attributes used in the e-Marketplace. It serves as a dictionary for the e-Marketplace and organizations in it. Having a well-defined attribute dictionary allows the e-Marketplace to offer a wide variety of attribute-based services such as search, access control, and decision support. All objects that have attribute-value pairs will use the dictionary. The only exception is the product attributes because there is a lot of existing functionality in WebSphere Commerce Suite that depends on its current structure.

Table 28. ATTRIBUTEDEF table

Column Name	Data Type	Description
ADREFNUM	Integer (not null)	The ID assigned to this particular attribute. Primary key.
ADORGID	integer	The organization that defined the attribute. Foreign key reference to MERCHANT.MERFNBR.
ADCREATOR	integer	The user who defined the attribute. Foreign key reference to SHOPPER.SHRFNBR.
ADNAME	varchar (128)	Name of the attribute.
ADDATATYPE	integer	The datatype ID of the attribute (data type definition in DATATYPES table).
ADDEFAULTUNIT	integer	The default units for this attribute definition. Reference to UNITS.UNREFNUM.
ADVISIBILITY	small int	Whether the attribute is visible only within the organization that defined it or across the market: 0: Org wants it to be private 1: Org wants it to be public to the market (pending approval to be a market public attribute) 2: market has approved it to be public
ADPUBLISH	integer not null	Indicates whether the attribute definition is ready to be used. This will also be used to determine whether the attribute definition can be modified. 0: Not yet published 1: Published
ADFIELD1	varchar(32)	Customization field.

UNITS (New): This table contains the domain of units for the e-Marketplace.

Table 29. UNITS table

Column Name	Data Type	Description
UNREFNUM	integer not null	Reference number, primary key.
UNIT	varchar(32)	This field can have such values as "MHz", "lb.", "oz.",....
UNDESC	varchar(254)	The display/description string for the units (subject to translation).
UNCREATOR	integer	Member who created the units. Default is market admin ID.
UNSCOPE	integer	Visibility of the unit -- a value from merchant. When market ID is used the type is visible to the market. Default value.
UNFIELD1	varchar(32)	Customization field.

OPERATORS (New): This table contains the domain of operators for the e-Marketplace. **Note**: Not all elements of the schema may be currently used in the WebSphere Commerce Suite, Marketplace Edition for AIX.

Table 30. OPERATORS table

Column Name	Data Type	Description
OPREFNUM	integer not null	Reference number, primary key.
OPERATOR	varchar(32) not null	This field can have such values as “<”, “<=”, “>”, “>=”,
OPERATOR2	varchar(32)	This is similar to the OPERATOR field, for compound operators (for example RANGE).
OPDESC	varchar(254)	The display string for the operator (subject to translation). For example: Between (for a range); Set; Equals, Greater Than; (UNIQUE)
OPTYPE	integer not null	The operator type: 1: Simple operator (allows a single value) 2: Compound operator (Range -- continuous) 3: Compound operator (Set) 4: Compound operator (Range w/ increment) All the compound operators allow multiple values. Additional types can be defined in the future.
OPAPPLYTO	integer not null	1: Applicable only to numeric data types 2: Applicable only to non-numeric data types 3: Applicable to both numeric and non-numeric
OPCREATOR	integer	Member who created the operator. Default market admin ID.
OPSCOPE	integer	Visibility of the type -- a value from merchant. When market ID is used the type is visible to the market. Default value.
OPFIELD1	varchar(32)	Customization field.

ATTACHMENT (New): This table contains control information regarding files that have been provided as attachments (currently only for RFQs and their responses/bids).

Table 31. ATTACHMENT Table

Column Name	Data Type	Description
ACREFNUM	bigint (not null)	File reference number, primary key
ACPATH	char(254) (not null)	File path (bidrfn/userid/[bid/product]/product reference number/
ACIPADDR	char(16)	IP address from where file was uploaded
ACTYPE	char(32)	File type
ACNAME	char(64)	Name of the file (without the path)
ACSIZE	integer	File size
ACMEMBER	integer (not null)	Member who uploaded the file Foreign key reference to SHOPPER.SHRFNBR
ACVRSCMT	char(254)	Virus scan comments
ACUPDATE	timestamp	Date of file update
ACCRDATE	timestamp	Date of file upload
ACICON	char(254)	GIF image for the file type
ACFIELD1	integer	Merchant customization field
ACFIELD2	varchar(254)	Merchant customization field
ACFIELD3	integer	Mmerchant customization field

14.3 Examples of catalog creation and maintenance

In this section we cover some examples of catalog subsystem-related creation and maintenance tasks that can be performed by the e-Marketplace administrator. These tasks are some of the very early activities an e-Marketplace administrator should engage in to establish the e-Marketplace catalog.

There are several methods to create and maintain an e-Marketplace catalog, such as using XML imports, Excel Spreadsheet, the e-Marketplace version of Catalog Architect ,or using the Web-based administrative tool also known as ncadmin. In this section we concentrate on the administration through the

Web interface of the ncadmin. The basic concept of working with the Web-based administrative tool of WebSphere Commerce Suite, Marketplace Edition for AIX is similar to that of WebSphere Commerce Suite 4.1, with the notable difference of exclusion of store administrative functions, since the concept of stores does not apply to an e-Marketplace.

We will cover creation and maintains tasks for: data dictionary, catalog hierarchy, and catalog content.

***** Note *****

All the operations in the Web-based administrative tool are performed with the “ncadmin” logon ID

For the remainder of this section we will use the default example, “Worldwide Shipbuilding Marketplace” shipped with the WebSphere Commerce Suite, Marketplace Edition for AIX, to demonstrate the creation and maintenance operations.

We will be adding a new subcategory to the main Ship Building Material called Communication Equipment. Then we will set up two further subcategories under Communication Equipment called Radios and EPIRB”(Emergency Positioning Indicating Radio Beacon). Finally we will list some product descriptions for each of these subcategories. Since we are creating a communications section for our e-Marketplace, we need to add new attributes to the data dictionary. We will add Frequency and Coverage stating the frequency range in MHz and distance covered in Km.

14.3.1 Creation and maintenance of the data dictionary

We mentioned earlier in this chapter the concept of the WebSphere Commerce Suite, Marketplace Edition for AIX data dictionary. In this section we explain how you would create new attributes and manage existing attributes in the data dictionary using the Web-based administrative tool. It is important to understand that you can only manage attributes that are not yet published to the e-Marketplace and therefore are still in the Draft state. This is simply because if an attribute is published to the e-Marketplace then there is a chance that a product description or an offering is using that attribute, so changing the attribute will create transactional errors.

14.3.1.1 Creating a new attribute definition

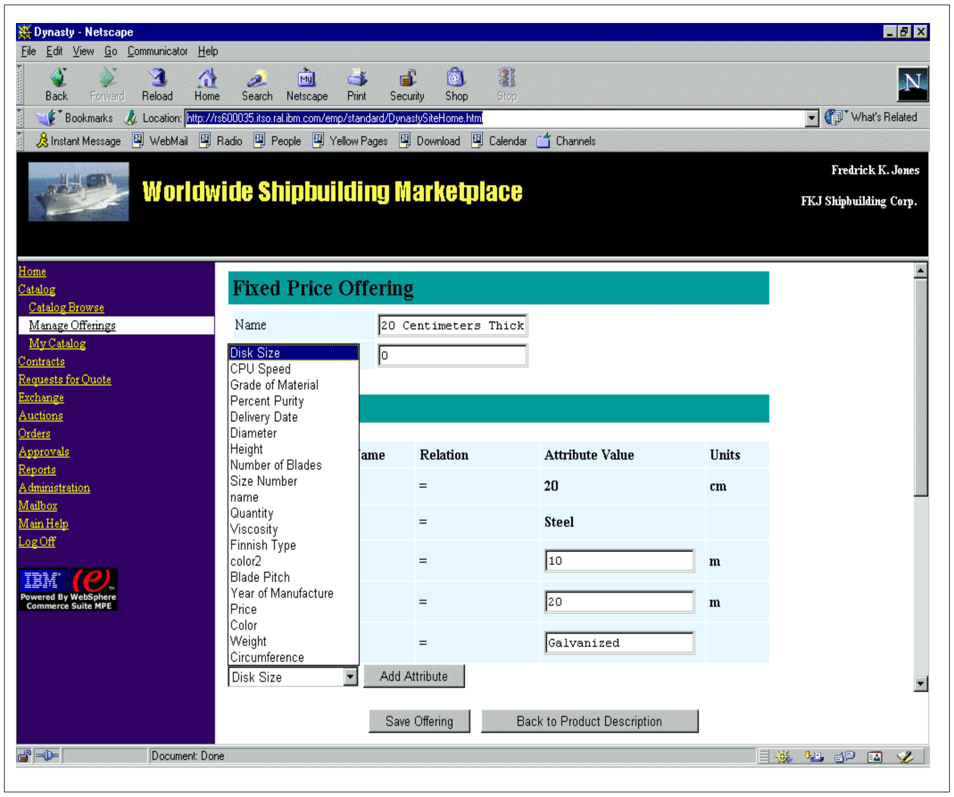
To add a new attribute to the data dictionary perform the following steps:

1. Log on to the Web-based administrative tool. Use “ncadmin” to log on.

2. Click on **Site Manager** tab.
3. Select **Attribute Dictionary.**
4. Fill in the form.
5. Click **Create.**
6. If a message stating successful creation of an attribute is displayed, then you are done. Otherwise review and correct any errors.
7. You can also use the **Search** button to verify that your entry has been successfully created.
8. After creation, you must click **Refresh Registry** from the Site Manager menu to update registry entries.

We will now provide a series of graphics to demonstrate the above-mentioned process and the effects of it. First we create the Frequency attribute and publish it, so that we can see it as part of the available attributes in the e-Marketplace.

Figure 81 on page 311 shows the list of attributes before creation of the Frequency attribute. For ease of demonstration we have selected the Create Fix price offering window to demonstrate the list of attributes.


Dynasty - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Shop Stop
Bookmarks Location: http://rs600035.itso.ral.ibm.com/emp/standard/DynastySiteHome.html
What's Related
Instant Message WebMail Radio People Yellow Pages Download Calendar Channels
Worldwide Shipbuilding Marketplace
Fredrick K. Jones
FKJ Shipbuilding Corp.
Home
Catalog
Catalog Browse
Manage Offerings
My Catalog
Contracts
Requests for Quote
Exchange
Auctions
Orders
Approvals
Reports
Administration
Mailbox
Main Help
Log Off
Powered By WebSphere Commerce Suite MPE
Document: Done

Fixed Price Offering

Name	20 Centimeters Thick
	0

Disk Size
CPU Speed
Grade of Material
Percent Purity
Delivery Date
Diameter
Height
Number of Blades
Size Number
name
Quantity
Viscosity
Finnish Type
color2
Blade Pitch
Year of Manufacture
Price
Color
Weight
Circumference

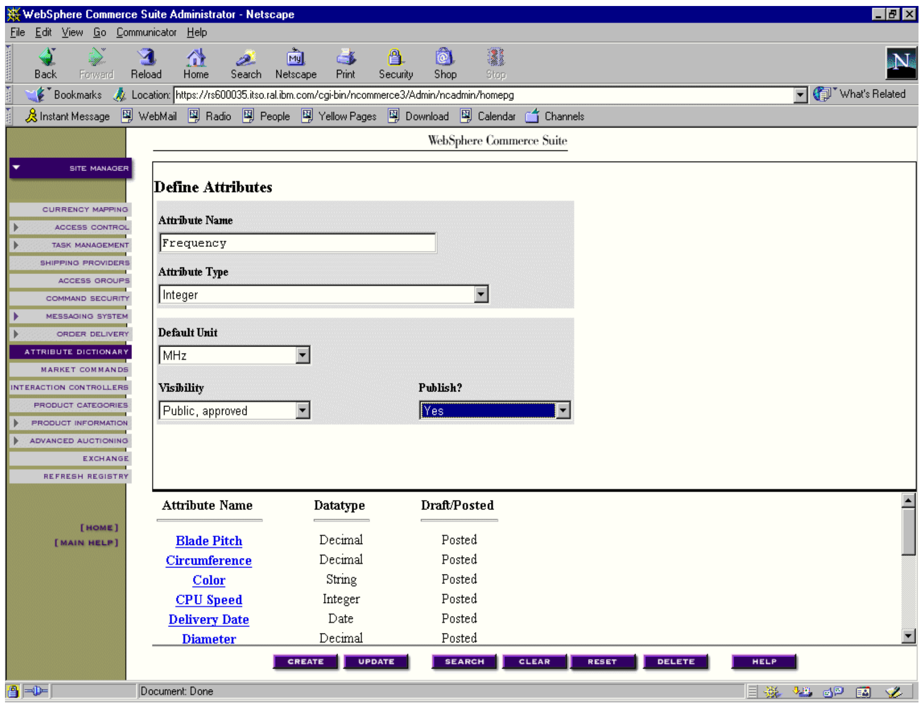
ame	Relation	Attribute Value	Units
	=	20	cm
	=	Steel	
	=	10	m
	=	20	m
	=	Galvanized	

Disk Size — Add Attribute

Save Offering — Back to Product Description

Figure 81. Attribute list before creating frequency

Figure 82 on page 312 shows the Define Attributes page of the Web-based administration tool where we have input the information for the Frequency attribute. Information from this page is stored in Table 28 on page 305. Refer to the column definitions of this table for a better understanding of what each of the options in the window represent.



Define Attributes

Attribute Name: Frequency

Attribute Type: Integer

Default Unit: MHz

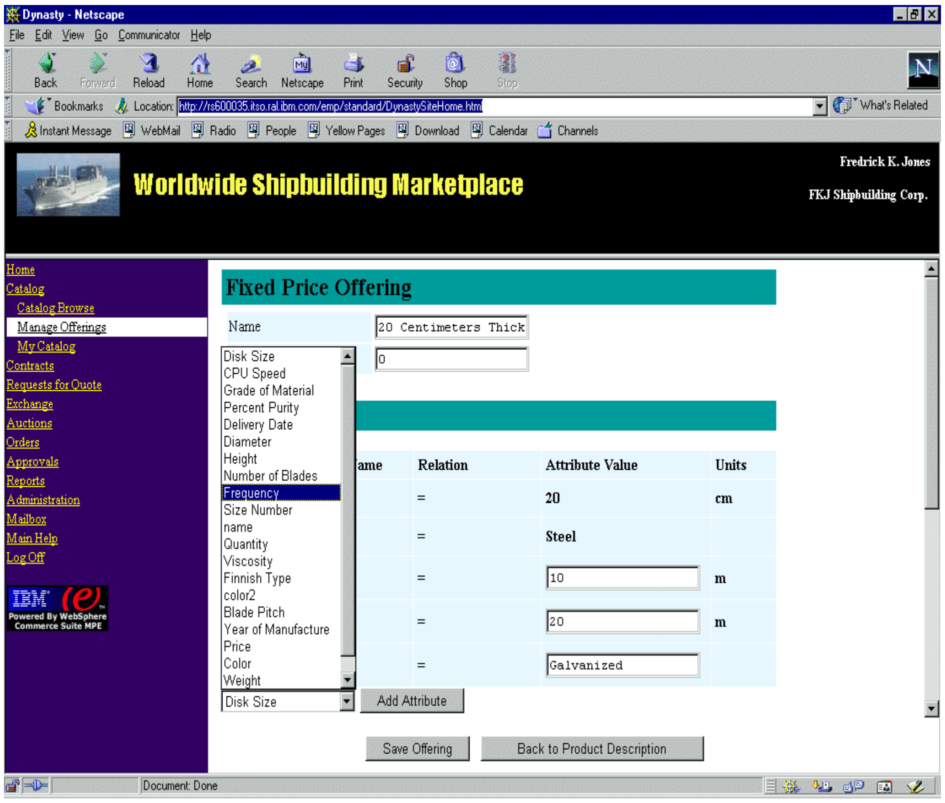
Visibility: Public, approved

Publish?: Yes

Attribute Name	Datatype	Draft/Posted
Blade Pitch	Decimal	Posted
Circumference	Decimal	Posted
Color	String	Posted
CPU Speed	Integer	Posted
Delivery Date	Date	Posted
Diameter	Decimal	Posted

Figure 82. Create data dictionary attribute

Figure 83 on page 313 shows the same window as Figure 81 on page 311 but now we see that the Frequency attribute has been added to the list of attributes.



ame	Relation	Attribute Value	Units
	=	20	cm
	=	Steel	
	=	10	m
	=	20	m
	=	Galvanized	

Name: 20 Centimeters Thick

0

Figure 83. Attribute list after creating frequency

14.3.1.2 Modifying attribute definition

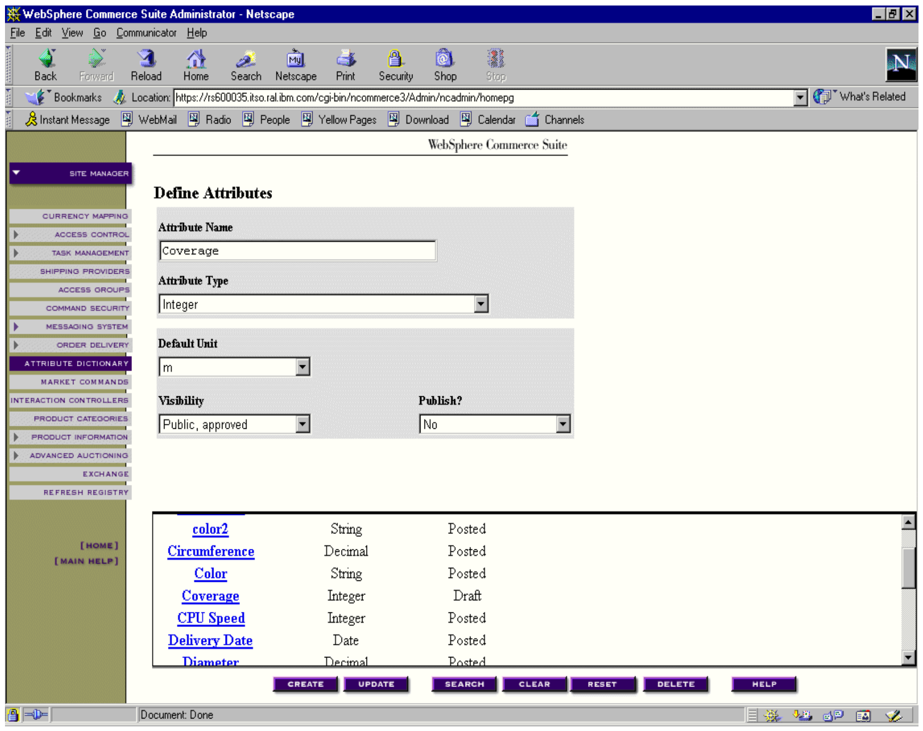
To modify an attribute definition in the data dictionary that is still in Draft state perform the following:

1. Log on to the Web-based administrative tool. Use “ncadmin” to log on.
2. Click the Site Manager tab.
3. Select **Attribute Dictionary.**
4. Click the **Search** button and select the attribute you wish to modify.
5. Make the changes and click on **Update** button.
6. If a message stating successful update is displayed, then you are done. Otherwise review your entries and correct any errors.
7. You can also use the **Search** button again to retrieve the attribute and verify that your changes have been successfully recorded.

8. If you modified the entry and decided to publish the attribute make sure you click **Refresh Registry** from the Site Manager menu to update registry entries.

To avoid unnecessary duplication we will not show the initial steps of creating the Coverage attribute. That process is identical to the process of creating the Frequency attribute, except that we selected not to publish the attribute to the data dictionary by setting the Publish field value to No.

Figure 84 shows the Coverage attribute that we created and verifies that it is in Draft state. Note that we have selected m (Meters) for the Default Unit, which is incorrect. By following the steps mentioned above we will change the Default Unit to Km (Kilometers) and change the Publish valueto Yes.


WebSphere Commerce Suite Administrator - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Shop Stop
Bookmarks Location: https://rs600035.itso.ral.ibm.com/cgi-bin/ncommerce3/Admin/ncadmin/homepg
What's Related
Instant Message WebMail Radio People Yellow Pages Download Calendar Channels
SITE MANAGER
CURRENCY MAPPING
ACCESS CONTROL
TASK MANAGEMENT
SHIPPING PROVIDERS
ACCESS GROUPS
COMMAND SECURITY
MESSAGING SYSTEM
ORDER DELIVERY
ATTRIBUTE DICTIONARY
MARKET COMMANDS
INTERACTION CONTROLLERS
PRODUCT CATEGORIES
PRODUCT INFORMATION
ADVANCED AUCTIONING
EXCHANGE
REFRESH REGISTRY
[HOME]
[MAIN HELP]
WebSphere Commerce Suite
CREATE UPDATE SEARCH CLEAR RESET DELETE HELP
Document: Done

Define Attributes

Attribute Name: Coverage

Attribute Type: Integer

Default Unit: m

Visibility: Public, approved

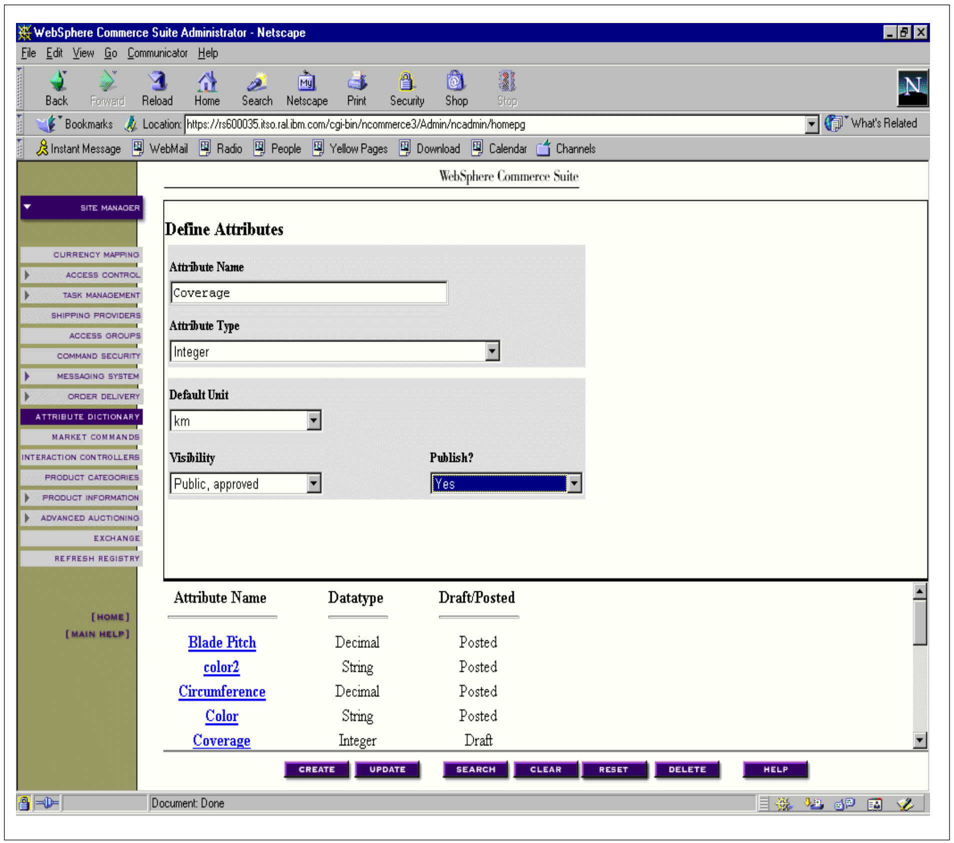
Publish?: No

color2	String	Posted
Circumference	Decimal	Posted
Color	String	Posted
Coverage	Integer	Draft
CPU Speed	Integer	Posted
Delivery Date	Date	Posted
Diameter	Decimal	Posted

Figure 84. Coverage attribute in draft state

Figure 85 on page 315 shows the modifications to the Coverage attribute. At this stage we have not yet updated the record, which is why the attribute listing at the lower part of the window still shows Draft status for this attribute.

After clicking the **Update** button and receiving a confirmation of a successful update, the status will change to Posted.



Define Attributes

Attribute Name: Coverage

Attribute Type: Integer

Default Unit: km

Visibility: Public, approved

Publish?: Yes

Attribute Name	Datatype	Draft/Posted
Blade Pitch	Decimal	Posted
color2	String	Posted
Circumference	Decimal	Posted
Color	String	Posted
Coverage	Integer	Draft

Figure 85. Coverage attribute modification window

14.3.2 Creation and maintenance of category hierarchy

Using the Web-based administrative tool you can add a new parent category, remove a parent category, manage parent categories, add and delete a product category or move a category. These functions are similar to those in the WebSphere Commerce Suite 4.1. In this section we will provide step-by-step directions for all these operations and use screen captures to show adding a new product category to the catalog.

14.3.2.1 Add a Parent Category

You can use this operation to assign a product description to a parent category or add the product description to more than one parent category if you wish to. The following steps allow you to perform this operation.

1. Log on to the Web-based administrative tool. Uuse “ncadmin” to log on.
2. Click on **Site Manager** tab.
3. Select **Product Information**.
4. From the Product Description Information page, click the **Search** button and select the Product Description you wish to work with.
5. Click the **Parent Categories** link under the Product Information link. This will display the Manage Parent Categories window.
6. Click **Select Parent...** button and from the next window navigate the catalog hierarchy and select the new parent category reference number.
7. Return to the Manage Parent Categories window and type in the category reference number in the input field next to the **Add** button.
8. Click the **Add** button.
9. When this operation completes successfully the new parent category name and reference number will appear in drop-down lists of both Add/Remove Parent and Define Presentation Order section.
10. Verify that your change has been successfully recorded.
11. Click **Refresh Registry** from the Site Manager menu to update registry entries to reflect your changes to the e-Marketplace catalog.

14.3.2.2 Remove a Parent Category

You can use this operation to remove a product description from a parent category. The following steps allow you to perform this operation.

1. Logon to the Web-based administrative tool. Use “ncadmin” to log on.
2. Click the **Site Manager** tab.
3. Select **Product Information**
4. From the Product Description Information page, click the **Search** button and select the Product Description you wish to work with.
5. Click the **Parent Categories** link under the Product Information link. This will display the Manage Parent Categories window.
6. From the drop-down list of the Add/Remove Parent section select a parent category and click the **Remove** button.

7. When this operation completes successfully, the selected parent category name and reference number will no longer appear in drop-down lists of either the Add/Remove Parent or Define Presentation Order section.
8. Verify that your change has been successfully recorded.
9. Click **Refresh Registry** from the Site Manager menu to update registry entries to reflect your changes to the e-Marketplace catalog.

14.3.2.3 Manage Parent Categories

You can use this operation to modify a product description's parent category information. The following steps allow you to perform this operation.

1. Log on to the Web-based administrative tool. Use ncadmin to log on.
2. Click the **Site Manager** tab.
3. Select **Product Information**.
4. From the Product Description Information page click the **Search** button and select the Product Description you wish to work with.
5. Click the **Parent Categories** link under the Product Information link. This will display the Manage Parent Categories"window.
6. Here you can modify the parent category information.
7. Click the **Update Form** button to store your changes in the database.
8. Verify that your change has been successfully ecorded.
9. Click **Refresh Registry** from the Site Manager" menu to update registry entries to reflect your changes to the e-Marketplace catalog.

14.3.2.4 Add a Product Category

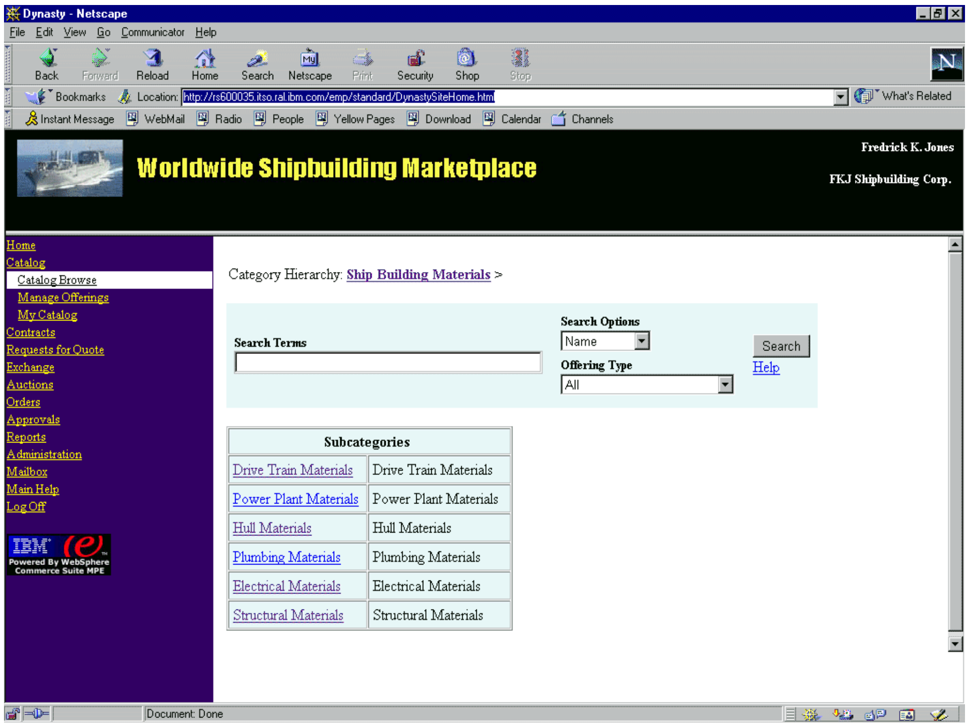
You can use this operation to create a new category in your category hierarchy. The following steps allow you to perform this operation.

1. Log on to the Web-based administrative tool. Use ncadmin to log on.
2. Click the **Site Manager** tab.
3. Select **Product Categories**
4. In the Product Categories page navigate through the catalog hierarchy and find the category under which you wish to create a new category. Note the new category will be placed under the category which is highlighted in red.
5. Click the **Add** button. The Add New Category form will be displayed.
6. Fill in the information on this form and click **Save**.

7. You will be returned to the Product Categories page with your newly created category showing up under the category that you had selected earlier.
8. Click **Refresh Registry** from the Site Manager menu to update registry entries to reflect your changes to the e-Marketplace catalog.

If you wish to add subcategories to your newly created category repeat these steps again.

We will demonstrate creation of the Communication Equipment category here as an example. We will also create all subcategories we mentioned at the beginning of this section. However, we will not include screen captures since they all follow the same steps.


Dynasty - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Shop Stop
Bookmarks Location: http://rs600035.itso.ral.ibm.com/emp/standard/DynastySiteHome.html
What's Related
Instant Message WebMail Radio People Yellow Pages Download Calendar Channels
Worldwide Shipbuilding Marketplace
Fredrick K. Jones
FKJ Shipbuilding Corp.
Home
Catalog
Catalog Browse
Manage Offerings
My Catalog
Contracts
Requests for Quote
Exchange
Auctions
Orders
Approvals
Reports
Administration
Mailbox
Main Help
Log Off
Powered By WebSphere Commerce Suite MPE
Search Terms
Search Options
Name
Offering Type
All
Search
Help
Document: Done

Category Hierarchy: Ship Building Materials >

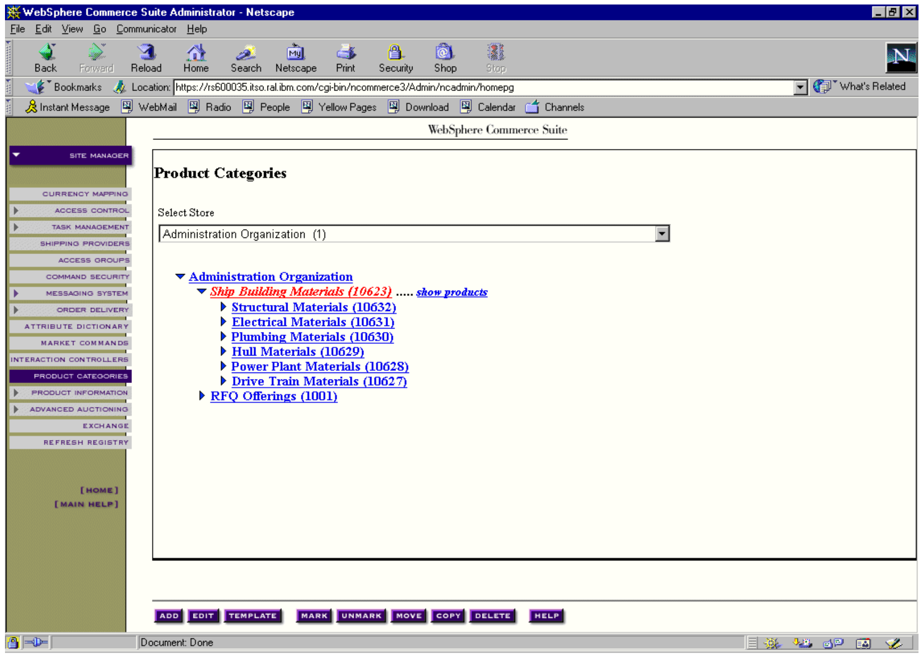
Subcategories	
Drive Train Materials	Drive Train Materials
Power Plant Materials	Power Plant Materials
Hull Materials	Hull Materials
Plumbing Materials	Plumbing Materials
Electrical Materials	Electrical Materials
Structural Materials	Structural Materials

Figure 86. e-Marketplace category structure.

Figure 86 shows the category structure before addition of the Communication Equipment category.

We would like to add this category under the Ship Building Materials category. Figure 87 shows the category structure, we have selected the Ship

Building Materials category. On this window you would click the **Add** button to create the new category.


WebSphere Commerce Suite Administrator - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Shop Stop
Bookmarks Location: https://rs600035.itso.ral.ibm.com/cgi-bin/ncommerce3/Admin/ncadmin/homepg What's Related
Instant Message WebMail Radio People Yellow Pages Download Calendar Channels
WebSphere Commerce Suite
SITE MANAGER
CURRENCY MAPPING
ACCESS CONTROL
TASK MANAGEMENT
SHIPPING PROVIDERS
ACCESS GROUPS
COMMAND SECURITY
MESSAGING SYSTEM
ORDER DELIVERY
ATTRIBUTE DICTIONARY
MARKET COMMANDS
INTERACTION CONTROLLERS
PRODUCT CATEGORIES
PRODUCT INFORMATION
ADVANCED AUCTIONING
EXCHANGE
REFRESH REGISTRY
[HOME]
[MAIN HELP]
ADD EDIT TEMPLATE MARK UNMARK MOVE COPY DELETE HELP
Document: Done

Product Categories

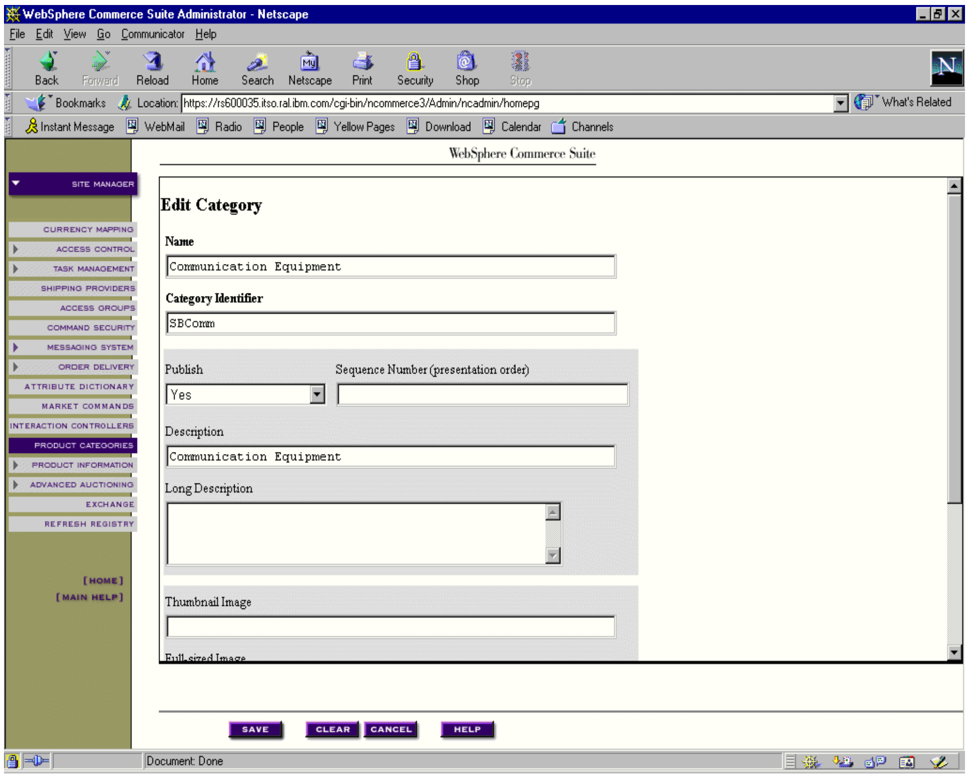
Select Store

Administration Organization (1)

- Administration Organization
  - *Ship Building Materials (10623)* ..... *show products*
    - Structural Materials (10632)
    - Electrical Materials (10631)
    - Plumbing Materials (10630)
    - Hull Materials (10629)
    - Power Plant Materials (10628)
    - Drive Train Materials (10627)
  - RFQ Offerings (1001)

Figure 87. Product categories window

Figure 88 on page 320 shows the Add New Category window. We have provided the Name, Category Identifier and a Short Description for the new category, if you wish you can provide more information. When you click the **Save** button the new category will be created.



Edit Category

Name: Communication Equipment

Category Identifier: SBComm

Publish: Yes

Sequence Number (presentation order):

Description: Communication Equipment

Long Description:

Thumbnail Image:

Full-sized Image

Figure 88. Add new category

Figure 89 on page 321 shows the newly created Communication Equipment under the Ship Building Materials category.

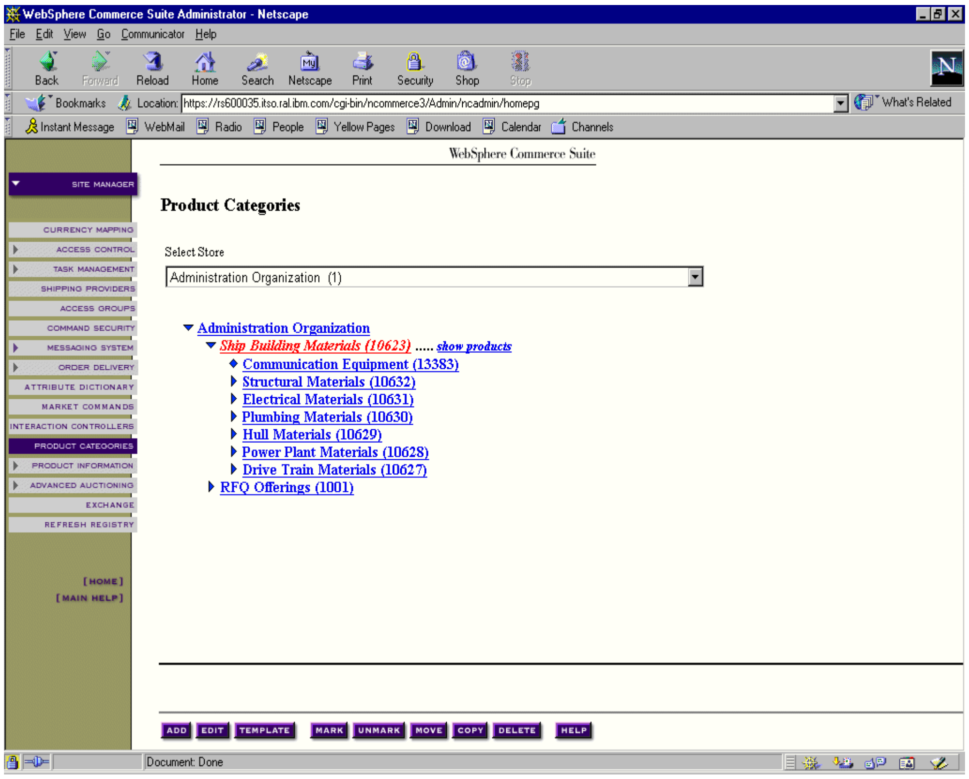
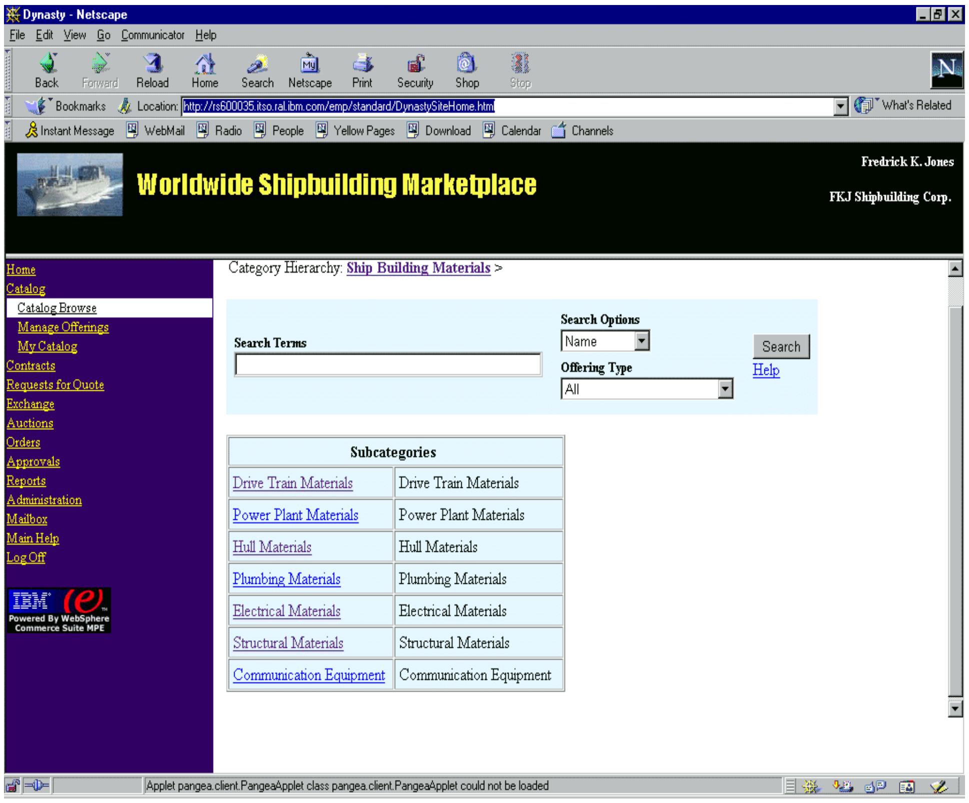

WebSphere Commerce Suite Administrator - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Shop Stop
Bookmarks Location: https://rs600035.itso.ral.ibm.com/cgi-bin/ncommerce3/Admin/ncadmin/homepg
What's Related
Instant Message WebMail Radio People Yellow Pages Download Calendar Channels
SITE MANAGER
CURRENCY MAPPING
ACCESS CONTROL
TASK MANAGEMENT
SHIPPING PROVIDERS
ACCESS GROUPS
COMMAND SECURITY
MESSAGING SYSTEM
ORDER DELIVERY
ATTRIBUTE DICTIONARY
MARKET COMMANDS
INTERACTION CONTROLLERS
PRODUCT CATEGORIES
PRODUCT INFORMATION
ADVANCED AUCTIONING
EXCHANGE
REFRESH REGISTRY
[HOME]
[MAIN HELP]
WebSphere Commerce Suite
Product Categories
Select Store
Administration Organization (1)
Administration Organization
Ship Building Materials (10623) ..... show products
Communication Equipment (13383)
Structural Materials (10632)
Electrical Materials (10631)
Plumbing Materials (10630)
Hull Materials (10629)
Power Plant Materials (10628)
Drive Train Materials (10627)
RFQ Offerings (1001)
ADD
EDIT
TEMPLATE
MARK
UNMARK
MOVE
COPY
DELETE
HELP
Document: Done

Figure 89. New category structure

Figure 90 on page 322 shows the e-Marketplace category structure after adding Communication Equipment.


Dynasty - Netscape

http://rs600035.itso.ral.ibm.com/emp/standard/DynastySiteHome.html

Worldwide Shipbuilding Marketplace

Fredrick K. Jones

FKJ Shipbuilding Corp.

Home
Catalog
Catalog Browse
Manage Offerings
My Catalog
Contracts
Requests for Quote
Exchange
Auctions
Orders
Approvals
Reports
Administration
Mailbox
Main Help
Log Off

Category Hierarchy: Ship Building Materials >

Search Terms

Search Options: Name

Offering Type: All

Subcategories	
Drive Train Materials	Drive Train Materials
Power Plant Materials	Power Plant Materials
Hull Materials	Hull Materials
Plumbing Materials	Plumbing Materials
Electrical Materials	Electrical Materials
Structural Materials	Structural Materials
Communication Equipment	Communication Equipment

Applet pangea.client.PangeaApplet class pangea.client.PangeaApplet could not be loaded

Figure 90. e-Marketplace category structure after creation of Communication Equipment category

14.3.2.5 Delete a product category

You can use this operation to delete a category in your category hierarchy. The following steps allow you to perform this operation.

1. Log on to the Web-based administrative tool. Use ncadmin to log on..
2. Click the **Site Manager** tab.
3. Select **Product Categories**.
4. In the Product Categories page navigate through the catalog hierarchy and find the category which you wish to delete.
5. Click the **Delete** button.
6. After deletion is performed you will be returned to the Product Categories page. The category you had selected for deletion will no longer appear in the category structure.

7. Click **Refresh Registry** from the Site Manager menu to update registry entries to reflect your changes to the e-Marketplace catalog.

14.3.2.6 Move a Product Category

You can use this operation to move a category in your category hierarchy. The following steps allow you to perform this operation.

1. Log on to the Web-based administrative tool. Use ncadmin to log on..
2. Click the **Site Manager** tab.
3. Select **Product Categories**.
4. In the Product Categories page navigate through the catalog hierarchy and find the category which you wish to move.
5. Click the **Mark** button. The selected category name will start to flash.
6. Select the destination for the category. The selected destination category will be highlighted.
7. Click **Move**.
8. Click the triangle to the left of the destination category to verify the modified hierarchy.
9. Click **Refresh Registry** from the Site Manager menu to update registry entries to reflect your changes to the e-Marketplace catalog.

14.3.3 Populating the catalog

In this section we will describe the steps required to set up product descriptions in the catalog hierarchy. Product descriptions provide the catalog taxonomy as we described it earlier in this chapter. By creating product descriptions in a specific subcategory of the catalog we provide a method to create product offerings that are arranged in the predefined subcategory and have a set of established base attributes. This allows consistency in the catalog both for users and administrators.

Product descriptions can be modified and managed through the Web-based administrative tool. In this section we cover the creation of a new product description and adding attributes to that product description. We will not detail the managing of product descriptions as they largely follow the same steps and use the same windows as creating a new product description.

14.3.3.1 Create a product description.

You can create new product descriptions in your category hierarchy so that different organizations can create offerings. The following steps allow you to perform this operation.

1. Log on to the Web-based administrative tool. Use ncadmin to log on.
2. Click the **Site Manager** tab.
3. Select **Product Information**.
4. In the Product Description Information page click **Select Parent**. From the Product Categories window navigate through the catalog hierarchy and find the category to which you wish to add the product description to. Make a note of the Category Reference number.
5. Return to the Product description Information page and place the Category Reference number into the Parent Reference Number field.
6. Select whether you wish to publish the product description or not.
7. Provide information for the remaining fields in this form.
8. Click the **Create** button to create the product description in the database.

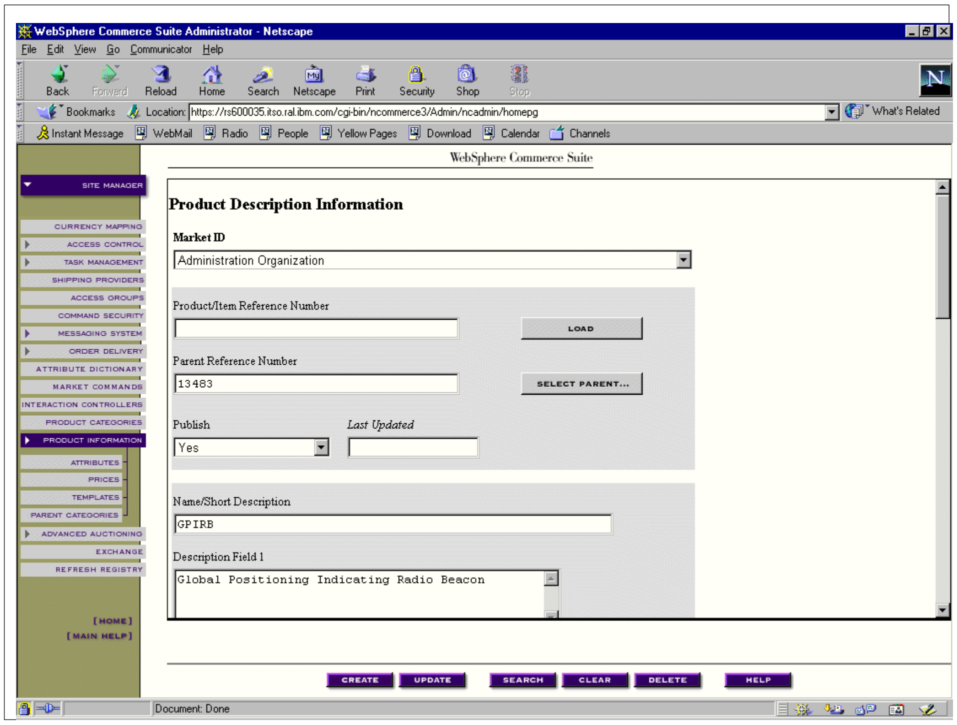
Next you need to add attributes to the product description:

1. Sselect **Product Information**.
2. Click **Search**.
3. Select the product description you just created.
4. Click the **Attributes** link under the Product Information.
5. In the Create/Remove Attribute"section select an attribute you wish to add to the Product description and click **Create**. Repeat this step for all the different attributes you wish to assign to the product description.
6. In the Define Attribute section, select one of the attributes just added to the product description.
7. Provide all the information you wish to assign to this attribute.
8. Click the **Update** button.
9. Click **Refresh Registry** from the Site Manager menu to update registry entries to reflect your changes to the e-Marketplace catalog.

You can use this same process to update product description information to manage attributes of a product description, etc. For more details on product description management, please see the online documentation.

We will now provide some window captures to better demonstrate the above-mentioned process.

Figure 91 shows the window in the Web based administrative tool that allows creation of product description.



Product Description Information

Market ID: Administration Organization

Product/Item Reference Number:

Parent Reference Number: 13483

Publish: Yes

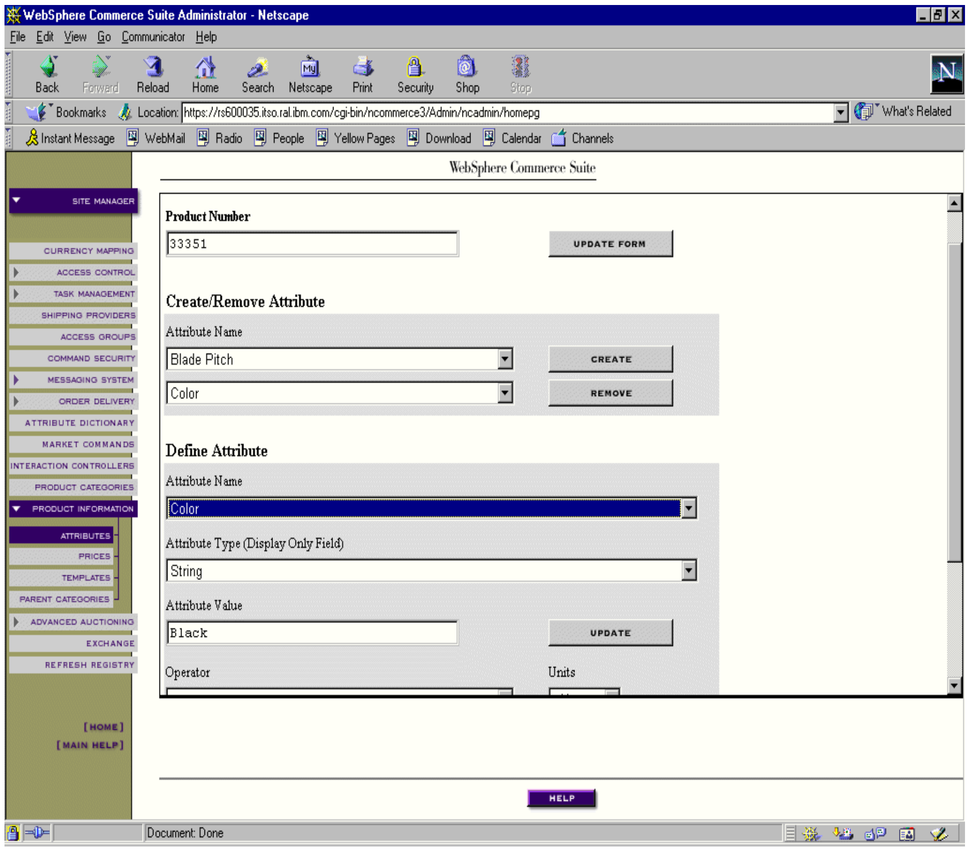
Last Updated:

Name/Short Description: GPIRB

Description Field 1: Global Positioning Indicating Radio Beacon

Figure 91. Create product description

Figure 92 on page 326 shows the window where attributes are assigned and managed for a given product description.



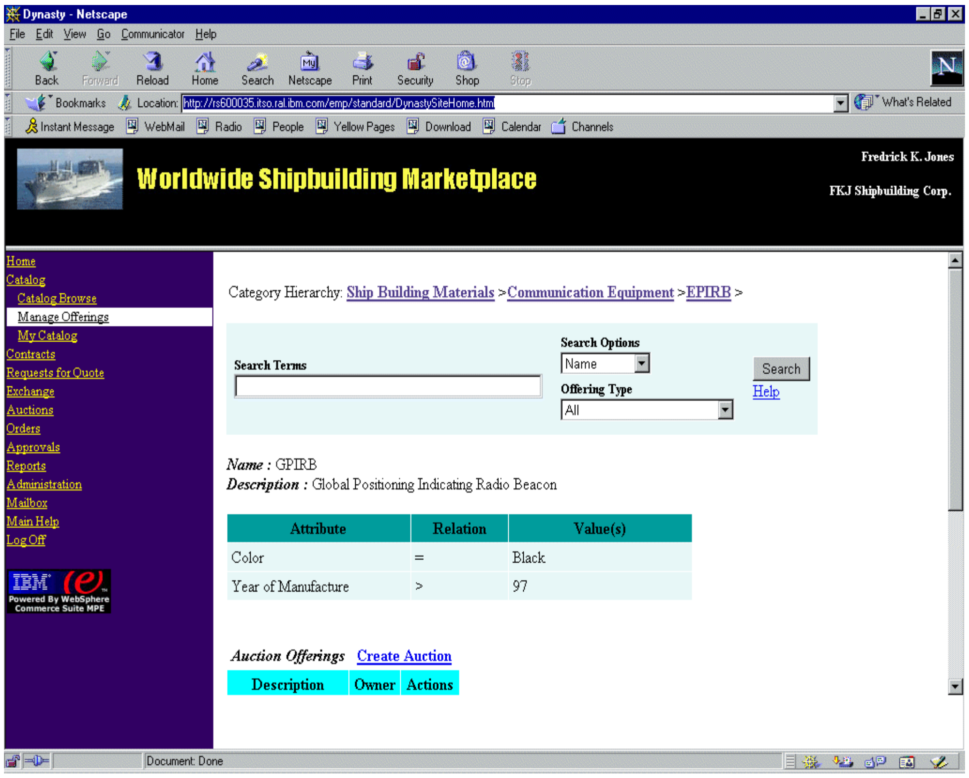
Product Number
33351

Create/Remove Attribute
Attribute Name
Blade Pitch
Color

Define Attribute
Attribute Name
Color
Attribute Type (Display Only Field)
String
Attribute Value
Black
Operator
Units

Figure 92. Add attributes

Figure 93 on page 327 shows the new product description in the e-Marketplace catalog.


Category Hierarchy: Ship Building Materials > Communication Equipment > EPIRB >

Name : GPIRB
Description : Global Positioning Indicating Radio Beacon

Attribute	Relation	Value(s)
Color	=	Black
Year of Manufacture	>	97

Auction Offerings Create Auction

Description	Owner	Actions

Figure 93. New product description in the e-Marketplace catalog

14.4 Supplier interaction: offering creation and maintenance

Organization members with supplier authority can interact with the catalog subsystem to perform a number of activities. The following list states some of the activities a typical supplier performs:

- Create and manage offerings
- Review and respond to RFQs
- Create and manage auctions
- Participate in exchanges
- Create and manage contracts
- Check orders from buyers
- Generate reports

The Figure 94 describes the supplier or seller use cases in the e-Marketplace.

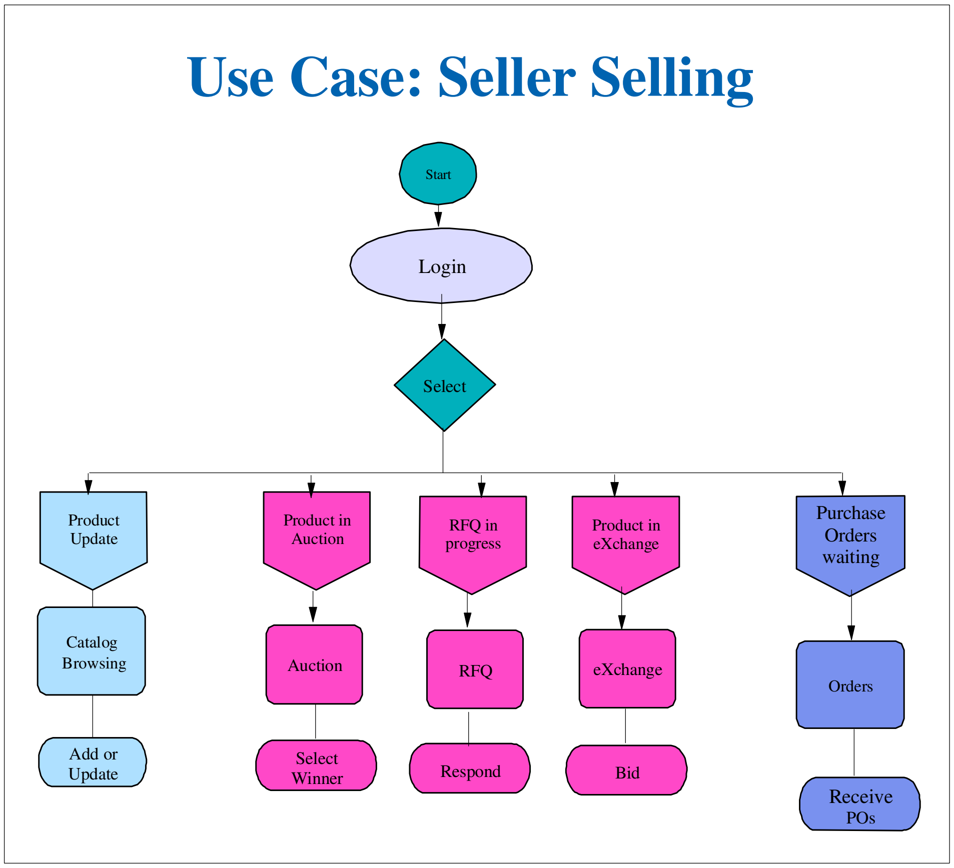


Figure 94. Supplier use case diagram

In this section we will cover the creation of a fixed price offering by a supplier in the e-Marketplace. Other supplier activities mentioned in the above list are covered as separate items in different chapters of this book. Chapter 15, “Example - pricing and contract subsystem” on page 341 describes contracts, Chapter 16, “Example - negotiation subsystem” on page 367 describes RFQs, auctions and exchanges and Chapter 17, “Example - additional e-Marketplace infrastructure” on page 441 describes orders and reports in detail.

14.4.1 Create a fixed price offering

In order to perform this task an organization must be registered and approved in the e-Marketplace. Also a supplier must be created and assigned the appropriate role. These topics are covered in detail in Chapter 13, “Example - membership and access control” on page 241.

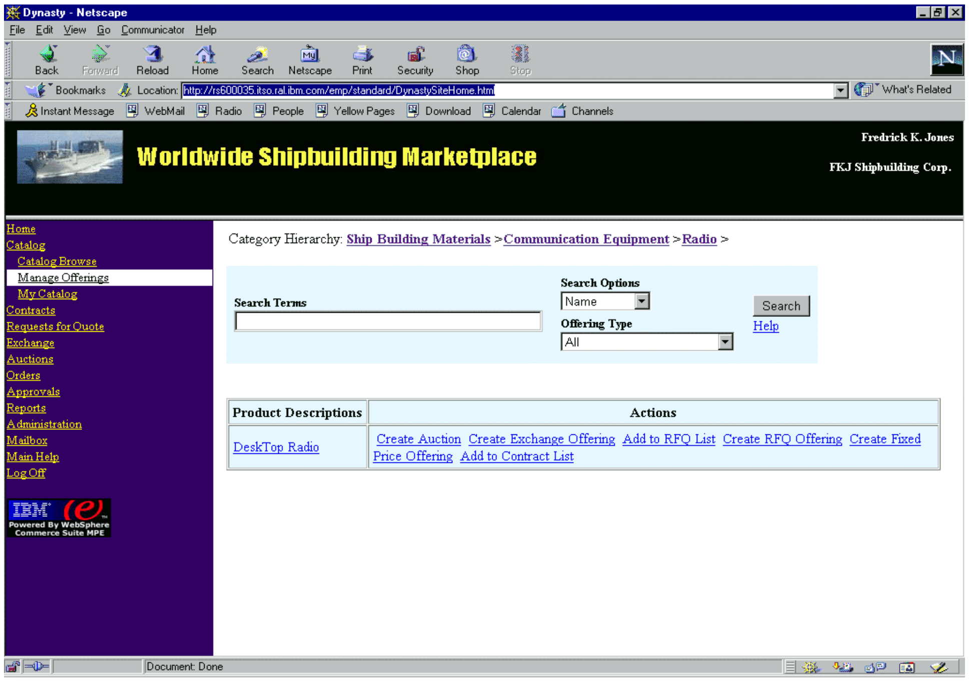
We will describe the steps to create a fixed price offering. Later on we will create an offering for the DeskTop Radio product description we created in the e-Marketplace catalog earlier to give you a visual presentation of windows.

Typically the following takes place when creating a fixed price offering.

1. Supplier logs on to the e-Marketplace.
2. Supplier navigates through the catalog hierarchy using Manage Offerings view.
   - Supplier finds the product description for which a fixed price offering is to be created.
   - From the product description window, the supplier selects **Create Fixed price offering**.
   - Supplier completes the form, determines if new attributes need to be added to the product offering and if the offering is targeted to a certain organization or it is a public offering.
   - After completion of the form the supplier selects **Save offering** to complete the fix price offering creation process.

The following few pages provide a visual representation of above mentioned procedure.

After successful logon, the authorized supplier would click the **Manage Catalog** link to navigate to the product description window.


Dynasty - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Shop Stop
Bookmarks Location: http://rs600035.itso.ral.ibm.com/emp/standard/DynastySiteHome.htm
What's Related
Instant Message WebMail Radio People Yellow Pages Download Calendar Channels
Worldwide Shipbuilding Marketplace
Fredrick K. Jones
FKJ Shipbuilding Corp.
Home
Catalog
Catalog Browse
Manage Offerings
My Catalog
Contracts
Requests for Quote
Exchange
Auctions
Orders
Approvals
Reports
Administration
Mailbox
Main Help
Log Off
IBM Powered By WebSphere Commerce Suite MPE
Search Terms
Search Options
Name
Offering Type
All
Search
Help
Document: Done

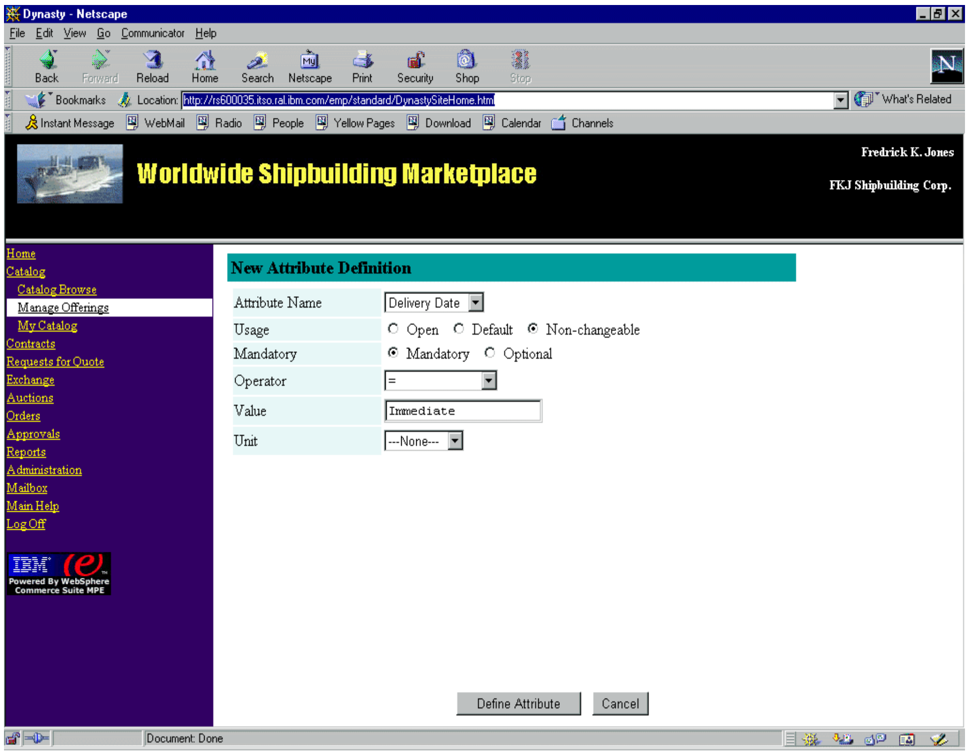
Category Hierarchy: Ship Building Materials > Communication Equipment > Radio >

Product Descriptions	Actions
DeskTop Radio	Create Auction Create Exchange Offering Add to RFQ List Create RFQ Offering Create Fixed Price Offering Add to Contract List

Figure 95. Actions available on DeskTop Radio product description

Figure 95 shows the DeskTop Radio product description entry we created earlier. Note that the Category Hierarchy describes our position in the e-Marketplace catalog. Under the Actions column we see all the actions that can be performed on the this product description. We will select the link **Create Fixed price offering**.

While in the Fixed price offering creation window, you can see that the product description attributes that were assigned during the creation of the product description earlier are displayed here as the standard attributes. From this window you must add inventory and price information and you must select if the offering is restricted to certain organizations or if it is a public offering. You also have some optional fields to add information specific to your product. You can add more attributes and assign values to them. We will add the Deliver Date option with the value of Immediate to indicate that the product is available and ready to be shipped to buyers immediately.



Dynasty - Netscape

Location: http://rs600035.itso.ral.ibm.com/emp/standard/DynastySiteHome.html

Worldwide Shipbuilding Marketplace

Fredrick K. Jones

FKJ Shipbuilding Corp.

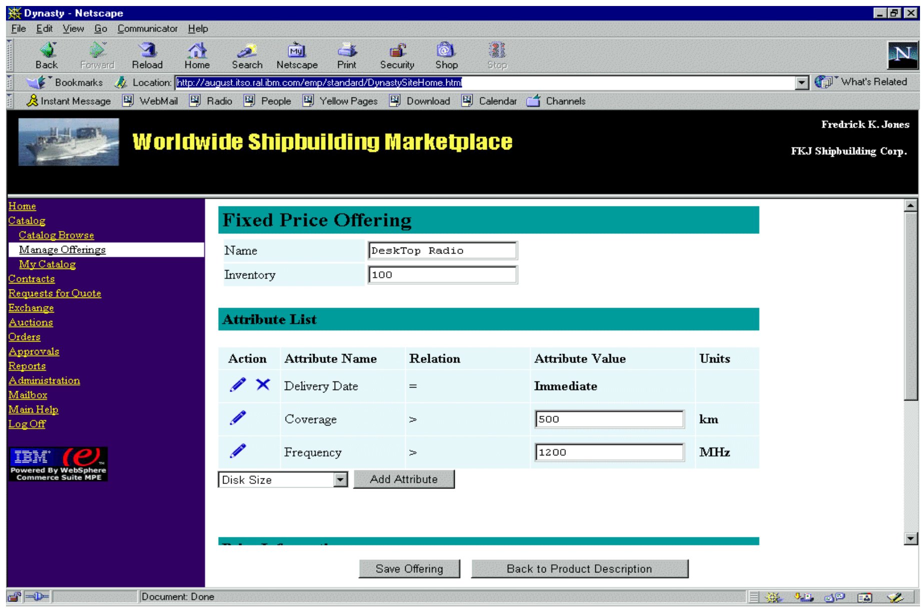
New Attribute Definition

Attribute Name	Delivery Date
Usage	Open Default [✓ Non-changeable]
Mandatory	[✓ Mandatory] Optional
Operator	=
Value	Immediate
Unit	---None---

Figure 96. Adding a new attribute

Figure 96 shows the New Attribute Definition window.

Figure 97 on page 332 and Figure 98 on page 332 together show all the fields in the create fixed price offering window. Next we click the **Save Offering** button to create the offering in the e-Marketplace catalog.


Dynasty - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Shop Stop
Bookmarks Location: http://august.itso.ral.ibm.com/emp/standard/DynastySiteHome.html What's Related
Instant Message WebMail Radio People Yellow Pages Download Calendar Channels
Worldwide Shipbuilding Marketplace
Fredrick K. Jones
FKJ Shipbuilding Corp.
Home
Catalog
Catalog Browse
Manage Offerings
My Catalog
Contracts
Requests for Quote
Exchange
Auctions
Orders
Approvals
Reports
Administration
Mailbox
Main Help
Log Off
IBM Powered By WebSphere Commerce Suite MPE
Document: Done

Fixed Price Offering

Name	DeskTop Radio
Inventory	100

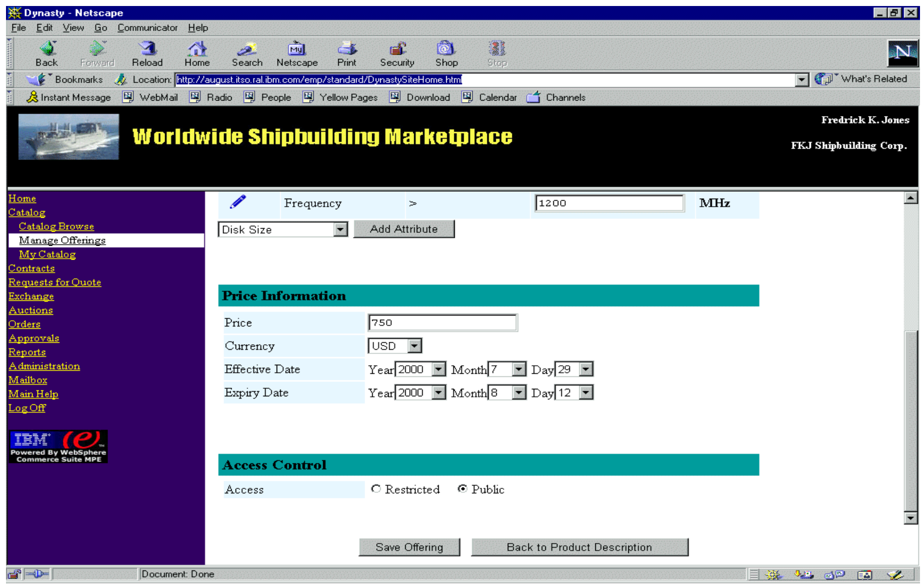
Attribute List

Action	Attribute Name	Relation	Attribute Value	Units
✎ ✕	Delivery Date	=	Immediate	
✎	Coverage	>	500	km
✎	Frequency	>	1200	MHz

Disk Size | Add Attribute

Save Offering | Back to Product Description

Figure 97. Create offering: top portion of window


Dynasty - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Shop Stop
Bookmarks Location: http://august.itso.ral.ibm.com/emp/standard/DynastySiteHome.html What's Related
Instant Message WebMail Radio People Yellow Pages Download Calendar Channels
Worldwide Shipbuilding Marketplace
Fredrick K. Jones
FKJ Shipbuilding Corp.
Home
Catalog
Catalog Browse
Manage Offerings
My Catalog
Contracts
Requests for Quote
Exchange
Auctions
Orders
Approvals
Reports
Administration
Mailbox
Main Help
Log Off
IBM Powered By WebSphere Commerce Suite MPE
Document: Done

✎	Frequency	>	1200	MHz

Disk Size | Add Attribute

Price Information

Price	750
Currency	USD
Effective Date	Year 2000 Month 7 Day 29
Expiry Date	Year 2000 Month 8 Day 12

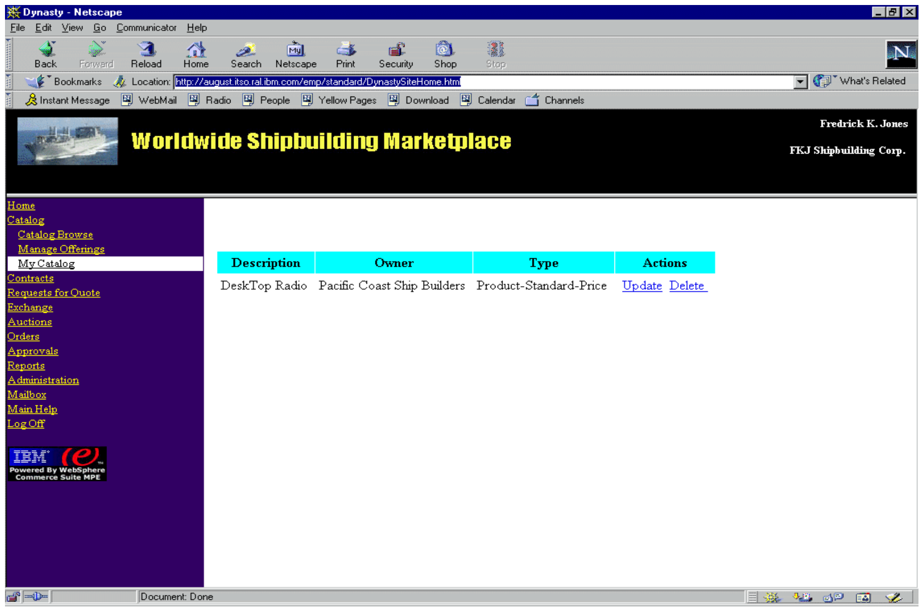
Access Control

Access	Restricted [✓ Public]

Save Offering | Back to Product Description

Figure 98. Create offering: bottom portion of the window

After the offering has been created if the supplier clicks on the My Catalog view link all the offerings that have been created by that supplier will be displayed. Since this is the first offering created by this supplier, Figure 99 shows only one entry. The supplier can work with the offering using the action links provided in this view of the catalog.


Dynasty - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Shop Stop
Bookmarks Location: http://august.itso.ral.ibm.com/emp/standard/DynastySiteHome.htm
What's Related
Instant Message WebMail Radio People Yellow Pages Download Calendar Channels
Worldwide Shipbuilding Marketplace
Fredrick K. Jones
FKJ Shipbuilding Corp.
Home
Catalog
Catalog Browse
Manage Offerings
My Catalog
Contracts
Requests for Quote
Exchange
Auctions
Orders
Approvals
Reports
Administration
Mailbox
Main Help
Log Off
IBM Powered By WebSphere Commerce Suite MPE
Document: Done

Description	Owner	Type	Actions
DeskTop Radio	Pacific Coast Ship Builders	Product-Standard-Price	Update Delete

Figure 99. My catalog view

14.5 Buyer interaction: examples of catalog based buying

Organization members with buyer authority can interact with the catalog subsystem to perform a number of activities. The following list states some of the activities a typical buyer performs:

- Catalog-based buying
- Create and review RFQs
- Bid on auctions items
- Participate in exchanges
- Work with contracts
- Check orders and order status
- Generate reports

From the list above, we will only look at catalog-based buying process in this section. The other buyer activities are covered in different chapters of this book. Chapter 15, “Example - pricing and contract subsystem” on page 341 describes contract, Chapter 16, “Example - negotiation subsystem” on page 367 describes RFQs, auctions, and exchanges, and Chapter 17, “Example - additional e-Marketplace infrastructure” on page 441 describes orders and reports in detail.

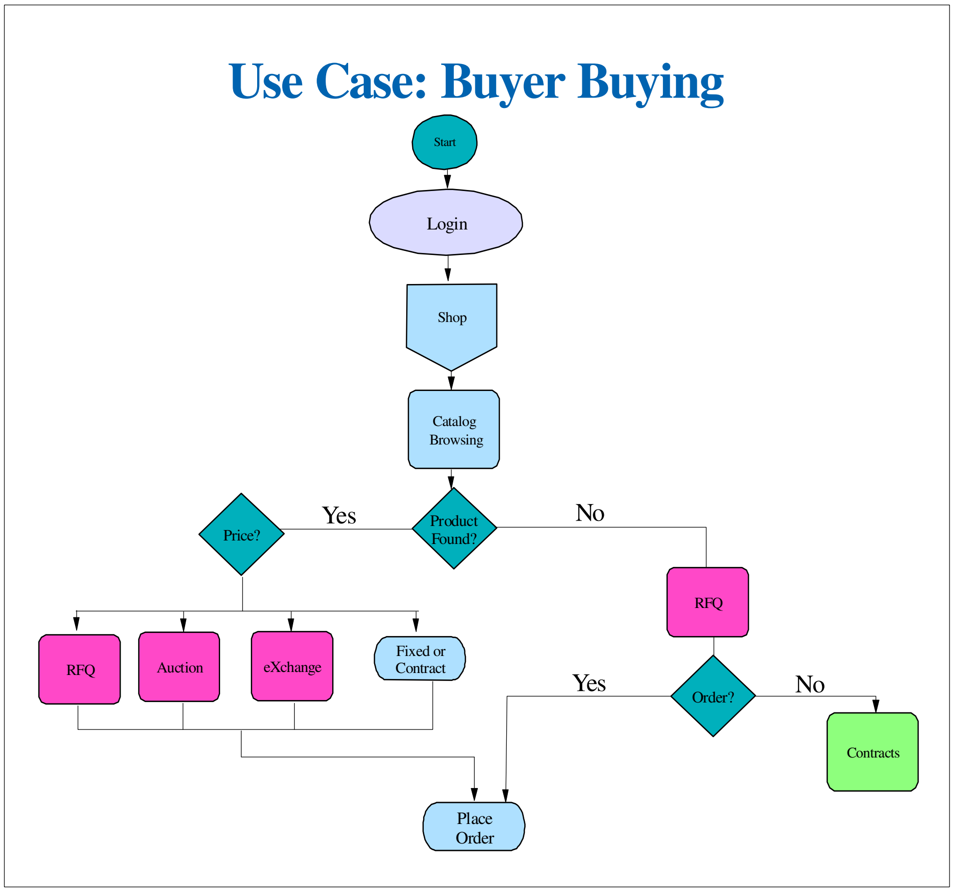


Figure 100. Buyer use cases

Figure 100 shows the buyer use cases. It describes the actions and paths a buyer can take in an e-Marketplace. One of the key activities of a buyer is to purchase products for their organization. Purchasing of products can be performed in many different ways, either via catalog-based buying or through auctions, exchanges or RFQs.

Catalog-based buying is referred to the process of a buyer navigating through the catalog hierarchy, or performing various searches, to find products they need and then initiate the buying process. The main difference between catalog-based buying and buying through auctions and exchanges is that, catalog-based buying uses a shopping cart, whereas auctions and exchanges do not. RFQs also do not use the shopping cart directly but indirectly they can lead into creation of a contract offering, which is part of catalog-based buying and uses the shopping cart.

Catalog-based buying can be categorized into two types:

- Buying from a contract offering
- Buying from a fixed price offering

Both offerings are listed under standard price offerings in the product description window. The actual process of buying, either from a contract or a fixed price offering, is the same. The only difference is that when the standard price offering details are displayed (by clicking the **Prices** link from the product description window) the buyer has the choice of selecting a contract offering or the fixed price offering.

We have already described how a supplier can create a fixed price offering. Let’s briefly look at what needs to take place before a buyer can purchase from a contract offering. A supplier and a buyer can enter into negotiations to establish a contract agreement. Contracts can also be created as a result of an RFQ process. Through either method, when an agreement is reached between the two parties a contract is entered into the e-Marketplace and approved by both parties. The contracts are against a specific product description and therefore an entry is created against the targeted product description. After this process has been completed a buyer can purchase from the contract offering.

We will now take you through the steps of performing catalog-based buying, specifically the buying process through a fixed price offering, since we do not yet have a contract offering created in the e-Marketplace catalog.

14.5.1 Process of catalog-based buying

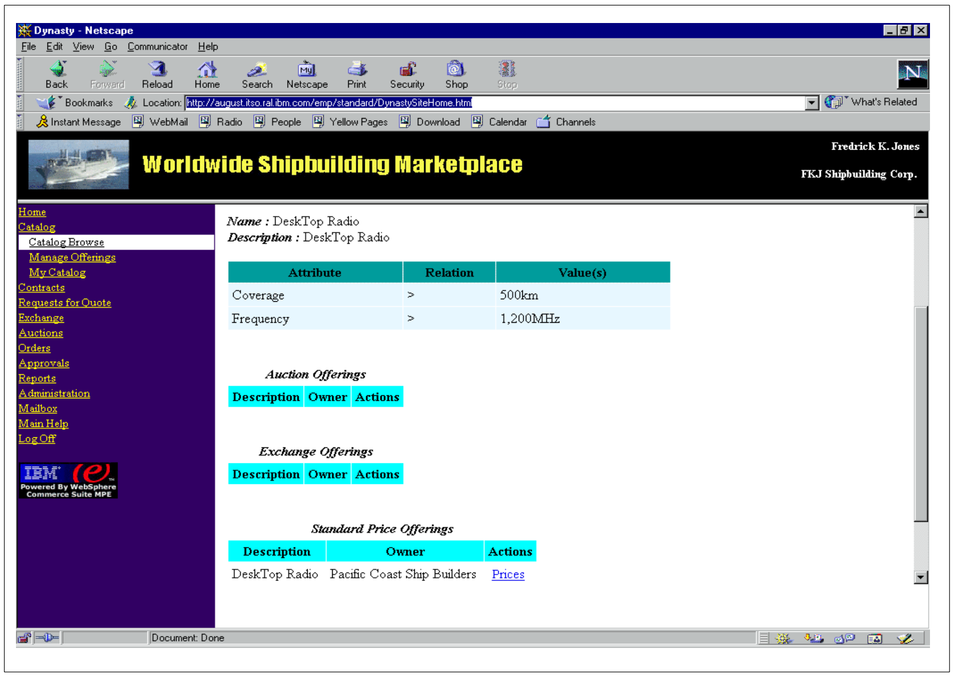
We will first provide a step-by-step approach to buying from the e-Marketplace catalog, then provide some screen captures to visually represent the process.

1. Buyer logs on to the e-Marketplace.
2. Buyer uses the Catalog Browse view to navigate and find the appropriate product.

3. In the product description details window, buyer views the standard price offerings to see if an offering is available.
4. Buyer clicks **Prices** and is presented with a form containing offering details and input fields for quantity required etc.
5. Buyer completes the above form and clicks **Place Order**.
6. A view of the shopping cart with all items in it are displayed.
7. The buyer selects the items for ordering and clicks **Submit Order**.
8. When the order submission is complete the buyer receives an Order Confirmation report.

Now let’s follow these steps and buy five DeskTop Radios.

Figure 101 displays the DeskTop Radio product description and shows that under standard price offerings there is an offering available.


Dynasty - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Shop Stop
Bookmarks Location: http://august.itso.ral.ibm.com/emp/standard/DynastySiteHome.html
What's Related
Instant Message WebMail Radio People Yellow Pages Download Calendar Channels
Worldwide Shipbuilding Marketplace
Fredrick K. Jones
FKJ Shipbuilding Corp.
Home
Catalog
Catalog Browse
Manage Offerings
My Catalog
Contracts
Requests for Quote
Exchange
Auctions
Orders
Approvals
Reports
Administration
Mailbox
Main Help
Log Off
Document: Done

Name : DeskTop Radio
Description : DeskTop Radio

Attribute	Relation	Value(s)
Coverage	>	500km
Frequency	>	1,200MHz

Auction Offerings

Description	Owner	Actions

Exchange Offerings

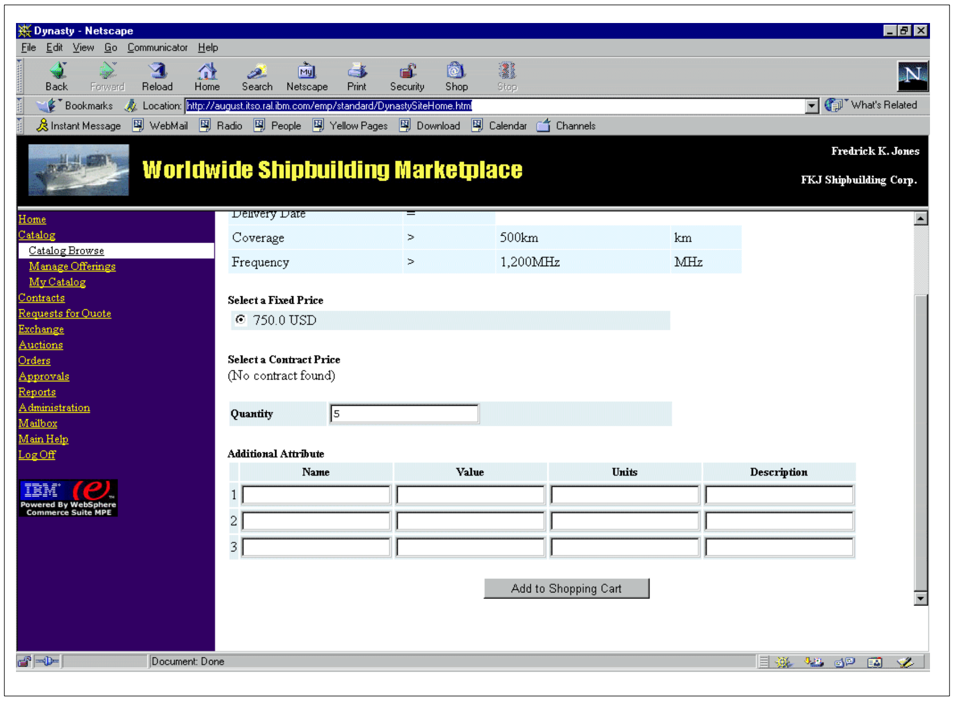
Description	Owner	Actions

Standard Price Offerings

Description	Owner	Actions
DeskTop Radio	Pacific Coast Ship Builders	Prices

Figure 101. Product offering window

The buyer clicks on the **Prices** link to view the details of the offering and select the quantity of the product and add the item to our shopping cart. It is at this step that the buyer can select a contract offering if one is available. Figure 102 shows this window.


Dynasty - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Shop Stop
Bookmarks Location: http://august.itso.ral.ibm.com/emp/standard/DynastySiteHome.html
What's Related
Instant Message WebMail Radio People Yellow Pages Download Calendar Channels
Worldwide Shipbuilding Marketplace
Fredrick K. Jones
FKJ Shipbuilding Corp.
Home
Catalog
Catalog Browse
Manage Offerings
My Catalog
Contracts
Requests for Quote
Exchange
Auctions
Orders
Approvals
Reports
Administration
Mailbox
Main Help
Log Off
IBM Powered By WebSphere Commerce Suite MPE
Add to Shopping Cart
Document: Done

Delivery Date	=		
Coverage	>	500km	km
Frequency	>	1,200MHz	MHz

Select a Fixed Price
(● 750.0 USD)

Select a Contract Price
(No contract found)

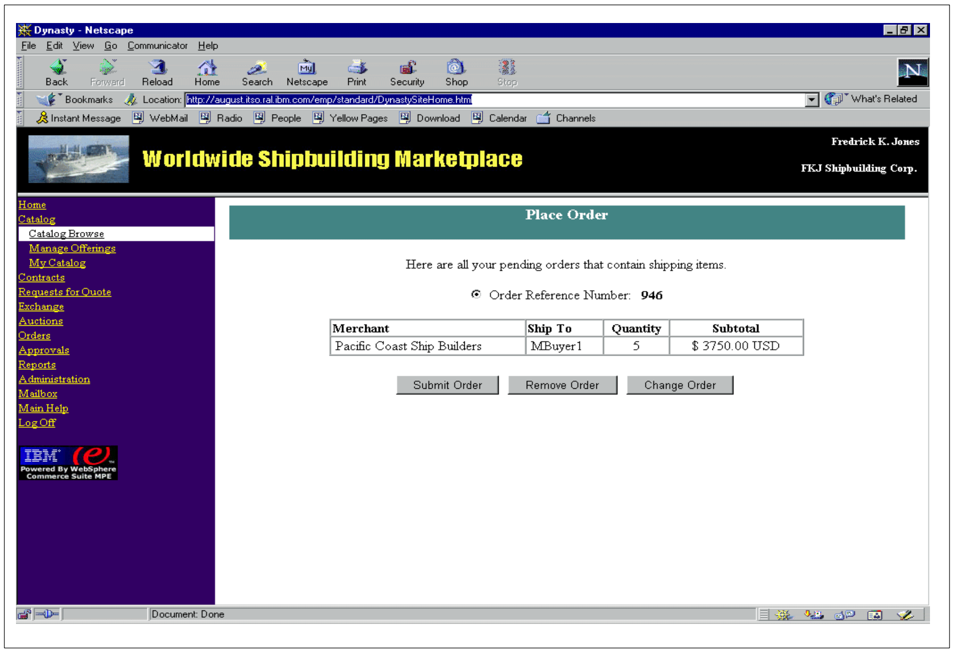
Quantity 5

Additional Attribute

	Name	Value	Units	Description
1				
2				
3				

Figure 102. Detailed product information

Figure 103 on page 338 shows our catalog-based cart and the item that we have selected to purchase. The catalog based cart can contain multiple items from several different selling organizations, but an order can contain items for a unique selling organization for processing reasons. From this window we click **Place Order** to prepare an order.


Dynasty - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Shop Stop
Bookmarks Location: http://august.itso.ral.ibm.com/emp/standard/DynastySiteHome.html
What's Related
Instant Message WebMail Radio People Yellow Pages Download Calendar Channels
Worldwide Shipbuilding Marketplace
Fredrick K. Jones
FKJ Shipbuilding Corp.
Home
Catalog
Catalog Browse
Manage Offerings
My Catalog
Contracts
Requests for Quote
Exchange
Auctions
Orders
Approvals
Reports
Administration
Mailbox
Main Help
Log Off
Powered By WebSphere Commerce Suite MPE
Place Order
Submit Order
Remove Order
Change Order
Document: Done

Here are all your pending orders that contain shipping items.

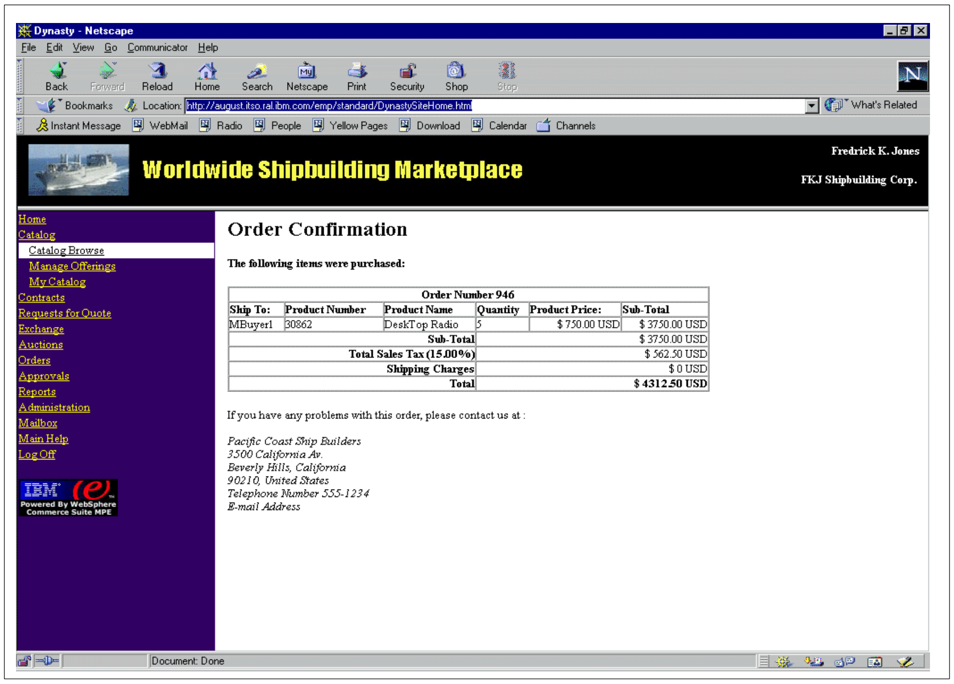
(●) Order Reference Number: **946**

Merchant	Ship To	Quantity	Subtotal
Pacific Coast Ship Builders	MBuyer1	5	$ 3750.00 USD

Figure 103. Place order

Figure 103 on page 338 shows the window from where we can Submit Order and complete the catalog-based buying process.

Figure 104 on page 339 shows the order confirmation information.


Dynasty - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Shop Stop
Bookmarks Location: http://august.itso.ral.ibm.com/emp/standard/DynastySiteHome.html What's Related
Instant Message WebMail Radio People Yellow Pages Download Calendar Channels
Worldwide Shipbuilding Marketplace
Fredrick K. Jones
FKJ Shipbuilding Corp.
Home
Catalog
Catalog Browse
Manage Offerings
My Catalog
Contracts
Requests for Quote
Exchange
Auctions
Orders
Approvals
Reports
Administration
Mailbox
Main Help
Log Off
IBM Powered By WebSphere Commerce Suite MPE
Document: Done

Order Confirmation

The following items were purchased:

Order Number 946					
Ship To:	**Product Number**	**Product Name**	**Quantity**	**Product Price:**	**Sub-Total**
MBuyer1	30862	DeskTop Radio	5	$ 750.00 USD	$ 3750.00 USD
			Sub-Total		$ 3750.00 USD
			Total Sales Tax (15.00%)		$ 562.50 USD
			Shipping Charges		$ 0 USD
			Total		**$ 4312.50 USD**

If you have any problems with this order, please contact us at :

Pacific Coast Ship Builders
3500 California Av.
Beverly Hills, California
90210, United States
Telephone Number 555-1234
E-mail Address

Figure 104. Order confirmation window

14.6 Interaction with other subsystems

The catalog subsystem interacts with almost all the other subsystems available in the e-Marketplace. These interactions take place using function calls between subsystems and objects shared by all subsystems in the database.

We will describe, on a functional level, how the WebSphere Commerce Suite, Marketplace Edition for AIX subsystems interacts with the catalog subsystem. The following subsystems are identified in the WebSphere Commerce Suite, Marketplace Edition for AIX:

- Pricing and Contract

  The catalog subsystem interacts with the pricing and contract subsystem to establish and manage fixed price and contract offerings.

- Negotiation

  The catalog subsystem interacts with the negotiation subsystem to facilitate creation of auctions, RFQs and exchanges.

- Membership and Registration

  The catalog subsystem has no direct interaction with the membership and registration subsystem.

- Access Control

  The catalog subsystem interacts with access control subsystem to validate access authority of uses to perform catalog-specific operations.

- Approval Flow

  The catalog subsystem interacts with the approval flow subsystem to establish an approval process for organizations requiring such processes for their members. An example of such an approval process would be the need to approve offerings created by the suppliers.

- Orders

  The catalog subsystem interacts with the order subsystem to provide a mechanism for buyers and suppliers to view orders submitted by their organization or other participating organizations.

- Hub Business

  The catalog subsystem does not interact with the hub business administrative operations.

- Report

  The catalog subsystem does not have a direct interaction with the report subsystem. However ,the report subsystem uses information that the catalog subsystem stores in the database to prepare the predefined e-Marketplace reports.

Chapter 15. Example - pricing and contract subsystem

In the previous chapter we covered the catalog subsystem and provided you with information and examples of how to create and manage the e-Marketplace catalog. We covered the supplier and buyer interaction with the catalog subsystem and provided examples of how to create and manage fixed price offerings or how to buy from an existing fixed price offering using the catalog-based buying process. We also noted that other than catalog based buying there are other methods through which transactions take place in an e-Marketplace. Methods such as:

- Contracts
- Auctions
- RFQs
- Exchanges

Auctions, RFQs and exchanges are considered part of the negotiation subsystem. We will cover them in Chapter 16, “Example - negotiation subsystem” on page 367.

Contracts are part of the pricing contract subsystem. In this chapter we will provide information and examples of how contracts work.

15.1 Contracts

A contract is a mutual agreement between a buying and a selling organization in the e-Marketplace. When a contract agreement is reached and entered into the system, the contract offering becomes part of standard price offerings. Buyers from the participating organization can use catalog-based buying to make purchases against the contract. For details on catalog-based buying please see Chapter 14, “Example - catalog subsystem” on page 269.

In this section we will cover the following topics:

- Contract high-level overview.
- Contract low-level design.
- Example: supplier interaction
- Example: buyer interaction
- Contract interaction with other subsystems

In this chapter we use several screen captures to provide visual representation for some of the topics we are covering. We use the “Worldwide Shipbuilding Marketplace” example provided with the default installation of the WebSphere Commerce Suite, Marketplace Edition for AIX.

15.1.1 Contract high-level overview

In the current implementation of the WebSphere Commerce Suite, Marketplace Edition for AIX the contract subsystem is limited to pricing contracts. This means contracts can only be established to extend a special price or a discount to a participating organization.

Contracts provide value to both selling organizations and buying organizations. The buying organization benefits from having a special price or discount for a product that they need to obtain. They also benefit from having secured a fixed quantity of a certain product. The selling organization benefits from contract agreements by knowing that the buying organization has an obligation to buy a fixed minimum number of products or spend a fixed minimum amount. In other words, they have secured sales for a predefined quantity and amount.

15.1.2 Contract low-level design

In this section we look at the low-level design of the contract subsystem. We list the code level components of the contract subsystem and give a brief explanation of their functionality. A detailed discussion of these components is outside the scope of this redbook. For details, we recommend the online documentation and the JavaDocs specified for each component.

15.1.2.1 Design principles

The contract subsystem is designed to provide the components necessary for the process of creating contracts. The contract subsystem is built based on the Java programming model of WebSphere Commerce Suite, Marketplace Edition for AIX, discussed in 10.5, “Marketplace Edition programming model” on page 209.

We explain the interaction controllers, commands, JSPs, objects and the database tables of the contract subsystem later in this chapter.

To successfully create a contract in the e-Marketplace, a contract must go through a predefined process and change several states before becoming an active contract. Figure 105 shows the process and state of a contract from submission to creation. On a high level the process of creating a contract is a two stage process. In the first stage, the creator initiates the process by filling out a new contract creation form and submitting it. In the second stage an authorized person from the participating organization reviews the submission and agrees to the terms and conditions. When the agreement in the second stage is recorded in the system the contract creation lifecycle is complete. When a successful contract creation process is complete, no modifications can be made to the details of a contract. However, with mutual agreement a contract can be terminated by the market administrator.

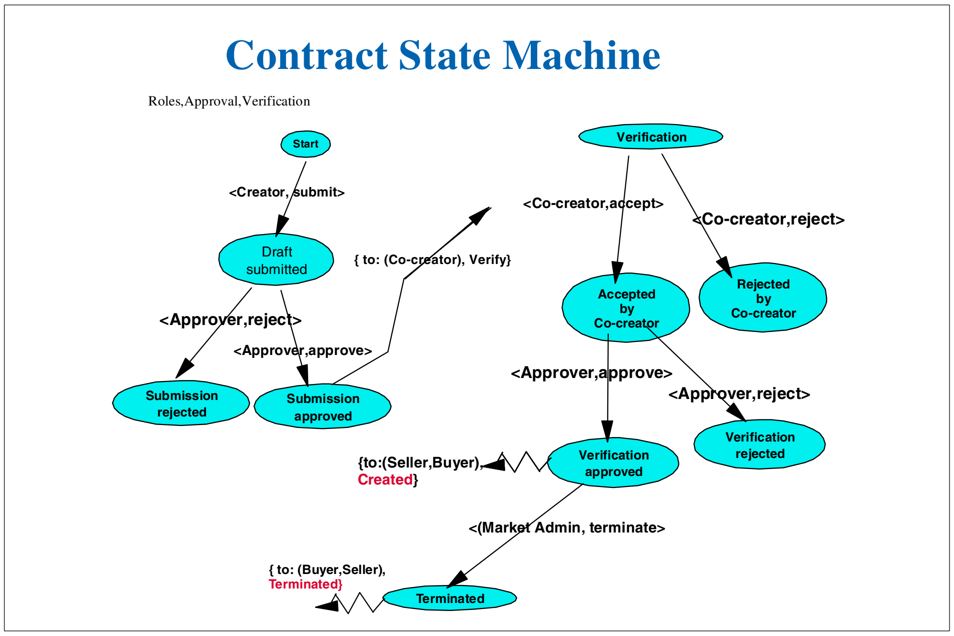


Figure 105. Contract state machine

In the WebSphere Commerce Suite, Marketplace Edition for AIX implementation of an e-Marketplace, contracts can be initiated by either a buyer or a seller. Buyers are allowed to initiate the creation of a contract from a winning RFQ. Suppliers can initiate the creation of a contract for products or categories where a fixed price offering already exists.

When a contract is entered in the system, the WebSphere Commerce Suite, Marketplace Edition for AIX uses some of the fields from the CONTRACT and CONTITEM tables to provide the buyers an up-to-date view of the contract in terms of monetary and quantity obligations and rights. *Obligations* are the minimum monetary amount or quantity of products that a buyer is committed to buy against a given contract. *Rights* are maximum monetary amount or quantity of products that a buyer is allowed to buy against a given contract.

The WebSphere Commerce Suite, Marketplace Edition for AIX also determines if a contract is still valid or if it is expired. A contract is considered expired when any one of the following conditions occur:

- End time is reached (requires that an end time was entered).
- Maximum monetary volume is reached.
- Maximum quantity is reached.

No transactions are allowed against an expired contract.

15.1.2.2 Use cases

The contract subsystem is used in two ways:

- To create new contracts
- To be updated by catalog-based buying.

Create new contracts

Buyers and sellers with Contract Administrator authority can create contracts in the e-Marketplace. In order to create a contract offering for a product description, a fix price offering must exist for the product. The reason for this is that if, for example, the supplier wishes to extend a discount for a given product, then a fixed price must exist in order to calculate the discounted price.

In here we explain the process of creating a contract begins when the contract subsystem interaction controller ContractPrepare receives a request to create a new contract and instantiates the ContractBean and passes it to the ContractPrepareCmd. This command uses the createcontractfirststep JSP to obtain information for the other participating organization and populates the bean. Then contract and item-specific information are populated in the ContractBean through use of the ContractCreate interaction controller, the ContractCreateCmd command and the contractdetail and contractitems JSPs. When all the information required is captured, a contract entry is created in the e-Marketplace system with a status of Submission Approved through the ContractSubmit interaction controller and the ContractSubmitCmd command.

At this point the contract administrator of the second participating organization begins the process of verification and approval. A request is made to the ContractVerify interaction controller. This interaction controller uses the ContractSearchCmd command and the searchContract JSP to prompt the user to perform a search of all the contracts that need to be verified. The result are displayed through VerifyContractList JSP, which allows the invocation of the ContractVerifyCmd command to accept the contract. When a contract is accepted the status changes from Submission Approved to Verification Approved.

A contract can also be created by a buyer from a winning RFQ. If the buyer chooses to submit a winning quote to be converted to a contract, a call is placed to the CreateContractFromRfqCmd. This command calls the ContractCreate interaction controller and passes information from the RFQ to populate the ContractBean. The rest of the process is similar to the one mentioned above.

Catalog-based buying

Catalog-based buying uses the contract subsystem and its information to perform the following tasks:

- Determine and use contract price or discount.
- Track the monetary volume and product quantity numbers when an order is placed or rejected.

The ContractGetProductContractCmd and ContractUndoTrackingCmd commands are used for this purpose.

15.1.2.3 Code Level Components

Interaction controllers

The contract subsystem interaction controllers parse the input parameters and forward requests to the WebSphere Commerce Suite, Marketplace Edition for AIX command manager. The interaction controllers also execute commands that are passed back by the command factory.

The following lists the contract subsystem interaction controllers for WebSphere Commerce Suite, Marketplace Edition for AIX:

- ContractCreate

  Used by a contract administrator to create a draft contract.
- ContractSubmit

  Used by a contract administrator to submit a drafted contract.
- ContractList

  Used by authorized users to generate a list of contracts.
- ContractView

  Used by authorized users to view the details of a specific contract.
- ContractVerify

Used by a contract administrator to accept or reject a contract drafted by the counterparty

- ContractTerminate

  Used by the hub administrator to terminate a contract representing two parties.
- ContractUndoTrack

  Undo operation of tracking the contract; used by the order subsystem.
- ContractSubmit

  Used by contract administrator to submit a request for contract creation.
- ContractPrepare

  List all organizations in Hub for contract creation.
- ContractGetCategoryContract

  Called by the catalog to show the contracts on a specific category.
- ContractGetProductContract

  Called by catalog-based buying to select a contract for purchasing.
- ContractReport

  Used by authorized users to view a contract report.
- ContractSearch

  Used by authorized users to search contracts.

More information on contract subsystem interaction controllers is available in the online documentation of WebSphere Commerce Suite, Marketplace Edition for AIX and com.ibm.commerce.emp.contract.icontrollers JavaDoc.

Commands

Commands are the component of the contract subsystem that carry out the execution of the business logic and ultimately address the requests of the users. Commands interact with almost all other components of the contract subsystem, from interaction controllers to JSPs and beans.

The following are the commands in the contract subsystem of the WebSphere Commerce Suite, Marketplace Edition for AIX:

- ContractSubmitCmd

  Create a draft contract for a user’s own organization’s approval.
- ContractVerifyCmd

  Accept or reject a draft contract.

- ContractCreateCmd

  Display a creation form for a user to input contract terms and conditions.
- ContractUndoTrackCmd

  Recover the contract if the order based on the contract is cancelled.
- ContractPrepareCmd

  List organizations to select a counterparty.
- ContractGetCategoryContractCmd

  Get all contracts on a specific category.
- ContractGetProductContractCmd

  List all contracts applicable to a specific product.
- ContractListCmd

  List all contracts based on organization and user.
- ContractSearchCmd

  Display a search form.
- ContractOrgReportCmd

  Generate a report for a specific organization.
- ContractProdReportCmd

  Generate a report for a specific product or category.
- ContractDetailReportCmd

  Generate a report for a specific counterparty and product.
- ContractTerminateCmd

  Terminate a contract.
- ContractViewCmd

  Display the detaiedl information of a contract.

More information on contract subsystem commands is available in the online documentation of the WebSphere Commerce Suite, Marketplace Edition for AIX and com.ibm.commerce.emp.contract.commands.interfaces and com.ibm.commerce.emp.contract.commands JavaDocs.

JSP pages

A JavaServer Page (JSP) is server-side Java code that facilitates the creation and display of dynamic content onto Web pages. JSP technology separates the creation of dynamic content from static HTML content. This feature allows the Web page designers to modify the design and the static content of Web pages without having to worry about the dynamic content. All they have to do is place the dynamic content on the page they are designing. For more information on JSPs and the latest specifications, please visit the`http://java.sun.com/products/jsp/` Web site.

The contract subsystem contains the following JSPs. It is important to note that not all JSPs render a Web page. Some are used to interact with JavaBeans and extract and pass information to other components of the contract subsystem.

- createcontractfirststep

  Displays the page to select another party.
- contractdetails

  Displays the contract creation form.
- confirmcontract

  Displays drafted contract to be confirmed.
- submitcontractsuccess

  Displays a success message to the user if submission is successful.
- createcontractdiscard

  Displays a message to confirm discarding a contract.
- contractitems

  Displays the add items to contract page.
- category

  Displays a child category list to be selected and added to contract items.
- category2

  Displays a product list to be selected and added to contract items.
- categoryTOC

  Displays a table of contents of a catalog for adding contract items.
- ListContract

  Lists all the contracts that a user is authorized to view.
- ViewContract

  Displays the details of a contract selected by a user.
- VerifyContractList

  Lists the contracts to be verified by a user and allows verification.

- VerifyContractSuccess

  A status message to show a user that the contract verification was successful.
- TerminateContract

  Displays a form for the user to identify the contract that needs to be terminated.
- TerminatedOver

  Displays a status message to verify operation success.
- reportContract

  Displays a form to select a type of report to be generated.
- reportperiod

  Displays a form to provide a time period for a report.
- searchContract

  Display a form to input search conditions for contracts.
- error

  Displays an error message in case of a failure.

For more information on JSPs please see the online documentation.

Objects
The WebSphere Commerce Suite, Marketplace Edition for AIX uses the following beans:

- ContractBean

  Contains the properties of a contract.
- ContractDisplayBean

  Contains the displaying properties of a contract.
- ContractItemBean

  Contains the properties of a specified contract item.
- ContractItemDisplayBean

  Contains the displaying properties of a specified contract item.
- ContractListBean

  Contains the properties of a list of contracts.
- InputErrorBean

Contains the error messages.

For detailed information on these beans please see the online documentation and the JavaDoc on com.ibm.commerce.emp.ccontract.db and the com.ibm.commerce.emp.contract.beans.

Figure 106 on page 350 depicts the class diagrams and interfaces of the contract subsystem.

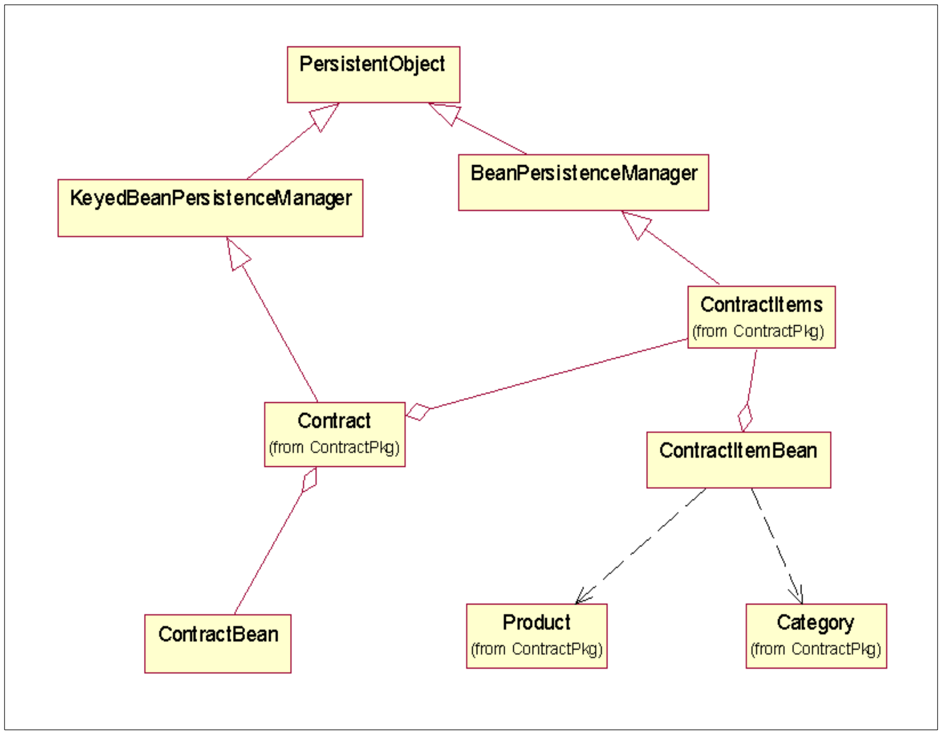


Figure 106. Class diagram and interfaces of the contract subsystem

Database tables

In this section we are including the additions to the WebSphere Commerce Suite 4.1 database schema with respect to the contract subsystem.

CONTRACT (New): This table stores information about each contract.

Table 32. CONTRACT

Column Name	Data Type	Description
cntrefn	integer not null	Contract reference number. This is the primary key.
cntbuyrefn	integer not null	Buying Organization Reference Number. This is a foreign key that references the MERFNBR column in the MERCHANT table.
cntsellrefn	integer not null	Selling Organization Reference Number. This is a foreign key references the MERFNBR column in the MERCHANT table.
cntstatus	char(4) not null	Contract Status: DRFT - Editable draft. SBMT - Submitted for approval. SAPP - Submission approved. SRJT - Submission rejected. VRFA - Accepted by verification. VRFR - Rejected by verification. VAPP - Verification approved. VRJT - Verification rejected. TMNT - Terminated .
cntdiscount	num(3,2)	A percent discount that applies to all purchases under the contract. This is set to 1 (that is 100%) if no discount applies.
cntcoverage	char(4) not null	What purchases the contract covers: ALL - All purchases between the buyer and the seller. The discount is specified in by the cntdiscount field. SPEC - Specific products or categories specified in the cntitem table.
cntbegintim	timestamp not null	Time at which the contract is scheduled to start.
cntendtim	timestamp	Time at which the contract is scheduled to end.
cntcrttim	timestamp	Date created.

cntcurrstr	char(3) not null	Currency for monetary value.
cntminval	num (15,2)	The buyer is required to make purchases at the contracted price(s) until the sum of the monetary amounts are at least this value. NULL if none is required.
cntmaxval	num (15,2)	The seller is required to accept the agreed price(s) until the order's monetary values sum to this amount. NULL if none is required.
cntpenadesc	char(254)	Specifies penalties for failure to perform.
cntdetailfile	char(254)	File name and path of file specifying additional contract details.
cntcreator	integer	Contract creator.
cntcountercreator	integer	Contract creator of counterparty.
cntbuysign	integer	Contract signee of buying organization.
cntsellsign	integer	Contract signee of selling organization.
cntcrtmethod	char(4)	The method used for creating the contract specifications. Initially, MAN (Manual creation) and RFQ (Request for Quote) will be supported.
cntcrtkey	integer	Foreign key link to the creating processes ID. For instance, for an RFQ, this field is the ID of the RFQ from which the contract was created.
cntsumval	num(15,2) not null	Sum of the monetary value of a purchase placed under contract. This is updated upon order submission.
cntfield1	num(15,2)	Merchant customization field.
cntfield2	integer	Merchant customization field.
cntfield3	varchar(254)	Merchant customization field.
cntfield4	varchar(254)	Merchant customization field.
cntversdesc	char(254)	For future use. A version description for the contract.

CNTITEM (New): This table stores contract item (product) information about each contract.

Table 33. CNTITEM

Column Name	Data Type	Description
cicntrefn	integer not null	Contract reference number. This is a foreign key that references the CNTREFN column in the CONTRACT table.

citype	char(1) not null	Type of contract item:
		P = product
		C = category

ciprrefn	integer	Product reference number. This is a foreign key that references the PRRFNBR column in the PRODUCT table. This field is NULL when the item is a category.
cicgrfnbr	integer	Product reference number. This is a foreign key that references the CGRFNBR column in the CATEGORY table. This field is NULL when the item is a product.
cipricetype	CHAR(5) not null	Type of contracted price, either DISC for discount or PRICE for a fixed price.
ciprice	num(15,2)	Contract price.
cicurrstr	char(3)	Currency for category monetary value if citype is category, or currency of product price if citype is product.
cisumquant	num(15,2) not null	The number of items that have been bought. This is only used for contract items that refer to a product.
ciminquant	num(15,2)	The buyer is required to make purchases whose quantities sum to at least this value. NULL if none required. This is only used for contract items that refer to a product.

cimaxquant	num(15,2)	The seller is required to accept the agreed price until the buyer’s orders have quantities summing to is value. NULL if no requirement. This is only used for contract items that refer to a product.
cisumval	num(15,2) not null	Sum of the monetary value of a purchase placed under contract for the category. This is updated upon order submission. This is only used for contract items that refer to a category.
ciminval	num (15,2)	The buyer is required to make purchases at the contracted price(s) until the sum of the monetary amounts are at least this value. NULL if none is required. This is only used for contract items that refer to a category.
cimaxval	num (15,2)	The seller is required to accept the agreed price(s) until the orders’ monetary values sum to this amount. NULL if none is required. This is only used for contract items that refer to a category.
cifield1	num(15,2)	Merchant customization field.
cifield2	integer	Merchant customization field.
cifield3	varchar(254)	Merchant customization field.

15.1.3 Example: supplier interaction

A supplier’s interaction to the contract subsystem is limited to starting the process of contract creation. In this section we first describe the steps involved in this process and then through an example and screen captures we demonstrate the process.

In order to create a contract, a supplier must have a Contract Administrator role, and a fixed price offering must exist for the product the supplier intends to create a contract offering for. If the above conditions are met then a supplier can create a contract by following these steps:

1. Supplier logs on o the e-Marketplace.
2. Supplier clicks **Contracts** in the Navigation pane and then clicks **New Contract**.

3. From the next window the supplier selects the trading partner and their role in this contract and clicks the **Create a New Contract** button.
4. The supplier fills in all the relevant information In the Edit Contract (New) form and clicks **Manage Item**.
5. The list of items currently in the contract is displayed. This is an empty window the first time a contract is created.
6. Supplier clicks **Add Items from Interest List**.
7. The Contract Interest List is displayed. If the desired items are in the list the supplier can select and add them to the contract. If the item desired is not on the list, supplier can click **Browse Catalog** to navigate and find the product.
8. When the desired product is selected, the supplier clicks the **Add Items to Contract** button.
9. Supplier is returned to the list of current items in the contract. The supplier specifies the detail product pricing or discount as well as product quantity or value obligations and rights. When the form is complete the supplier clicks the **Save and Back** button.
10. Supplier is returned to the Edit Contract (New) form. When all the entries are made the supplier clicks **Submit** to complete the process.
11. A successful completion window is displayed and a contract reference number is provided to the supplier.

In order to complete the creation of the contract ,the buying organization must verify and accept the submitted contract. We will cover this in the 15.1.4, “Example: buyer interaction” on page 361.

Before we proceed to the next example let’s look at some of the contract-specific fields presented to the supplier during the contract creation process.

- **Buying Organization** is the name of the organization with which a supplier organization is creating the contract with.
- **Seller Organization** is the name of the supplier organization.
- **Creator** is the name of the creator of the contract.
- **Co-creator** is the email id of the person in the counterparty organization. This is the person who needs to agree to the terms and conditions of the contract.

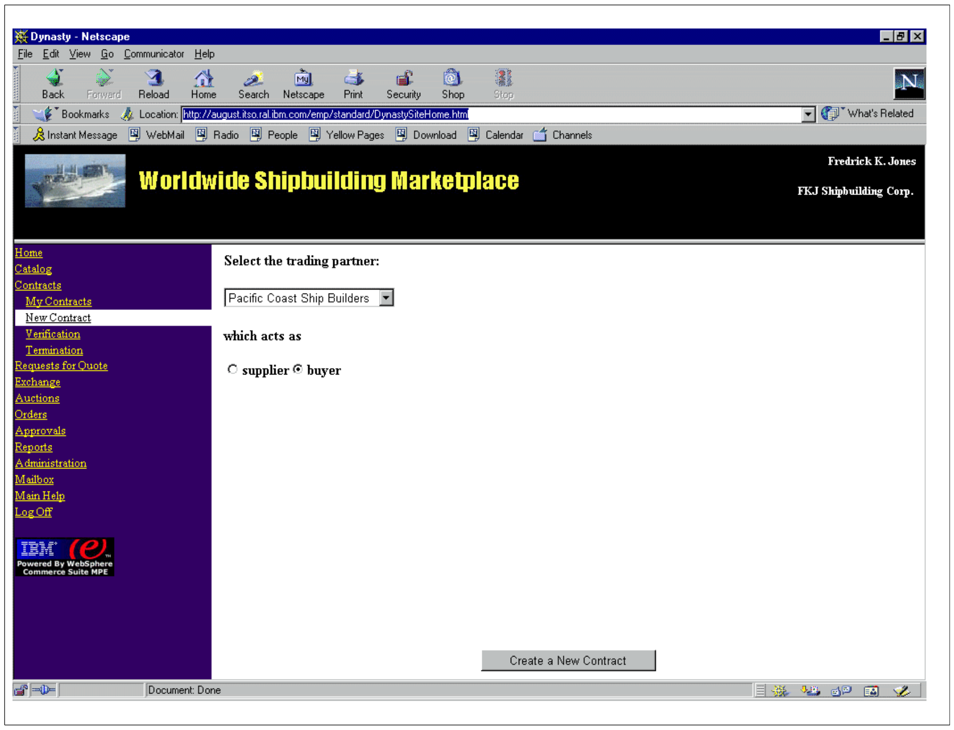
- **Buying Signee** and **Selling Signee** are the members from both organizations who need to approve the contract. They may be the same as thecreator and co-creator.
- **Begin Time** and **End Time** establishes a time period during which the contract is valid.
- **Minimum Monetary Volume** is the obligation, in monetary volume, made by the buyer towards the entire contract. This value is not a product-specific value. It is the overall obligation value, and can cover many products that are included in the contract.
- **Maximum Monetary Volume** is the right given to the buying organization to buy up to the maximum monetary volume amount. This value is not a product-specific value. It is the overall maximum value, and can cover many products that are included in the contract.
- **Coverage** identifies if the contract is for a specific product or for all products**.**
- **Discount** applies only if the coverage is set to all products.

On the item-specific level the following fields contain key information:

- **Minimum Item Quantity/Value** is the obligation, in number of items or value, made by the buyer towards a specific product in the contract.
- **Maximum Item Quantity/Value** is the right given to the buying organization to buy up to the maximum quantity or value of the specific product.
- **Contract Price or Discount** specifies the special price or discount being offered for the specific product.

In the following few pages we will create a new contract for “Light Duty Propellers”. In this example the “Maritime Ship Building Co.” plays the supplier role and extends a contract to the buying “Pacific Coast Ship Builders” organization. A fixed price offering for $1500.00 exists and the agreed contract value is $1250.00.

Figure 107 on page 357 shows the window where the participating buying organization is selected.


Dynasty - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Shop Stop
Bookmarks Location: http://august.itso.ral.ibm.com/emp/standard/DynastySiteHome.html
What's Related
Instant Message WebMail Radio People Yellow Pages Download Calendar Channels
Worldwide Shipbuilding Marketplace
Fredrick K. Jones
FKJ Shipbuilding Corp.
Home
Catalog
Contracts
My Contracts
New Contract
Verification
Termination
Requests for Quote
Exchange
Auctions
Orders
Approvals
Reports
Administration
Mailbox
Main Help
Log Off
Powered By WebSphere Commerce Suite MPE
Create a New Contract
Document: Done

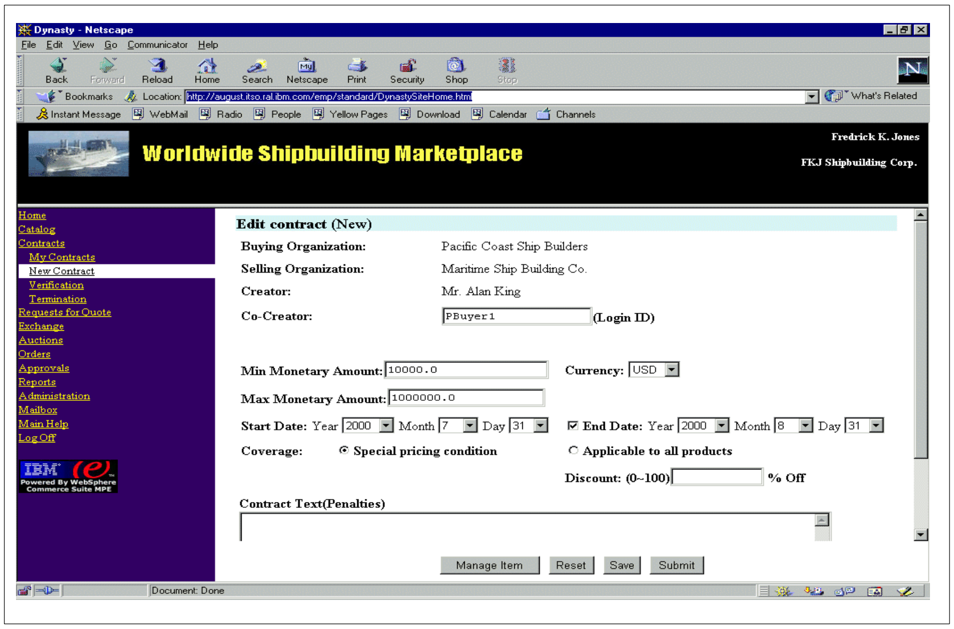
Select the trading partner: Pacific Coast Ship Builders

which acts as

supplier [✓ buyer]

Figure 107. Participating organization

Figure 108 on page 358 shows the new contract window. We have filled in the required fields. From this window we click **Manage Item** to add a product item to the contract. The next window is an empty since there are currently no items added to the contract. From this window we select **Add Items from Interest List**. The contract interest list is shown in Figure 109 on page 358, and from it we select the **Light Duty Propeller** and click **Add Items to Contract**.


Dynasty - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Shop Stop
Bookmarks Location: http://august.itso.ral.ibm.com/emp/standard/DynastySiteHome.html What's Related
Instant Message WebMail Radio People Yellow Pages Download Calendar Channels
Worldwide Shipbuilding Marketplace
Fredrick K. Jones
FKJ Shipbuilding Corp.
Home
Catalog
Contracts
My Contracts
New Contract
Verification
Termination
Requests for Quote
Exchange
Auctions
Orders
Approvals
Reports
Administration
Mailbox
Main Help
Log Off
Manage Item
Reset
Save
Submit
Document: Done

Edit contract (New)

Buying Organization:	Pacific Coast Ship Builders
Selling Organization:	Maritime Ship Building Co.
Creator:	Mr. Alan King
Co-Creator:	PBuyer1 (Login ID)

Min Monetary Amount: 10000.0 **Currency:** USD

Max Monetary Amount: 1000000.0

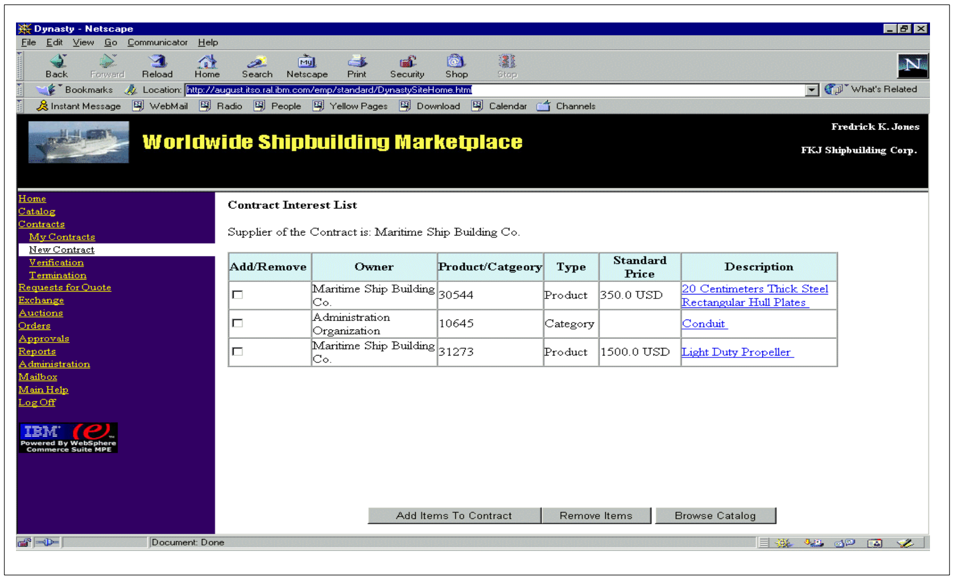
Start Date: Year 2000 Month 7 Day 31 ✓ **End Date:** Year 2000 Month 8 Day 31

Coverage: ● **Special pricing condition** ○ **Applicable to all products**

Discount: (0–100) % Off

Contract Text(Penalties)

Figure 108. Edit contract


Dynasty - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Shop Stop
Bookmarks Location: http://august.itso.ral.ibm.com/emp/standard/DynastySiteHome.html What's Related
Instant Message WebMail Radio People Yellow Pages Download Calendar Channels
Worldwide Shipbuilding Marketplace
Fredrick K. Jones
FKJ Shipbuilding Corp.
Home
Catalog
Contracts
My Contracts
New Contract
Verification
Termination
Requests for Quote
Exchange
Auctions
Orders
Approvals
Reports
Administration
Mailbox
Main Help
Log Off
Add Items To Contract
Remove Items
Browse Catalog
Document: Done

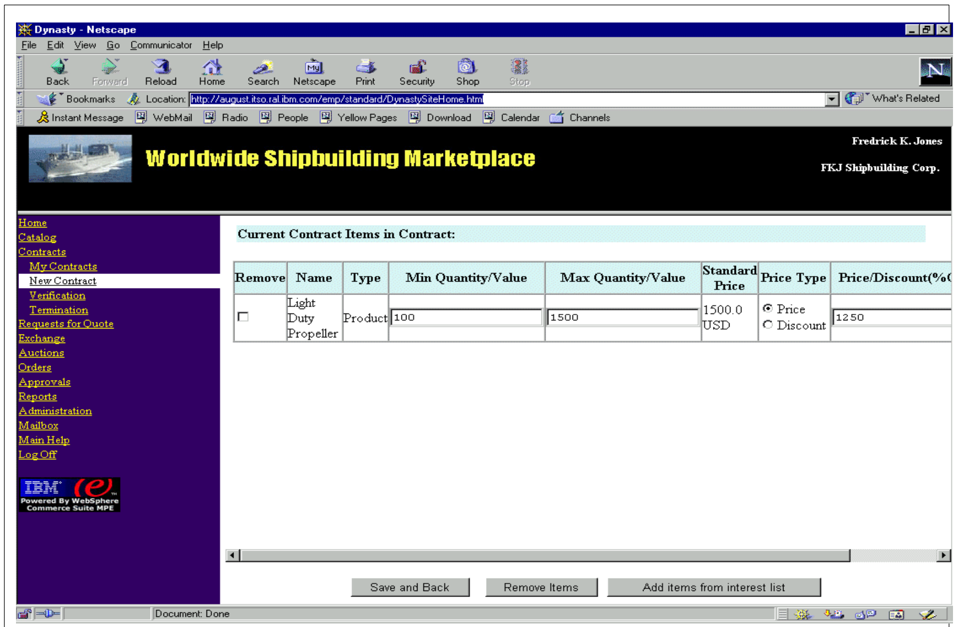
Contract Interest List

Supplier of the Contract is: Maritime Ship Building Co.

Add/Remove	Owner	Product/Catgeory	Type	Standard Price	Description
☐	Maritime Ship Building Co.	30544	Product	350.0 USD	20 Centimeters Thick Steel Rectangular Hull Plates
☐	Administration Organization	10645	Category		Conduit
☐	Maritime Ship Building Co.	31273	Product	1500.0 USD	Light Duty Propeller

Figure 109. Contract interest list

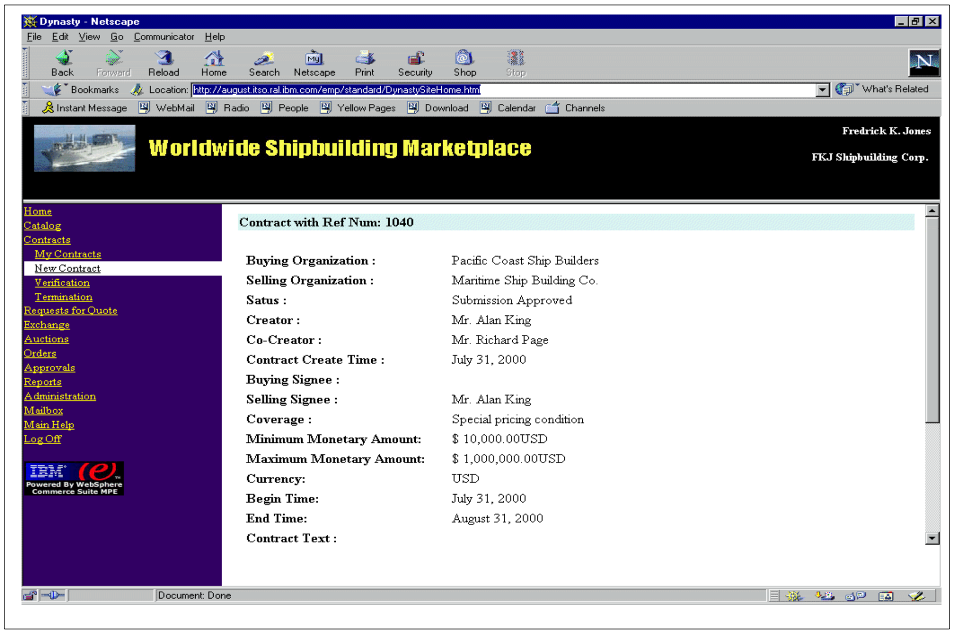
Figure 110 on page 359 shows the item we just added to the contract. In this window we provide item-specific information for the contract. When completed we click **Save and Back** to return to the Edit Contract window. At this stage the contract is ready for submission. We click **Submit** to complete the supplier-side process. Figure 111 on page 360 and Figure 112 on page 361 together show the confirmation and details of the contract we just created.


Dynasty - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Shop Stop
Bookmarks Location: http://august.itso.ral.ibm.com/emp/standard/DynastySiteHome.htm
What's Related
Instant Message WebMail Radio People Yellow Pages Download Calendar Channels
Worldwide Shipbuilding Marketplace
Fredrick K. Jones
FKJ Shipbuilding Corp.
Home
Catalog
Contracts
My Contracts
New Contract
Verification
Termination
Requests for Quote
Exchange
Auctions
Orders
Approvals
Reports
Administration
Mailbox
Main Help
Log Off
IBM Powered By WebSphere Commerce Suite MPE
Save and Back
Remove Items
Add items from interest list
Document: Done

Current Contract Items in Contract:

Remove	Name	Type	Min Quantity/Value	Max Quantity/Value	Standard Price	Price Type	Price/Discount(%
☐	Light Duty Propeller	Product	100	1500	1500.0 USD	[✓ Price] Discount	1250

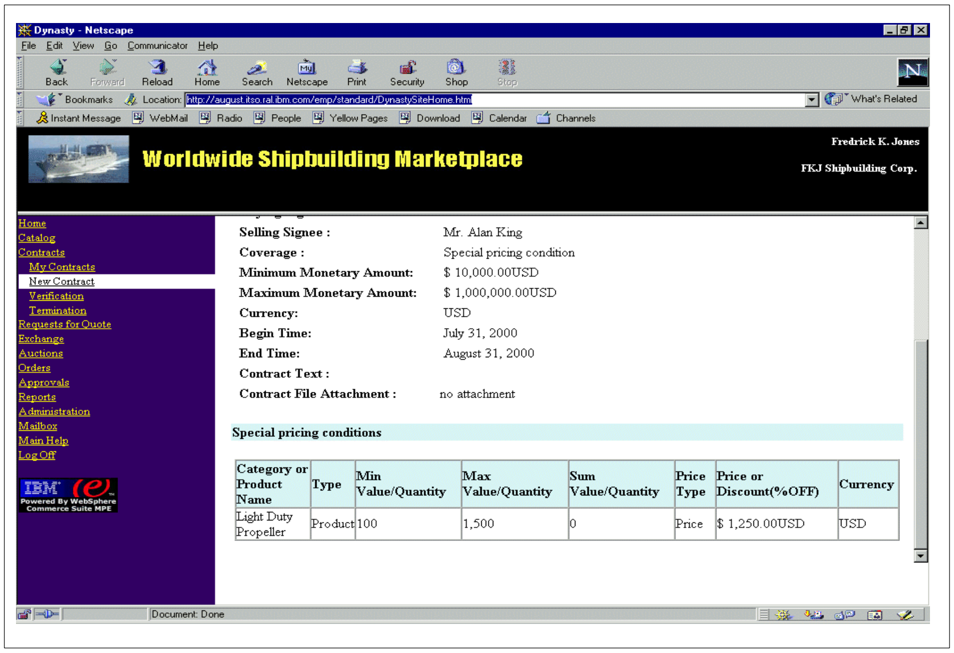
Figure 110. Contract current items


Dynasty - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Shop Stop
Bookmarks Location: http://august.itso.ral.ibm.com/emp/standard/DynastySiteHome.html What's Related
Instant Message WebMail Radio People Yellow Pages Download Calendar Channels
Worldwide Shipbuilding Marketplace
Fredrick K. Jones
FKJ Shipbuilding Corp.
Home
Catalog
Contracts
My Contracts
New Contract
Verification
Termination
Requests for Quote
Exchange
Auctions
Orders
Approvals
Reports
Administration
Mailbox
Main Help
Log Off
Document: Done

Contract with Ref Num: 1040

Buying Organization :	Pacific Coast Ship Builders
Selling Organization :	Maritime Ship Building Co.
Satus :	Submission Approved
Creator :	Mr. Alan King
Co-Creator :	Mr. Richard Page
Contract Create Time :	July 31, 2000
Buying Signee :	
Selling Signee :	Mr. Alan King
Coverage :	Special pricing condition
Minimum Monetary Amount:	$ 10,000.00USD
Maximum Monetary Amount:	$ 1,000,000.00USD
Currency:	USD
Begin Time:	July 31, 2000
End Time:	August 31, 2000
Contract Text :	

Figure 111. Confirmation window: top portion


Dynasty - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Shop Stop
Bookmarks Location: http://august.itso.ral.ibm.com/emp/standard/DynastySiteHome.html What's Related
Instant Message WebMail Radio People Yellow Pages Download Calendar Channels
Worldwide Shipbuilding Marketplace
Fredrick K. Jones
FKJ Shipbuilding Corp.
Home
Catalog
Contracts
My Contracts
New Contract
Verification
Termination
Requests for Quote
Exchange
Auctions
Orders
Approvals
Reports
Administration
Mailbox
Main Help
Log Off
IBM Powered By WebSphere Commerce Suite MPE
Document: Done

Selling Signee :	Mr. Alan King
Coverage :	Special pricing condition
Minimum Monetary Amount:	$ 10,000.00USD
Maximum Monetary Amount:	$ 1,000,000.00USD
Currency:	USD
Begin Time:	July 31, 2000
End Time:	August 31, 2000
Contract Text :	
Contract File Attachment :	no attachment

Special pricing conditions

Category or Product Name	Type	Min Value/Quantity	Max Value/Quantity	Sum Value/Quantity	Price Type	Price or Discount(%OFF)	Currency
Light Duty Propeller	Product	100	1,500	0	Price	$ 1,250.00USD	USD

Figure 112. Contract confirm: bottom portion

15.1.4 Example: buyer interaction

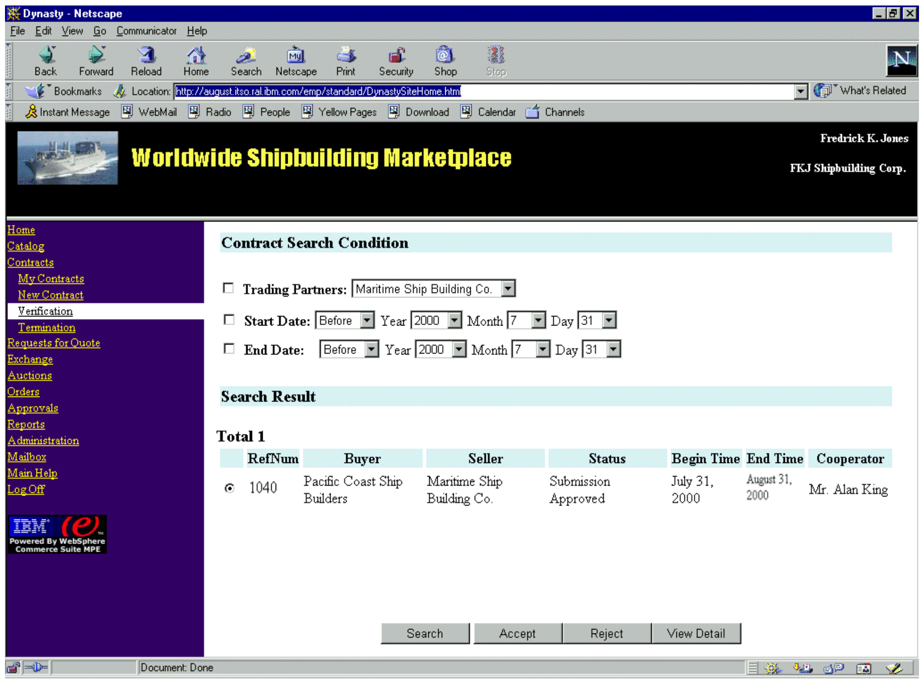
Buyers can interact with the contract subsystem either by working with contracts requiring verification or by using a winning RFQ to create a new contract. In this section we cover the process of verifying and accepting a contract. The creation of a contract from an RFQ is covered in 16.1, “RFQs” on page 367. We will also show how a contract purchase can be made against a contract.

Here are the steps a buyer with the Contract Administrator role takes to verify submitted contracts:

1. Buyer logs onto the e-Marketplace.
2. Buyer clicks **Contracts** in the navigation frame and then clicks **Verification**.
3. From the Verification window the buyer clicks the **Search** button to see all the contracts that require verification.
4. From the list, the buyer selects the contract and clicks **Accept**.
5. When accepted, a confirmation window is displayed.

A contract can be rejected through the same process. When the list of all contracts requiring verification is displayed the buyer has the option of rejecting the contract submission.

Figure 113 shows the list of contracts that require verification. Figure 114 on page 363 and Figure 115 on page 364 show the verified contract after terms and conditions of the contract have been accepted by the buyer.


Dynasty - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Shop Stop
Bookmarks Location: http://august.itso.ral.ibm.com/emp/standard/DynastySiteHome.html What's Related
Instant Message WebMail Radio People Yellow Pages Download Calendar Channels
Worldwide Shipbuilding Marketplace
Fredrick K. Jones
FKJ Shipbuilding Corp.
Home
Catalog
Contracts
My Contracts
New Contract
Verification
Termination
Requests for Quote
Exchange
Auctions
Orders
Approvals
Reports
Administration
Mailbox
Main Help
Log Off
IBM Powered By WebSphere Commerce Suite MPE
Search Accept Reject View Detail
Document: Done

Contract Search Condition

☐ Trading Partners: Maritime Ship Building Co.

☐ Start Date: Before Year 2000 Month 7 Day 31

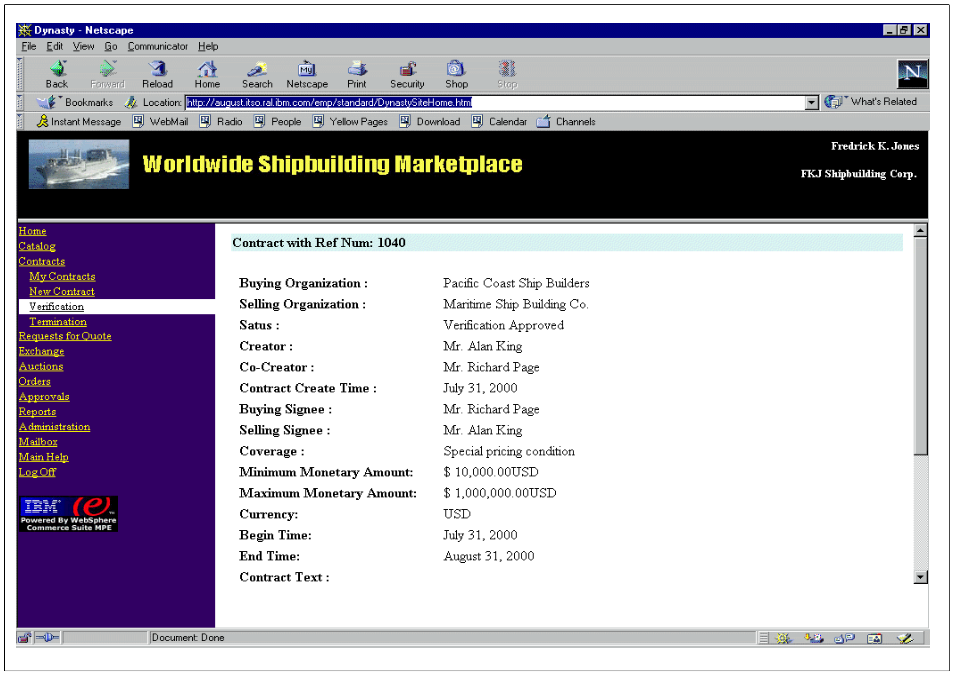
☐ End Date: Before Year 2000 Month 7 Day 31

Search Result

Total 1

	RefNum	Buyer	Seller	Status	Begin Time	End Time	Cooperator
◉	1040	Pacific Coast Ship Builders	Maritime Ship Building Co.	Submission Approved	July 31, 2000	August 31, 2000	Mr. Alan King

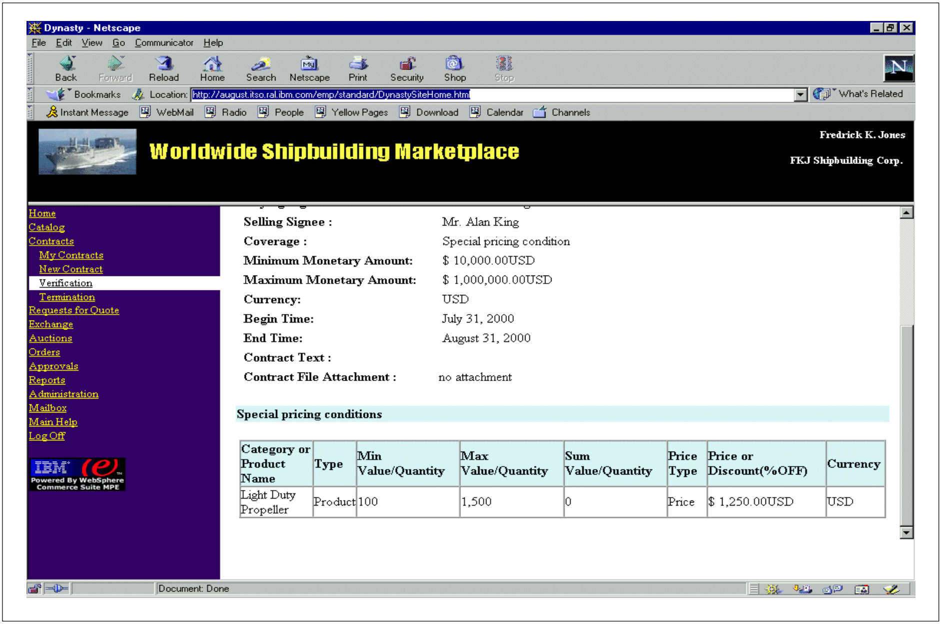
Figure 113. Contract list


Dynasty - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Shop Stop
Bookmarks Location: http://august.itso.ral.ibm.com/emp/standard/DynastySiteHome.html What's Related
Instant Message WebMail Radio People Yellow Pages Download Calendar Channels
Worldwide Shipbuilding Marketplace
Fredrick K. Jones
FKJ Shipbuilding Corp.
Home
Catalog
Contracts
My Contracts
New Contract
Verification
Termination
Requests for Quote
Exchange
Auctions
Orders
Approvals
Reports
Administration
Mailbox
Main Help
Log Off
IBM Powered By WebSphere Commerce Suite MPE
Document: Done

Contract with Ref Num: 1040

Buying Organization :	Pacific Coast Ship Builders
Selling Organization :	Maritime Ship Building Co.
Satus :	Verification Approved
Creator :	Mr. Alan King
Co-Creator :	Mr. Richard Page
Contract Create Time :	July 31, 2000
Buying Signee :	Mr. Richard Page
Selling Signee :	Mr. Alan King
Coverage :	Special pricing condition
Minimum Monetary Amount:	$ 10,000.00USD
Maximum Monetary Amount:	$ 1,000,000.00USD
Currency:	USD
Begin Time:	July 31, 2000
End Time:	August 31, 2000
Contract Text :	

Figure 114. Confirmation window: top portion


Dynasty - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Shop Stop
Bookmarks Location: http://august.itso.ral.ibm.com/emp/standard/DynastySiteHome.html
What's Related
Instant Message WebMail Radio People Yellow Pages Download Calendar Channels
Worldwide Shipbuilding Marketplace
Fredrick K. Jones
FKJ Shipbuilding Corp.
Home
Catalog
Contracts
My Contracts
New Contract
Verification
Termination
Requests for Quote
Exchange
Auctions
Orders
Approvals
Reports
Administration
Mailbox
Main Help
Log Off
Document: Done

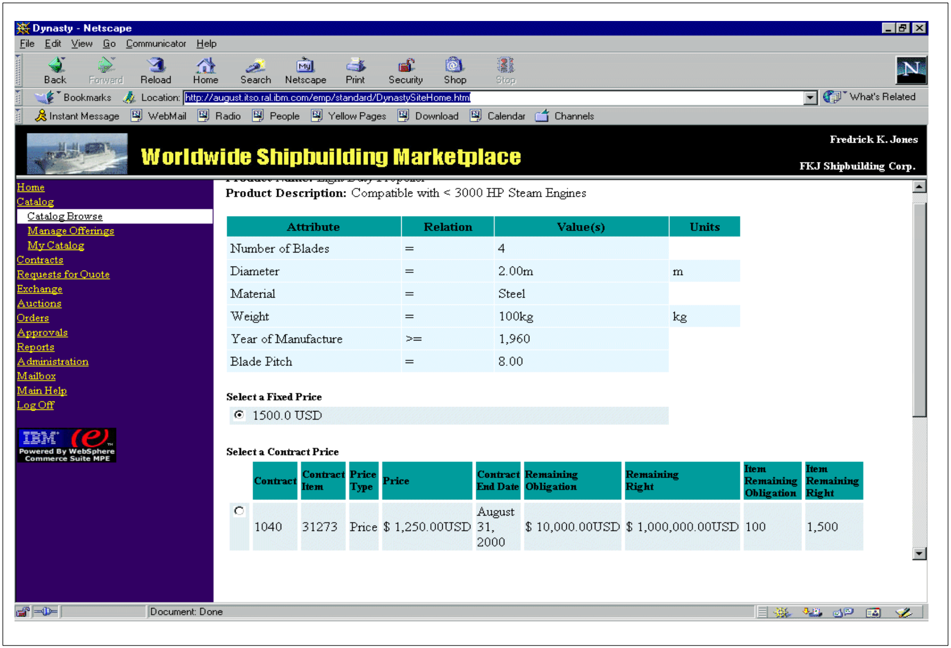
Selling Signee :	Mr. Alan King
Coverage :	Special pricing condition
Minimum Monetary Amount:	$ 10,000.00USD
Maximum Monetary Amount:	$ 1,000,000.00USD
Currency:	USD
Begin Time:	July 31, 2000
End Time:	August 31, 2000
Contract Text :	
Contract File Attachment :	no attachment

Special pricing conditions

Category or Product Name	Type	Min Value/Quantity	Max Value/Quantity	Sum Value/Quantity	Price Type	Price or Discount(%OFF)	Currency
Light Duty Propeller	Product	100	1,500	0	Price	$ 1,250.00USD	USD

Figure 115. Confirmation window: bottom portion

Now that the contract is accepted and signed by both parties, the buyer can use catalog-based buying to make purchases against the contract. Please refer to 14.5, “Buyer interaction: examples of catalog based buying” on page 333 for the details of catalog based buying. In Figure 116 on page 365 you can see that a contract offering is now available to the buyer as a result of actions we took in this section. When a buyer makes purchases against this contract, the obligation and rights are updated to reflect the current values.


Dynasty - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Shop Stop
Bookmarks Location: http://august.itso.ral.ibm.com/emp/standard/DynastySiteHome.html
What's Related
Instant Message WebMail Radio People Yellow Pages Download Calendar Channels
Worldwide Shipbuilding Marketplace
Fredrick K. Jones
FKJ Shipbuilding Corp.
Home
Catalog
Catalog Browse
Manage Offerings
My Catalog
Contracts
Requests for Quote
Exchange
Auctions
Orders
Approvals
Reports
Administration
Mailbox
Main Help
Log Off
Powered By WebSphere Commerce Suite MPE
Document: Done

Product Description: Compatible with < 3000 HP Steam Engines

Attribute	Relation	Value(s)	Units
Number of Blades	=	4	
Diameter	=	2.00m	m
Material	=	Steel	
Weight	=	100kg	kg
Year of Manufacture	>=	1,960	
Blade Pitch	=	8.00	

Select a Fixed Price

● 1500.0 USD

Select a Contract Price

	Contract	Contract Item	Price Type	Price	Contract End Date	Remaining Obligation	Remaining Right	Item Remaining Obligation	Item Remaining Right
○	1040	31273	Price	$ 1,250.00USD	August 31, 2000	$ 10,000.00USD	$ 1,000,000.00USD	100	1,500

Figure 116. Catalog-based buying using a contract offering

15.1.5 Interaction with other components and subsystems

The contract subsystem interacts with the other subsystems using function calls between subsystems and objects shared by all subsystems in the database.

The following interactions exist between the contract and other subsystems:

- Catalog:

  The contract subsystem interacts with the catalog subsystem to provide contract-specific information to the catalog subsystem and to retrieve product category information from the catalog subsystem.

- Access Control

  The contract subsystem interacts with the access control subsystem to validate user access and execute authority.

- Approval Flow

  The contract subsystem interacts with the approval flow subsystem to establish an approval process.

Figure 117 on page 366 is a pictorial representation of the interaction between the contract subsystem and other e-Marketplace subsystems.

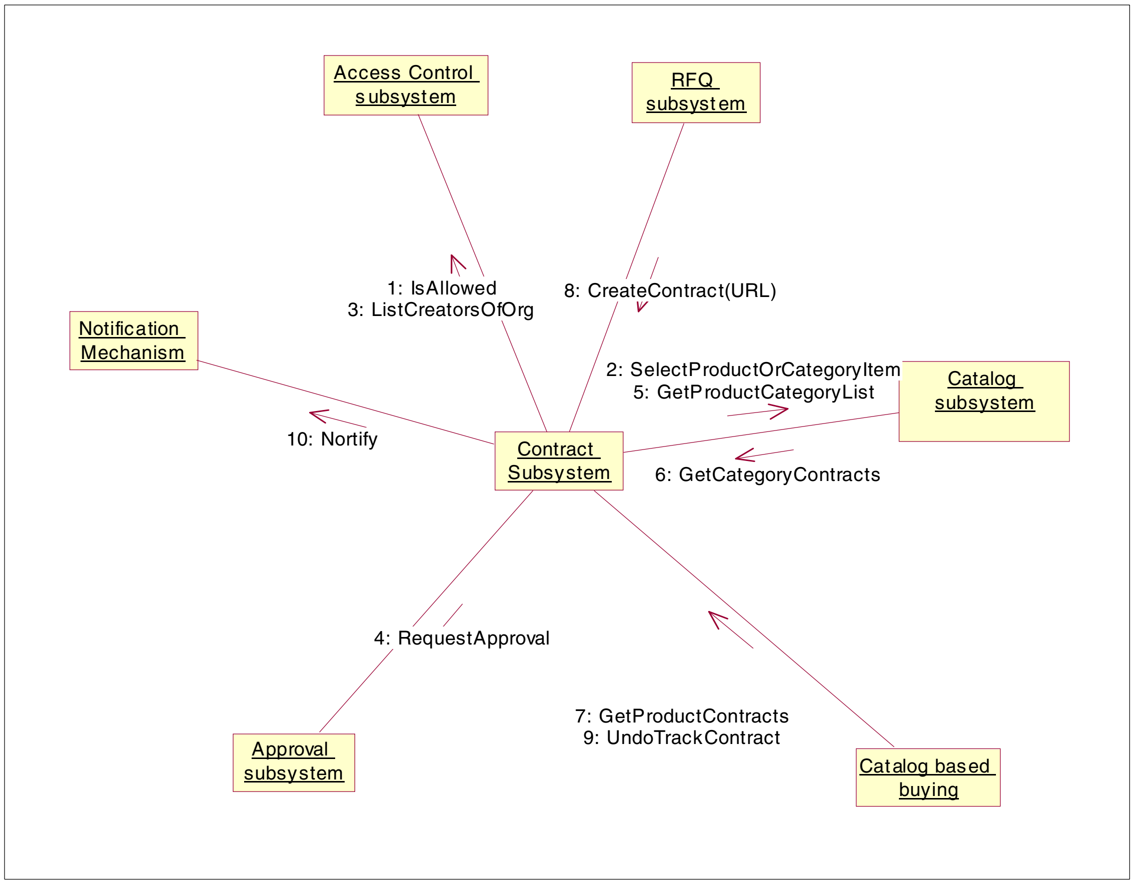


Figure 117. Subsystem interaction

Chapter 16. Example - negotiation subsystem

In the previous chapters we covered the catalog subsystem and pricing and contract subsystems and provided you with information and examples of how to create and manage the e-Marketplace catalog and contracts. We noted that as well as catalog-based buying, including contracts, there are other methods through which transactions take place in an e-Marketplace. These methods include:

- Auctions
- RFQs
- Exchanges

Auctions, RFQs, and exchanges are considered part of the negotiation subsystem in the WebSphere Commerce Suite, Marketplace Edition. In this chapter we provide information and examples of how each of these buying methods work.

It is important to note that RFQs and exchanges are new components and there is no equivalent of them in the WebSphere Commerce Suite 4.1. Auctions are a carry-over from WebSphere Commerce Suite 4.1 with very little change. When discussing auctions we will only provide information on the modifications to the auctions component of the WebSphere Commerce Suite 4.1.

In this chapter we provide you with details for RFQs, auctions, and exchanges about the following:

- High-level overview
- Low-level design
- Supplier interaction
- Buyer interaction
- Interaction with other components and subsystems

16.1 RFQs

A Request For Quotation (RFQ) is the process where a buying organization solicits offers from selling organizations to obtain a suitable price for a given product or set of products.Typically the buyers who issue the RFQs validate the responses from the suppliers and select the best offer, which is most often the lowest price offer. In fact the RFQ process is the exact opposite of an auction process, where buyers bid and the highest bid is selected as a winner. Due to this similarity, RFQs are sometimes referred to as reverse auctions.

In the WebSphere Commerce Suite, Marketplace Edition for AIX, RFQs are part of the negotiation subsystem.

16.1.1 High-level overview

An RFQ is always created by the buying organization members. RFQs can be created for existing product descriptions in the e-Marketplace, to obtain a better price, or for products that are not part of the e-Marketplace catalog, to create an alternate method of buying in the e-Marketplace. In either case, the buying organization issues an RFQ either to a fixed set of organizations or to all organizations in the e-Marketplace. The supplier organizations review the RFQs and decide if they wish to participate. If a supplier organization decides to participate, a response is created and submitted. As long as an RFQ is still active, suppliers can modify or retract the responses they have submitted. When an RFQ closes, the buyer reviews and evaluates responses and selects one or more winners. After evaluation is complete,the suppliers can view the result of the evaluation.

A winning RFQ can be used to create and place orders or it can be used to create a contract offering.

16.1.2 Low-level design

In this section we focus on the low-level design of the RFQs. This is particularly useful to developers who will be using the WebSphere Commerce Suite, Marketplace Edition for AIX. We will cover the design principles, use cases, state diagrams and code-level components of the RFQs. We do not intend to discuss the code-level components in detail, since that is outside the scope of this redbook. For detailed information, we recommend the online documentation and the JavaDocs where appropriate.

16.1.2.1 Design principles

The RFQ component of the negotiation subsystem is built using the existing WebSphere Commerce Suite 4.1 technology, mainly the auctions component, and some new components created in the WebSphere Commerce Suite, Marketplace Edition for AIX, Java Programming Model.

We provide information about the interaction controllers, commands, JSPs, objects and database tables that make up the RFQ component.

Just like contracts, RFQs need to go through several states before resulting in a winning RFQ. The procedure is a three-stage process. In the first stage, the buyer creates and publishes an RFQ to targeted organizations. In the current WebSphere Commerce Suite, Marketplace Edition for AIX implementation of RFQs there could only be one version of a given RFQ. Upcoming releases of the WebSphere Commerce Suite, Marketplace Edition for AIX may include versioning for RFQs. In the second stage, suppliers from targeted organizations prepare and submit responses. Versioning is allowed for responses. The final stage encompasses the buyer closing the RFQ and selecting a winner or winners.

When winning responses are created in the e-Marketplace, buyers can proceed to place orders or create contracts from those winning responses.

Throughout this process the states of the RFQ and responses change constantly. This state transition is managed by the flex flow state machine. Flex flow is explained in details in chapter Chapter 17, “Example - additional e-Marketplace infrastructure” on page 441

Figure 118 on page 370 displays the state transition diagram of an RFQ with respect to buyer interaction.

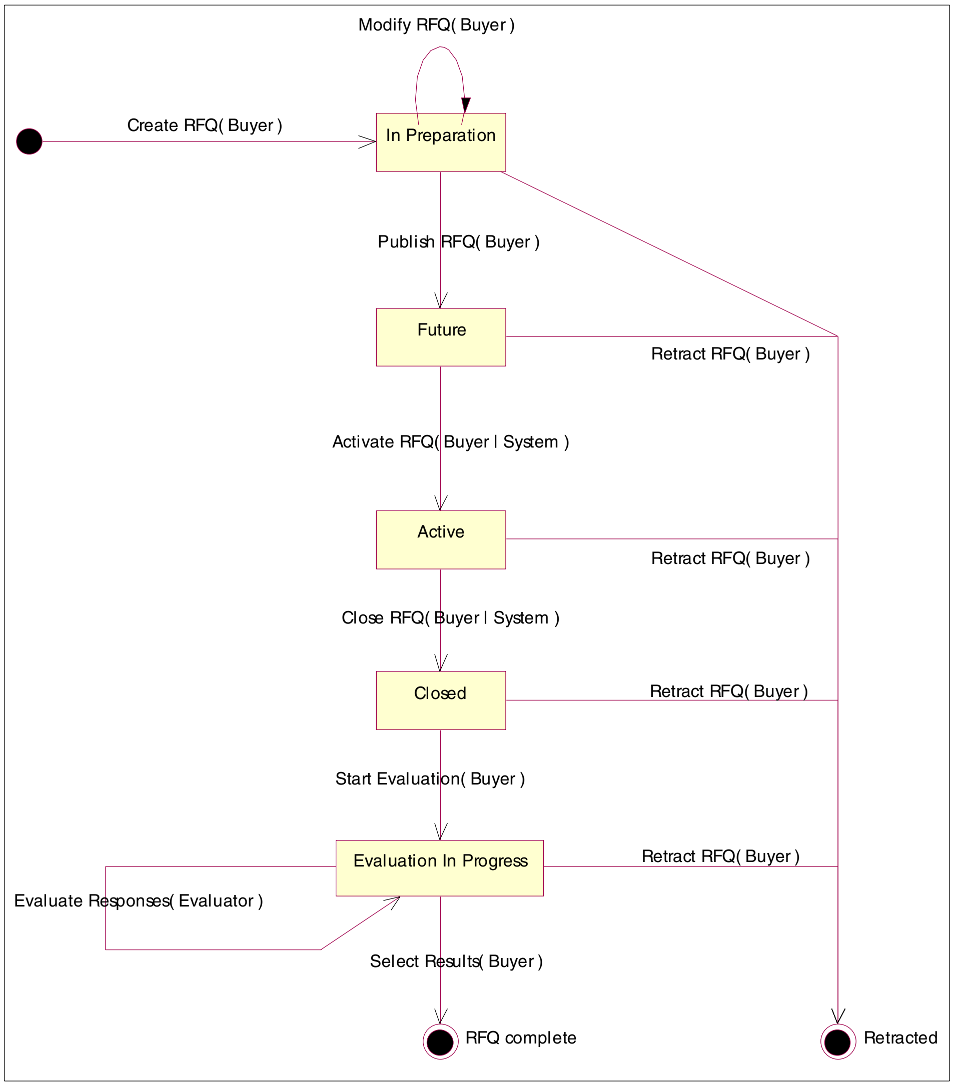


Figure 118. RFQ state transition (buyer side)

Figure 119 on page 371 displays the state transition diagram of responses with respect to the state of the RFQ and the supplier interaction.

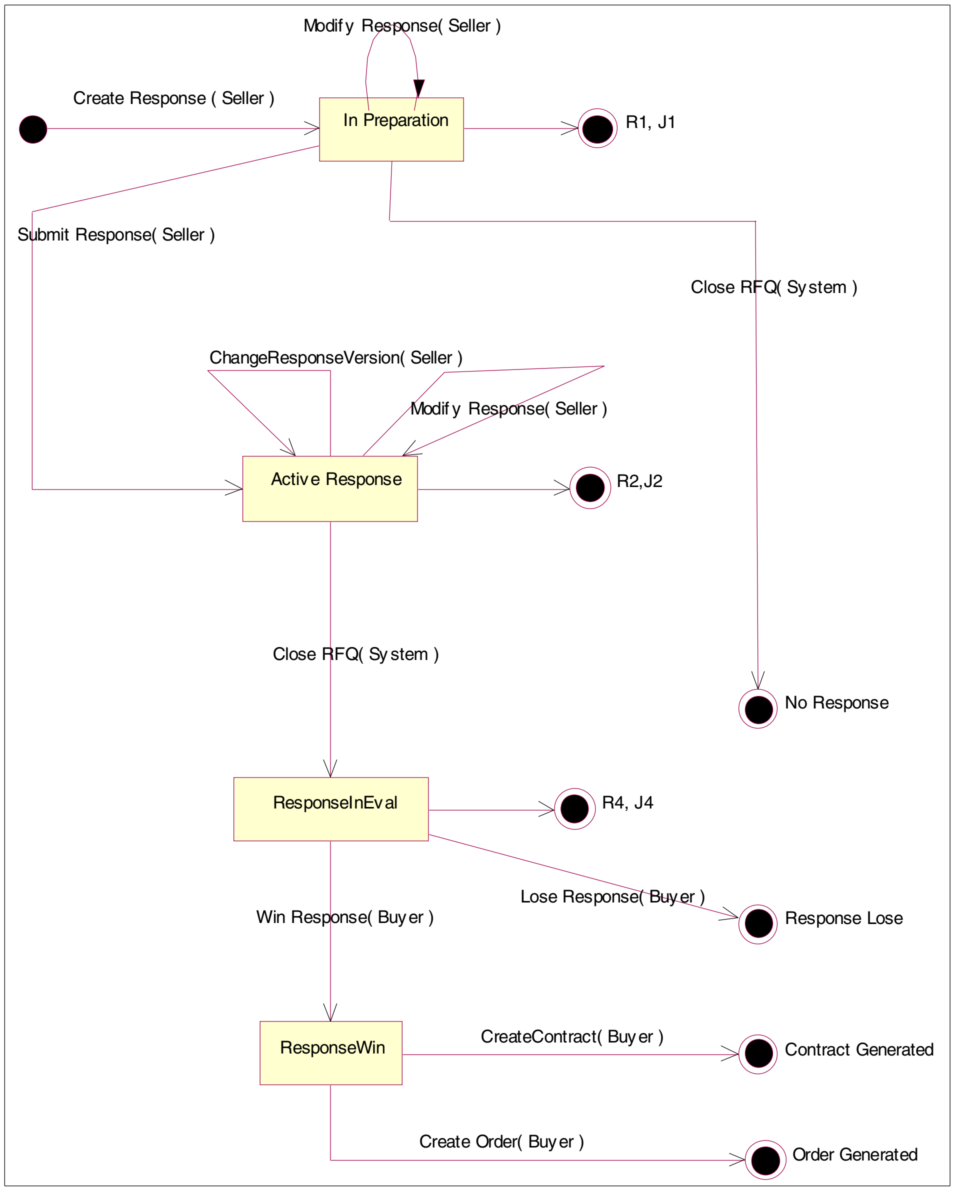


Figure 119. RFQ state transition (supplier side)

A scheduler mechanism is built into the WebSphere Commerce Suite, Marketplace Edition for AIX to determine if an RFQ should still be active or if it should be closed. This is determined based on the conditions provided at the time of creation. Four possible closing conditions exist for RFQs. They are:

- Closed at a fixed time
- Closed if a specified number of bids are received
- Closed at a fixed time *or* closed if specified number of bids are received
- Closed at a fixed time *and* closed if specified number of bids are received

The scheduler can be automatically close an RFQ if anyone of the above conditions are met. An authorized user can also manually close the RFQ before the closing condition is satisfied.

16.1.2.2 Use cases

The RFQ component of the negotiation subsystem is used by the e-Marketplace members in one of the following ways:

1. Create an RFQ and select winners

   The buyer selects a list of products in the catalog, and calls the `CreateRFQ` command. The RFQ is edited and products and attributes are added or deleted by using the `ModifyRFQ` command. Next target organizations are added to the RFQ. Every member in this target list is given access permissions using the access control interfaces to the products in the RFQ, The target list is maintained using access control to the RFQ, and products under it. The buyer publishes the RFQ using the `PublishRFQ` command. Once the RFQ is published sellers can view the RFQ. Either the scheduler, or the buyer calls the `ActivateRFQ` command to activate the RFQ. Sellers can copy the RFQ into a draft response object and create responses to the RFQs.

   Either the scheduler or the buyer issues the `CloseRFQ` command to close the RFQ. At this stage no more responses are accepted. When an RFQ closes, all responses are converted from In Preparation to No Response. By doing this all the responses that were not submitted are invalidated.

   The buyer starts the evaluation by calling the `StartEvaluation` command. All responses are viewed and an evaluation record is created using the `EvaluateResponses` command. Once the buyer is satisfied with the evaluations, the buyer issues the `SelectResults` command to complete the evaluation. This command looks into the BIDEVAL table and selects the bids for which there is at least one product that the response is identified as a winner, and issues the `WinResponse` command to the Response. For all other responses, a `LoseResponse` command is issued. At this stage results are final. The buyers can now go to the winning responses, and place orders or start a process to create a contract. Creation of an order or a contract is the final state of an RFQ.

2. Create responses

   The sellers can view the RFQs that their organization is authorized to participate in and are in future, active, closed, evaluation in progress or completed states. The seller selects an RFQ and responds to it by executing the `CreateResponse` command. The `CreateResponse` command creates a response by copying all of the RFQ into a response object and associating it to the seller and the RFQ. Using the `ModifyResponse` command, the seller will then respond to RFQ attributes, select products and attributes to respond to. Once the seller is satisfied with the response, the `SubmitResponse` command submits the response to the system.

3. Place an order

   The buyers select a winning response and issue a `CreateOrderFromRfqCmd` command to place an order.

4. Create a contract

   The buyers select a winning response and issue a `CreateContractFromRfqCmd` command to start the process of creating a contract.

16.1.2.3 Code level components

Interaction controllers

The RFQ component interaction controllers parse the input parameters and forward requests to the WebSphere Commerce Suite, Marketplace Edition for AIX command manager. The interaction controllers also execute commands that are passed back by the command factory.

The following lists the RFQ interaction controllers for WebSphere Commerce Suite, Marketplace Edition for AIX:

- CreateRFQ

  Used by buyers to create RFQs.

- ManageRFQ

  Used by buyers to manage an RFQ (for example, change RFQ status, add attributes, add offerings, etc.)

- ViewRFQ

  Used by buyers to view RFQs.

- CreateResponse

Used by suppliers to create responses.

- ManageResponse

  Used by suppliers to manage responses (for example change response status, etc)

- ViewResponse

  Used by suppliers to view responses.

Commands

Commands are the components of the RFQ that carry out the execution of the business logic and ultimately address the requests of the users. Commands interact with almost all other components of the RFQ from interaction controllers to JSPs and beans.

The following are the commands that exist in the RFQ component of the negotiation subsystem of WebSphere Commerce Suite, Marketplace Edition for AIX

- Buyer Commands
  - CreateRFQCmd

    Create an RFQ record in the AUCTINFO table using the inputs.

  - ModifyRFQCmd

    Change an RFQ record in the AUCTINFO table using the inputs.

  - PublishRFQCmd

    Change the status of an RFQ record from In-Preparation to Future.

  - ActivateRFQCmd

    Change the status of an RFQ record from Future to Active.

  - CloseRFQCmd

    Change the status of an RFQ record from Active to Closed.

  - StartEvaluationRFQCmd

    Change the status of a RFQ record from Closed to EvalInProgress. This command will allow evaluators' access to the response by adding appropriate rows into the RUCMEMREL table for each response. The list of evaluators is stored in AUCMEMREL table. In the current phase, we will have the buyer as the evaluator also. So, at CreateRFQ , we will add one more row in AUCMEMREL with the buyer as the evaluator role, and at StartEvaluation, we will add a row for the buyer (as an evaluator) for each of the responses.

  - EvaluateResponsesCmd

    Only to review the response.

  - SelectResultsCmd

    Change the status of an RFQ record from EvalInProgress to ResultsSelected. Send e-mail notification to those responding to this RFQ.

  - RetractRFQCmd

    Change the status of an RFQ record from In-Preparation, Future, Active or Closed status to Retracted.

- Seller Commands
  - CreateResponseCmd

    Create a response row for an RFQ. Insert all the attributes and attribute groups, and associate them with the response row that has just been created.

  - ModifyResponseCmd

    Change a response row for this response. Change all the attributes and attribute groups, and associate them with the specified response row.

  - SumbitResponseCmd

    Change the status of a response from In-Preparation to Active.

  - RetractResponseCmd

    Change the status of a response from current status to Retracted.

  - ChangeResponseVersionCmd

    Change the version of the response.

  - CreateContractFromRfqCmd

    Create a contract out of a winning response.

  - CreateOrderFromRfqCmd

    Create an order out of a winning response.

- Scheduler commands
  - Execute ActivateRFQ

    If an RFQ exists with an elapsed start time and current status of Future, activate it.

  - Execute CloseRFQ

if an RFQ exists with an elapsed close time and current status of Active, close it.

- ActivateRFQ

  Update status of RFQ to active, send messages to targeted suppliers.

- CloseRFQ

  Update status of RFQ to Closed.

- Message commands
  - CloseRFQMessage

    Issue a Closed RFQ message.
- Data Bean Commands
  - RfqVersionBeanCmd

    Get the RFQ version bean from the persistent RfqVersion object.
  - RfqVersionListBeanCmd

    Get a list of RFQ version beans.
  - RfqVersionAttributeBeanCmd

    Get an RFQ attribute bean for a given RFQ version.
  - RfqVersionAttributeListBeanCmd

    Get a list of RFQ attribute beans for a given RFQ version.
  - ResponseVersionBeanCmd

    Get the response version bean from the persistent ResponseVersion object.
  - ResponseVersionListBeanCmd

    Get a list of response version beans.
  - RespVersionAttrBeanCmd

    Get a response attribute bean for a given response version.
  - RespVersionAttrListBeanCmd

    Get a list of response attribute beans for a given response version.
  - RespVersionOfferingBeanCmd

    Get a response offering bean from the persistent RespVersionOffering object.
  - RespVersionOfferingListBeanCmd

    Get a list of response offering beans.

- RespVersionOfferingAttrBeanCmd

  Get a response offering attribute bean for a given response offering.
- RespVersionOfferingAttrListBeanCmd

  Get a list of response offering attribute beans for a given response offering.
- OfferingEvaluationBeanCmd

  Get a evaluation bean from the persistent OfferingEvaluation object.
- OfferingEvalListBeanCmd

  Get a list of evaluation beans for a given response version.

JSPs

The following are the JavaServer Pages (JSPs) that the RFQ component of the negotiation subsystem uses:

- reqListRFQ

  This JSP displays the list of RFQs to a buyer.
- reqViewRFQ

  This JSP displays the details of an RFQ to a buyer.
- reqViewResponse

  This JSP displays the responses to an RFQ to a buyer.
- reqCreateRFQ

  This JSP displays the RFQ creation form to a buyer.
- reqChangeStatus

  This JSP displays changes in status of an RFQ.
- reqEvaluateResponse

  This JSP displays the evaluation window to a buyer.
- reqModifyRFQ

  This JSP displays the RFQ allowing the buyer to perform modifications.
- reqRFQComplete

  This JSP displays the completion window of an RFQ.
- resListRFQ

  This JSP displays all RFQs to an authorized supplier.
- resViewRFQ

This JSP displays RFQ details to an authorized supplier.

- resCreateResponse

  This JSP displays the RFQ response form to an authorized supplier.

- resListHistory

  This JSP displays all RFQs that an authorized supplier has worked with.

- resResponseAttributes

  This JSP displays the form to a supplier to respond to attributes of an RFQ.

- resResponseProduct

  This JSP displays the form to a supplier to respond to a product in an RFQ.

- resViewResponse

  This JSP displays the details of a response the supplier has created.

- resModifyResponse

  This JSP displays the form to a supplier to modify a response.

- SearchResponse

  This JSP displays the search form to a supplier to find RFQs in the e-Marketplace.

  For more detailed information about these JSPs, please refer to the online documentation of the WebSphere Commerce Suite, Marketplace Edition for AIX.

Objects

WebSphere Commerce Suite, Marketplace Edition for AIX uses the following beans in the RFQ component of the negotiation subsystem:

- Beans:
  - BRfqVersion

    A specific version of RFQ; in the current release, only one version is supported for an RFQ.
  - BRfqVersionList

    A list of RFQ versions.
  - BRfqVersionAttributeValue

    An instance of an RFQ attribute value.
  - BRfqVersionAttribute

A combination of the attribute definition and one or more RFQ attribute values.

- BRfqVersionAttributeList

  A list of RFQ attributes.

- BResponseVersion

  A specific version of response.

- BResponseVersionList

  A list of response versions.

- BRespVersionAttrValue

  An instance of a response attribute value.

- BRespVersionAttr

  A combination of the attribute definition and one or more response attribute values.

- BRespVersionAttrList

  A list of response attributes.

- BRespVersionOffering

  An offering with a response to a RFQ.

- BRespVersionOfferingList

  A list of response offerings.

- BRespVersionOfferingAttrValue

  An instance of a response offering attribute value.

- BRespVersionOfferingAttr

  A combination of the attribute definition and one or more response offering attribute values.

- BRespVersionOfferingAttrList

  A list of response offering attributes.

- BOfferingEvaluation

  An entry representing the evaluation to a response offering.

- BOfferingEvaluationList

  A list of evaluations.

• Persistent Objects:

  - RfqVersion

The persistent object representing an RFQ version in the AUCTINFO table.

- RfqVersionAttributeValue

  The persistent object representing an RFQ attribute entry in the AUCATTRVAL table.

- ResponseVersion

  The persistent object representing a response version in the BIDTABLE table.

- RespVersionAttrValue

  The persistent object representing a response attribute entry in the BIDATTRVAL table.

- RespVersionOffering

  The persistent object representing a response offering entry in the BIDPRODLIST table.

- RespVersionOfferingAttrValue

  The persistent object representing a response offering attribute entry in the BIDPRODATRVAL table.

- OfferingEvaluation

  The persistent object representing a response evaluation entry in the BIDEVAL table.

Database tables
In this section we include the modifications and additions to the WebSphere Commerce Suite 4.1 database schemas with respect to the RFQs. Some new tables have been added and some existing tables have been modified to facilitate the storage data required for the e-Marketplace RFQs.

Figure 120 on page 381 depicts the relationship model of the RFQs component of the Negotiation subsystem.

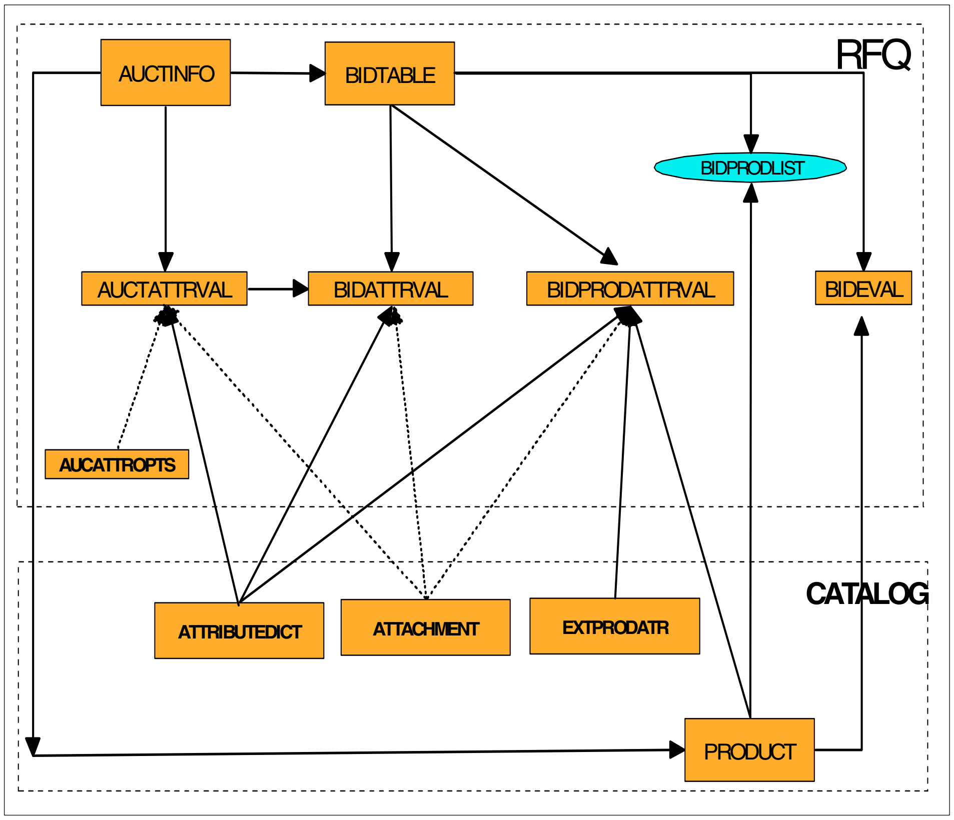


Figure 120. RFQ entity relationship diagram

AUCTINFO: The AUCTINFO table contains header (control) information for auctions and RFQs (reverse auctions). Every auction points to a product in the catalog. In the case of RFQs, the product pointed to is the product bundle that represents all the products in the RFQ. Several extensions to the existing

NC AUCTINFO table have been made in order to accommodate multiple supplier auctions and RFQs.

Table 34. AUCTINFO

Column Name	Data Type	Description
AUREFNUM	integer not null	RFQ reference number. This is the primary key.
AUBYRNUM	char(128)	Tracking number (provided by the member). If RFQ is a template, then this field will contain the template name.
AUTMPL	char(128)	The template used in the creation of this RFQ. This field is NULL if no templates are used.
AUCREATOR	integer not null	The creator of the RFQ. Foreign key reference to SHRFNBR in SHOPPER (delete rule = no action).
AUPRREFN	integer not null	Product Reference Number. This is a foreign key that references the PRRFNBR column in the PRODUCT table. In RA this points to a bundle (or a package) which point s to the set of products in the RFQ (delete rule = cascade).
AUMERREFN	integer not null	Merchant Reference Number. Points to the organization on whose behalf the RFQ is created. This is a foreign key referencing the MERFNBR column in the **MERCHANT** table. (delete rule = cascade.)
AUDIRECTION	char(4) not null	Direction: F - Forward (Current NC auctions) R - Reverse (RFQ style)
AUTYPE	char(4) not null	RFQ Type: O - Open Cry SB - Sealed Bid D - Dutch For this release of RA, it must only be SB.

AUSTATUS	char(4) not null	RFQ Status: We will use the prefix R for all reverse auction statuses: RIP - RA In Preparation RIF - RA that will accept responses in the future RAC - Active: Bids can be submitted. State reached at auction start time. RCL - Closed: Bids can no longer be submitted (or will be marked LATE) RRE - Responses being evaluated RAW - Awarded: Winners have been selected RAR - Archived (to be used in later version) RWI - Retracted/Withdrawn RT - RFQ Template
AUQUANT	num(15,2) not null	Not applicable for RA (set to 0)
AUMINBID	num(15,2)	Not applicable for RA
AUCUR	char(10) not null	Currency in which the price is expressed. There is currently no check for the validity of the format .
AURULETYPE	integer not null	RFQ closing rules: 1 -RFQ closes at a fixed end time 2 -RFQ closes if a specified number of bids are received 3 -RFQ closes based on logical OR of 1 and 2 4 -RFQ closes based on logical AND of 1 and 2
AUSTTIM	timestamp	Time at which the auction is scheduled to start.
AUENDTIM	timestamp	Time at which the auction is scheduled to end.
AUCURENDTIM	timestamp	Closing time of the RFQ. This field is filled when the `CloseRFQ` command is executed.
AUDURATION	timestamp	If the RFQ is a template, then this will contain the duration of the RFQ. So, the RFQs created using this template will run for this duration.

AUMINNOBIDS	integer	Minimum number of bids needed for the RFQ to close.
AUDEPOST	num(15,2)	Payment authorization required with each bid. Bidder with winning bid would forfeit this deposit if he/she does not complete the RFQ contract or order.
AURULEMACRO	char(254) not null	Stores the macro/JSP file name for the RFQ display.
AUPRDMACRO	char(254) not null	This field is not used by RA.
AUOPRDMACRO	char(254)	This field is not used by RA.
AUBDRULE	integer	Not applicable for RA.
AUDESC	varchar (254)	RFQ description.
AULONGDESC	bigint	Long description. It is stored as an HTML file. Directed to the FILEREFNUM in ATTACHMENTS table, but not a foreign key.
AUTERMS	bigint	Terms and conditions. It is stored as an HTML file. Directed to the FILEREFNUM in ATTACHMENTS table, but not a foreign key. Specified bidders must accept terms before being allowed to bid.
AUBESTBID	char(36)	Not applicable for RA .
AUSTARTPRICE	num(15,2)	Not applicable for RA.
AUCURQUANT	num(15,2)	Not applicable for RA.
AUCURPRICE	num(15,2)	Not applicable for RA.
AUUPDTIME	timestamp	Timestamp of the row creation. This contains the timestamp of when the RFQ was created or modified.
AUFLAGS	char(32)	C Current published version N new version O Old version. P Version pending approval
AULASTUPD	timestamp	The time when the auction bookkeeping activities, such as dispatch of notification messages, computation of winning bids, and end times, were last performed.

AUFIELD1	integer	Customization field. Not used in base RA.
AUFIELD2	integer	Customization field. Not used in base RA.
AUFIELD3	num(15,2)	Customization field. Not used in base RA.
AUFIELD4	num(15,2)	Customization field. Not used in base RA.
AUFIELD5	varchar(254)	Customization field. Not used in base RA.
AUFIELD6	varchar(254)	Customization field. Not used in base RA.
AUPRICING	char(4)	Not applicable for RA.
AUHIGHBID	char(36)	Not applicable for RA.
AUHISTORYNUM	integer	Contains the parent RFQ reference number. If equal to the AUREFNUM, then this is the parent record.
AUVERSIONNUM	integer	Currently not used in RA. It could be used in the future versions. A version number for the RFQ. Currently defaulted to 1.
AUVERSIONDESC	char(254)	In current release, this column is used as the name of RFQ.

AUCATTRVAL: This table contains attribute values for RFQs. The attributes defined here come from ATTRIBUTEDEF, the attribute definition table. The attribute type definition is derived from the ATTRIBUTEDEF table.

Table 35. AUCTTRVAL

Column Name	**Data Type**	**Description**
ATREFNUM	bigint not null	Primary key. Automatically generated.
ATADREFNUM	integer not null	Reference to the attribute definition. Used as a foreign key to ATTRIBUTEDEF.ADREFNUM.
ATNAME	varchar(128)	Attribute name. This will always carry the attribute name copied from the attribute dictionary.
ATAUREFNUM	integer not null	Foreign key reference to AUCTINFO.AUREFNUM.
ATUSAGE	small int not null	The usage of this attribute 0: Non-changeable 1: Default value (changeable/negotiable)

ATMANDATORY	small int not null	1: mandatory attribute 0: optional attribute
ATNUMVAL	small int not null	The number of values for this attribute 1: indicates it is a single-valued attribute N (>1): It is a multi-valued attribute (N indicates that number)
ATATTROPTS	small int not null	0: No attribute options (default) 1: Attribute options exist for at least one of ATUNIT, ATOPERATOR, ATVALUE
ATCONTEXT	integer not null	1: RFQ 2: Reserved for auction, will be used later
ATUNIT	varchar(32)	UNITS of the data: Can be MHz, lb., Oz., Kgs, in,..... This list is created using the UNITS table. The actual units are copied from the UNITS table to this column. if the value is “AUCATTROPTS”, use the AUCATTROPTS table to find the options available for this column.
ATOPERATOR	varchar(32) not null	Values described in OPERATORS table. The actual OPERATORS are copied from the operators table to this column. if the value is “AUCATTROPTS”, use the AUCATTROPTS table to find the options available for this column. This contains the relationship between the current value in the row and the name of the attribute. The restrictions posed by the type value on the range of values for this field should be controlled by the application logic using the attributes table. Possible values include: <,<=,>,>= for type Range = for type Fixed value or File I for Boolean C for Enumerated The idea is that we will have one row for each value of the attribute and they are all linked with the same name, and each value is appropriately related to that attribute name.

ATVALUE	char(254)	Value of the attribute. For FILE, this contains the text tobe displayed that can be clicked to access the file. If null, the file URL is displayed as the text. For ENUMERATED, this contains values separated by a comma “,” For URL it is assumed to have the HTTP:// or FTP:// or other protocol identifier. For DATE, TIME, DATETIME, it is assumed to have the appropriate syntax. The actual syntax of the data is currently not performed in the RFQ process. It will be added later. If the value is AUCATTROPTS, use the AUCATTROPTS table to find the options available for this column.
ATATTACH	bigint	The file attachment for this attribute (if any). Foreign key reference to ATTACHMENT table.
ATSEQNUM	integer	Order in which this field is presented. Unique for a set of attributes belonging to same trading instance (for example RFQ, Auction).
ATFIELD1	integer	Customization field.
ATFIELD2	varchar (254)	Customization field.

AUCATTROPTS (New): This table contains options specified for attributes in the AUCATTRVAL table. Options could include what operators or units or even values are allowed. This information is especially useful for specifying templates such as RFQ templates.

Table 36. AUCTTROPTS

Column Name	Data Type	Description
AOATREFNUM	bigint not null	Foreign key to AUCATTRVAL.ATREFNUM.
AOOBJTYPE	char(1) not null	Enumeration to distinguish between Operators, Units, and Values: O: Operators U: Units V: Values
AOVALUE	char (254)	Value corresponding to this particular option, for example: For UNITS: MHz, Kgs,... copied from the UNITS table For OPERATORS: >,<,<=..., copied from the OPERATORS table For VALUES: Actual value options like 100, 20/12/2000 etc.
AODEFAULT	small int	Indicates if the specified option is default option for that attribute 1: YES, 0: NO

BIDTABLE: The BIDTABLE contains information about all the bids created in response to active RFQs.

Table 37. BIDTABLE

Column Name	Date Type	Description
BDREFNUM	char(36) not null	Bid reference number. This is the primary key.
BDVERSION	integer	The version number (that is the number of iterations that the bidder has gone through changing a bid). Useful for displaying change history. In the current edition of RFQ this will default to 1
BDAUCREF	integer not null	Auction/RFQ reference number. This is a foreign key which references the AUREFNUM column in the AUCTINFO table. (delete rule = cascade)
BDSHRFN	integer not null	Sellers' member number. This is a foreign key which references the SHRFNBR column in the **shopper** table. (delete rule = no action)

BDTYPE	char (4)	The type of bid: F: Response to forward auction (current NC bids) R: Response to reverse auction (RFQ bids)
BDCUSTRATING	integer	Not applicable for RA.
BDSALESRFN	integer	Not applicable for RA.
BDORDERRFN	integer	Not applicable for RA.
BDQUANT	num(15,2) not null	Not applicable for RA.
BDVALUE	num(15,2) not null	Not applicable for RA.
BDTIME	timestamp not null	Time at which the bid was submitted.
BDCREATETIME	timestamp	Time at which the bid was created.
BDCHANGETIME	timestamp	Last time of status change.
BDSTATUS	char(4) not null	Bid status: We will use the prefix R for all reverse auction statuses: RIP - Bid In Preparation/Future RSU - Submitted (on time) -- available for evaluation REV - Bid being evaluated RCH - Bid awaiting sellers approval of his/her bid for new version of RFQ. RWO - Bid won/awarded RLO - Bid lost ROR - Bid converted to order RCO - Bid converted into a contract RNP - No response to RFQ RRW - RFQ retracted/withdrawn RWI - Retracted/withdrawn
BDCHILD	char(36)	Bid reference number of an earlier version of this bid (if any). Same as bdrefnum if this is the first version.
BDPAYMTHD	char(5) not null	Not applicable for RA.
BDPAYDEVC	char(64) not null	Not applicable for RA.
BDDATEXP	timestamp	Not applicable for RA.

BDMAXAAMT	num(15,2)	The amount deposited for the current bid.
BDSARFN	integer not null	Not applicable for RA.
BDSMRFN	integer	Not applicable for RA.
BDWINOPT	char(4) not null	Not applicable for RA.
BDWINVALUE	num(15,2)	Not applicable for RA.
BDWINQTY	num(15,2)	Not applicable for RA.
BDEXPTIME	timestamp	For future use. This field will be used to store the expiration date of the bid Currently this will be defaulted to NULL
BDMESSAGE	long varchar	Response text for RA.
BDFLAGS	char(32)	C Current published version N new version O Old Version P Version pending approval
BDOTHERNEG	integer	Not applicable for RA
BDFIELD1	integer	Customization field. Not used in base RA.
BDFIELD2	num(15,2)	Customization field. Not used in base RA.
BDFIELD3	varchar(254)	Customization field. Not used in base RA.

BIDPRODLIST: This table lists the products is a bid.

Table 38. BIDPRODLIST

Column Name	Data Type	Description
BPLBIDREFNUM	char(36) not null	Bid ID, Foreign key reference to BIDTABLE.BDREFNUM (delete rule = cascade)
BPLPRNBR	integer not null	Product reference number, references PRRFNBR in product table (delete rule = cascade)
BPLSEQNUM	integer	The order in which this field is presented (similar to serial number). Unique for a bid.
BPLFIELD1	integer	Customization field. Not used in base RA.
BPLFIELD2	integer	Customization field. Not used in base RA.

BIDATTRVAL: This table contains attribute values for responses. The attributes come from ATTRIBUTEDEF, the attribute definition table. The attribute type definition is also derived from the ATTRIBUTEDEF table.

Table 39. BIDATTRVAL

Column Name	Data Type	Description
BRREFNUM	bigint not null	Primary key. Automatically generated.
BRBIDREFNUM	char(36) not null	Foreign key reference to BIDTABLE.BDREFNUM.
BRRESTOATR	bigint	Attribute to which this is a response. Foreign key reference to AUCATTRVAL.ATREFNUM.
BRADREFNUM	integer not null	Reference to the attribute definition. Used as a Foreign key to ATTRIBUTEDEF.ADREFNUM.
BRNAME	varchar(128)	Attribute name. This will always carry the attribute name copied from the attribute dictionary.
BRUSAGE	small int not null	The usage of this attribute 0: Non-changeable 1: Default value (changeable/negotiable)
BRMANDATORY	small int not null	1: mandatory attribute 0: optional attribute
BRNUMVAL	small int not null	The number of values for this attribute 1: Indicates it is a single-valued attribute N (>1): It is a multi-valued attribute (N indicates that number)
BRATTROPTS	small int not null	0: No attribute options (default) 1: Attribute options exist for at least one of BRUNIT, BROPERATOR, BRVALUE
BRCONTEXT	integer not null	1: RFQ 2: Reserved for auction, will be used later
BRUNIT	varchar(32)	Units of the data: Can be MHz, lb., Oz., Kgs, in,..... This list is created using the UNITS table. The actual units are copied from the UNITS table to this column. if the value is “AUCATTROPTS”, use the AUCATTROPTS table to find the options available for this column.

BROPERATOR	varchar(32) not null	Values described in OPERATORS table. The actual operators are copied from the OPERATORS table to this column. if the value is “AUCATTROPTS”, use the AUCATTROPTS table to find the options available for this column. This contains the relationship between the current value in the row and the name of the attribute. The restrictions posed by the type value on the range of values for this field should be controlled by the application logic using the attributes table. Possible values include: <,<=,>,>= for type Range = for type Fixed value or File I for Boolean C for Enumerated The idea is that we will have one row for each value of the attribute and they are all linked with the same name, and each value is appropriately related to that attribute name.
BRVALUE	char(254)	Value of the attribute. For FILE, this contains the text to be displayed that can be clicked to access the file. If null, the file URL is displayed as the text. For ENUMERATED, this contains values separated by a comma “,” For URL it is assumed to have the HTTP:// or FTPFILE:// or whatever protocol identifier. For DATE, TIME, DATETIME, it is assumed to have the appropriate syntax. The actual syntax of the data is currently not performed in the RFQ process. It will be added later. If the value is AUCATTROPTS, use the AUCATTROPTS table to find the options available for this column.

BRATTACH	bigint	The file attachment for this attribute (if any). Foreign key reference to ATTACHMENT table.
BRSEQNUM	integer	The order in which this field is presented. Unique for a set of attributes belonging to same trading instance (for example RFQ, auction).
BRFIELD1	integer	Customization field.
BRFIELD2	varchar (254)	Customization field.

BIDPRODATRVAL: This table connects each product in a bid to a set of attribute values. These attributes are currently used only by the RFQ process.

Table 40. BIDPRODATRVAL

Column Name	Data Type	Description
BPREFNUM	bigint not null	Primary key. Automatically generated.
BPBIDREFNUM	char(36) not null	Foreign key reference to BIDTABLE.BDREFNUM.
BPPRODREFNUM	integer not null	Product Reference Number. This is a foreign key that references the PRRFNBR column in the PRODUCT table
BPRESPRODATR	bigint	Attribute of the offering to which this is a response. Foreign key reference to EXTPRODATR. EAREFNUM.
BPADREFNUM	integer not null	Reference to the attribute definition. Used as a foreign key to ATTRIBUTEDEF.ADREFNUM.
BPNAME	varchar(128)	Attribute name. This will always carry the attribute name copied from the attribute dictionary.
BPUSAGE	small int not null	The usage of this attribute 0: Non-changeable 1: Default value (changeable/negotiable)
BPMANDATORY	small int not null	1: mandatory attribute 0: optional attribute

BPNUMVAL	small int not null	The number of values for this attribute 1: Indicates it is a single-valued attribute N (>1): It is a multi-valued attribute (N indicates that number)
BPATTROPTS	small int not null	0: No attribute options (default) 1: Attribute options exist for at least one of BPUNIT, BPOPERATOR, BPVALUE
BPCONTEXT	integer not null	1: RFQ 2:Reserved for auction, will be used later
BPUNIT	varchar(32)	Units of the data: Can be MHz, lb., Oz., Kgs, in,..... This list is created using the UNITS table. The actual units are copied from the UNITS table to this column. if the value is “AUCATTROPTS”, use the AUCATTROPTS table to find the options available for this column.
BPOPERATOR	varchar(32) not null	Values described in OPERATORS table. The actual operators are copied from the OPERATORS table to this column. if the value is “AUCATTROPTS”, use the AUCATTROPTS table to find the options available for this column. This contains the relationship between the current value in the row and the name of the attribute. The restrictions posed by the type value on the range of values for this field should be controlled by the application logic using the attributes table. Possible values includ: <,<=,>,>= for type Range = for type Fixed value or File I for Boolean C for Enumerated The idea is that we will have one row for each value of the attribute and they are all linked with the same name, and each value is appropriately related to that attribute name. if the value is ATTROPTS, use the ATTROPTS table to find the options available for this column.

BPVALUE	char(254)	Value of the attribute. For FILE, this contains the text to be displayed that can be clicked to access the file. If null, the file URL is displayed as the text. For ENUMERATED, this contains values separated by a comma “,” For URL it is assumed to have the HTTP:// or FTP:// or other protocol identifier. For DATE, TIME, DATETIME, it is assumed to have the appropriate syntax. The actual syntax of the data is currently not performed in the RFQ process. It will be added later. If the value is AUCATTROPTS, use the AUCATTROPTS table to find the options available for this column.
BPATTACH	bigint	The file attachment for this attribute (if any). Foreign key reference to ATTACHMENT table.
BPSEQNUM	integer	The order in which this field is presented. Unique for a set of attributes belonging to same trading instance (for example RFQ, auction).
BPFIELD1	integer	Customization field.
BPFIELD2	varchar (254)	Customization field.

BIDEVAL: This table contains all the bid evaluations for the RFQ. In the current RFQ process, the buyer will enter which of the products from which of the sellers are accepted. For each product the buyer selects, there will be an entry in this table. This table also allows the buyer to specify explicitly that some products are not selected. If there is no entry for a response in this table, then it is assumed to default to be not winning anything. When the buyer completes all evaluations and executes the `SelectResults` command, the `SelectResults` command finds what responses won at least one product that they have responded ta and sends them a WINRESPONSE message, and sends a LOSERESPONSE message to everyone else.

Table 41. BIDEVAL

Column Name	Data Type	Description
EVALRFN	bigint not null	Evaluation ID, primary key
EVUSERID	integer	Evaluator’s user ID. Foreign key. References to SHRFNBR field in the SHOPPER table
EVRFQID	integer not null	RFQ id. Foreign Key. References aurefnum in the AUCTINFO table
EVBIDID	char(36) not null	Rresponse ID. Foreign key. References BDREFNUM in the RESPONSE table
EVPRDID	integer	Offering ID for which the supplier has responded. If null, the response is for the whole response, not a product. If not null, it would refer to PRFNBR in the PRODUCT table.
EVCOMNT	long varchar	Comments.
EVRANK	integer	Rranking number, a numeric value to denote the evaluator’s preference. In current edition of RFQ, 1 for product accepted, and 0 for product not accepted .
EVDATE	timestamp	Timestamp when the evaluation was done.
EVFIELD1	integer	Merchants’ customization field.
EVFIELD2	integer	Merchants’ customization field.
EVFIELD3	integer	Mmerchants’ customization field.

DISCUSSION: This table contains questions from sellers and responses from buyers related to a specific RFQ. Entire discussion threads can be constructed from this table by means of the DSPARENTID column. In the current version of RFQ, the macros created by the auctions for discussions will be used without modifications.

Table 42. DISCUSSION

Column Name	Date Type	Description
dsmsgid	integer not null	Message ID. This is the primary key for the table.
dsparentid	integer not null	Parent ID of the message.
dsstatus	char(4) nut null	Status of the message: A - Active D - Deleted
dssender	integer not null	Reference number of the shopper who created the message. This is a foreign key that references the SHRFNBR column in the SHOPPER table.
dsdate	timestamp not null	Date and time message was sent.
dsadminres	timestamp	Date and time the administrator of the auction responded.
dsadmresid	integer	The message ID of the administrator’s response.
dsprfnbr	integer	Product reference number.
dsaurfnbr	bigint	Auction reference number.
dssubject	varchar(254)	Subject of the message.
dsmessage	long varchar	Content of the message.
dsadmact	char(4)	Action performed by the administrator on the message.
dsview	char(4)	Message view: P - Public NULL - Private
dscust1	integer	Merchant’s customization field 1.
dscust2	integer	Merchant’s customization field 2.
dscust3	integer	Merchant’s customization field 3.

AUCMEMREL(New): This table captures the relationship between RFQs and members. This purpose of this table is to enforce access control for individual products. Examples: who is allowed to modify an RFQ, view / Respond to an RFQ, approve an RFQ, etc.

Table 43. AUCMEMREL

Column Name	Data Type	Description
AMRFNBR	bigint not null	Primary key.
AMAURFNBR	integer (not null)	The product reference number. Primary key. Foreign key to AUCTINFO.AUREFNUM.
AMENTRYTYPE	small int (not null)	Indicates whether the entry points to an individual member, a user-defined group, an organization, or to the entire membership (public). This flag serves as a switch on which of the following three columns (if any) is to be used for this entry. At most one of the three columns should be on-null (all three can be null when the entry is Public / Un-restricted). 1: Member 2: Organization 3: Group 4: Public / Un-restricted
AMMEMBERID	integer	The ID of the member who has a role to play for this product. Refers to SHOPPER.SHRFNBR.
AMORGID	integer	The ID of the organization that has a role to play for this product. Refers to MERCHANT.MERFNBR.
AMGROUPID	integer	The ID of the group that has a role to play for this product. Refers to MEMGROUP.MGREFNUM.

AMROLE	char(4)	The role that this entry plays with respect to this product: A: Approver C: Creator E: Evaluator T: Targeted receiver S: Self-service receiver (found a broadcast RFQ through browse/search and expressed interest)
AMROLECREATOR	integer	Who created the member's role: could be the same as AMMEMBERID if members added themselves. Refers to SHOPPER.SHRFNBR
AMADDACTION	char(4)	The action that created this entry. Currently, this is an enumeration. C: Create D: Delegate T: Copy from template Other flags TBD
AMCREATETIME	TIMESTAMP	Time at which the entry was created.

BIDMEMREL (New): This table captures the relationship between responses and members. The purpose of this table is to enforce access control for individual products, for example who is allowed to modify an RFQ, view / respond to an RFQ, approve an RFQ, etc.

Table 44. BIDMEMREL

Column Name	Data Type	Description
BMRFNBR	bigint not null	Primary key.
BMBDRFNBR	char(36) (not null)	The response reference number. Primary Key. Foreign key to BIDTABLE.BDREFNUM.
BMENTRYTYPE	small int (not null)	Indicates whether the entry points to an individual member, a user-defined group, an organization, or to the entire membership (public). This flag serves as a switch on which of the following three columns (if any) is to be used for this entry. At most one of the three columns should be on-null (all three can be null when the entry is Public / Un-restricted). 1: Member 2: Organization 3: Group 4: Public / Un-restricted

BMMEMBERID	integer	The ID of the member who has a role to play for this product. Refers to SHOPPER.SHRFNBR.
BMORGID	integer	The ID of the organization that has a role to play for this product. Refers to MERCHANT.MERFNBR.
BMGROUPID	integer	The ID of the group that has a role to play for this product. Refers to MEMGROUP.MGREFNUM.
BMROLE	char(4)	The role that this entry plays with respect to this product: A: Approver C: Creator E:Evaluator T: Targeted receiver
BMROLECREATO R	integer	Who created the member's role; could be the same as AMMEMBERID if members added themselves. Refers to SHOPPER.SHRFNBR.
BMADDACTION	char(4)	The action that created this entry. Currently, this is an enumeration. C: Create D: Delegate T: Copy from template Other flags TBD
BMCREATETIME	TIMESTAMP	Time at which the entry was created

16.1.3 Example: buyer interaction

In this section we provide step-by-step directions for performing buyer-side interaction with the RFQ component of the negotiation subsystem. We will also use some window captures to illustrate our example.

The following are the buyer interactions with RFQs:

1. Create an RFQ

   An authorized buyer can create an RFQ by following these steps:

   a. The buyer logs on to the e-Marketplace.

   b. The buyer uses Manage Oofferings to navigate through the catalog to find the product for which an RFQ needs to be created. If the desired product does not exist, an RFQ can be created from scratch by selecting the **RFQ Offerings** subcategory.

c. The buyer selects the **Create RFQ Offering** link and is presented with the RFQ Offering window. If an existing product description is used for RFQ creation then the attributes of the product description are copied into the RFQ Offering window. The buyer may add new attributes to the RFQ using this window. When ready, the buyer clicks **Done**. This action will navigate the buyer to the Compose RFQ window.

d. The buyer completes the Compose RFQ window by providing information such as a name, a closing rule, and target list for the RFQ and selects **Submit**.

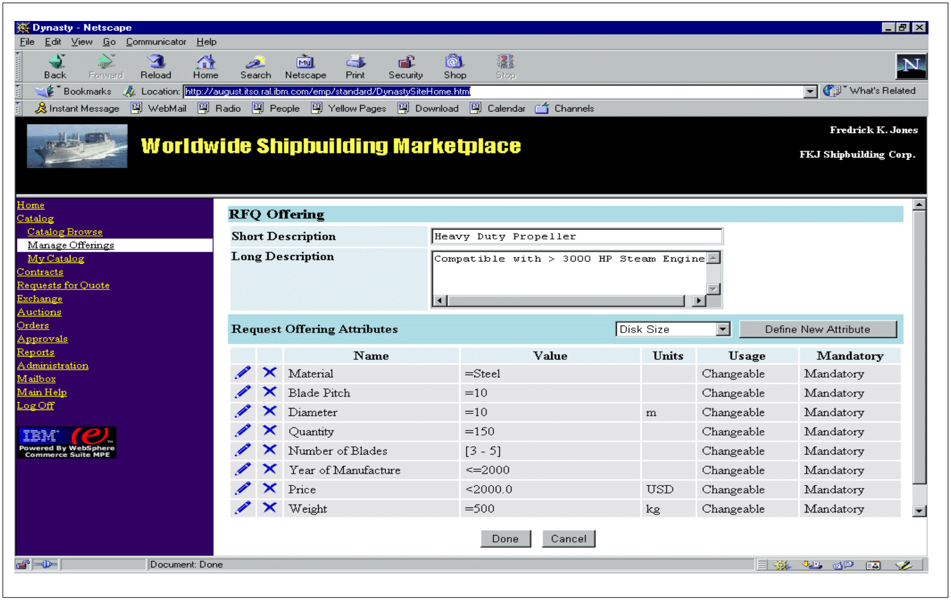
e. A list of all RFQs that belong to the current buyer is displayed indicating the status of each RFQ. The newly created RFQ at this time has an In Preparation status. The buyer clicks **Change Status**. This action navigates the buyer to the Change Status of RFQ window.

f. From this window the buyer may choose to Publish RFQ, Retract RFQ or View Details. To continue creating an RFQ, the buyer clicks **Publish RFQ**. The RFQ status changes to Future and a new button, Activate RFQ, appears allowing the buyer to activate the RFQ.

At this stage the RFQ is active and suppliers can create responses for this RFQ. We will cover the process of creating of a response in the next section.

Here’s an example, illustrated with windows.

We need to solicit suppliers to provide us with a price for heavy duty propellers. We navigate the Manage Offerings view of the catalog and select **Create RFQ Offering** for the Heavy Duty Propellers. Figure 121 on page 402 displays the attributes of the Heavy Duty Propeller product description. We will add a Price and Quantity attribute since the product description does not have one. We click **Done** in this window and go to the Compose RFQ window. Figure 122 shows the Compose RFQ window.


Dynasty - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Shop Stop
Bookmarks Location: http://august.itso.ral.ibm.com/emp/standard/DynastySiteHome.html What's Related
Instant Message WebMail Radio People Yellow Pages Download Calendar Channels
Worldwide Shipbuilding Marketplace
Fredrick K. Jones
FKJ Shipbuilding Corp.
Home
Catalog
Catalog Browse
Manage Offerings
My Catalog
Contracts
Requests for Quote
Exchange
Auctions
Orders
Approvals
Reports
Administration
Mailbox
Main Help
Log Off
IBM Powered By WebSphere Commerce Suite MPE
Document: Done

RFQ Offering

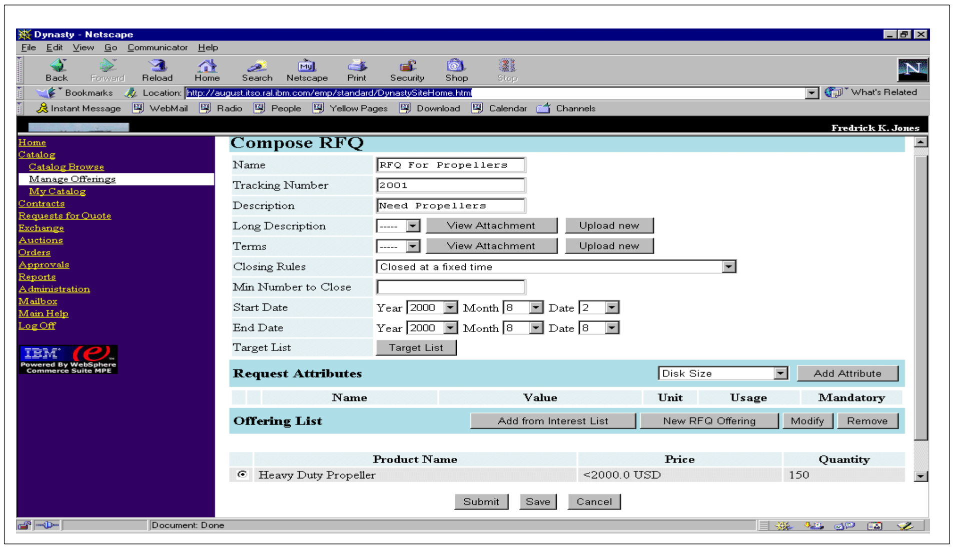
Short Description	Heavy Duty Propeller
Long Description	Compatible with > 3000 HP Steam Engine

Request Offering Attributes Disk Size | Define New Attribute

		Name	Value	Units	Usage	Mandatory
✎	✕	Material	=Steel		Changeable	Mandatory
✎	✕	Blade Pitch	=10		Changeable	Mandatory
✎	✕	Diameter	=10	m	Changeable	Mandatory
✎	✕	Quantity	=150		Changeable	Mandatory
✎	✕	Number of Blades	[3 - 5]		Changeable	Mandatory
✎	✕	Year of Manufacture	<=2000		Changeable	Mandatory
✎	✕	Price	<2000.0	USD	Changeable	Mandatory
✎	✕	Weight	=500	kg	Changeable	Mandatory

Done | Cancel

Figure 121. RFQ offering form


Dynasty - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Shop Stop
Bookmarks Location: http://august.itso.ral.ibm.com/emp/standard/DynastySiteHome.html What's Related
Instant Message WebMail Radio People Yellow Pages Download Calendar Channels
Fredrick K. Jones
Home
Catalog
Catalog Browse
Manage Offerings
My Catalog
Contracts
Requests for Quote
Exchange
Auctions
Orders
Approvals
Reports
Administration
Mailbox
Main Help
Log Off
IBM Powered By WebSphere Commerce Suite MPE
Document: Done

Compose RFQ

Name	RFQ For Propellers
Tracking Number	2001
Description	Need Propellers
Long Description	----- View Attachment Upload new
Terms	----- View Attachment Upload new
Closing Rules	Closed at a fixed time
Min Number to Close	
Start Date	Year 2000 Month 8 Date 2
End Date	Year 2000 Month 8 Date 8
Target List	Target List

Request Attributes Disk Size | Add Attribute

Name	Value	Unit	Usage	Mandatory

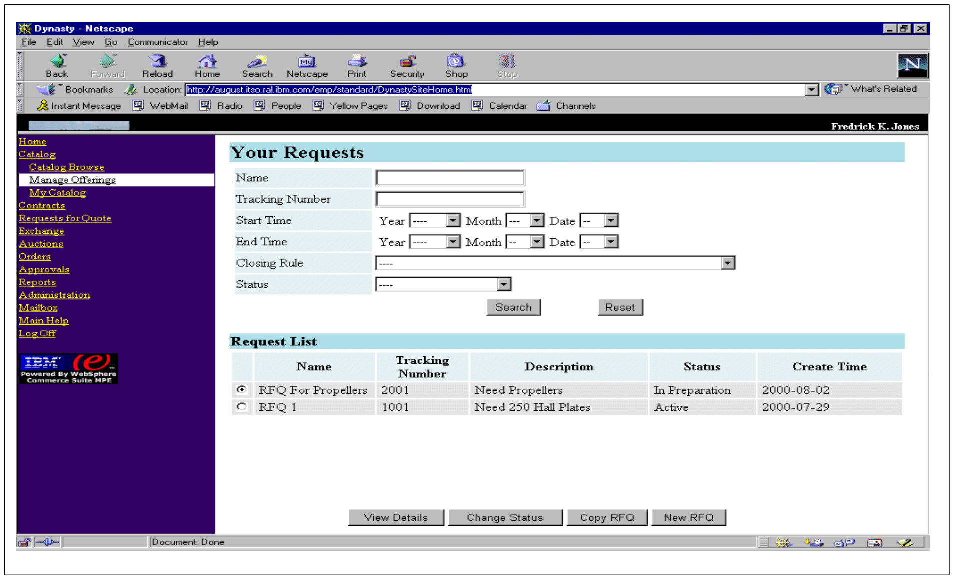
Offering List Add from Interest List | New RFQ Offering | Modify | Remove

	Product Name	Price	Quantity
●	Heavy Duty Propeller	<2000.0 USD	150

Submit | Save | Cancel

Figure 122. Compose RFQ

When we are satisfied with the content of our RFQ we select **Submit**. Figure 123 shows the list of RFQs we have created and their status. We select the newly created RFQ and click **Change Status**.


Dynasty - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Shop Stop
Bookmarks Location: http://august.itso.ral.ibm.com/emp/standard/DynastySiteHome.html
What's Related
Instant Message WebMail Radio People Yellow Pages Download Calendar Channels
Fredrick K. Jones
Home
Catalog
Catalog Browse
Manage Offerings
My Catalog
Contracts
Requests for Quote
Exchange
Auctions
Orders
Approvals
Reports
Administration
Mailbox
Main Help
Log Off
IBM Powered By WebSphere Commerce Suite MPE
Search
Reset
View Details
Change Status
Copy RFQ
New RFQ
Document: Done

Your Requests

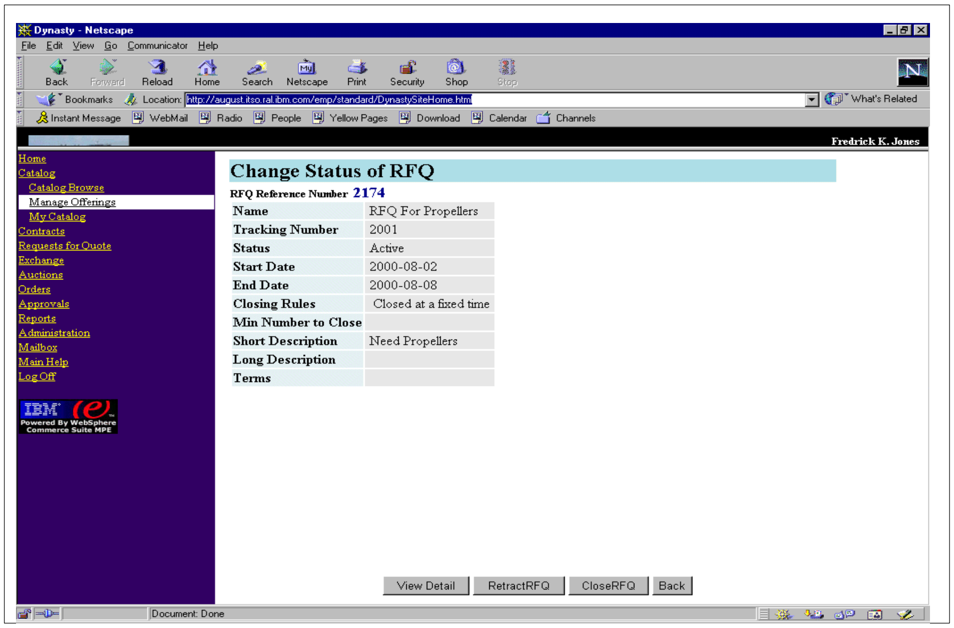
Name	
Tracking Number	
Start Time	Year ---- Month --- Date --
End Time	Year ---- Month --- Date --
Closing Rule	----
Status	----

Request List

	Name	Tracking Number	Description	Status	Create Time
●	RFQ For Propellers	2001	Need Propellers	In Preparation	2000-08-02
○	RFQ 1	1001	Need 250 Hall Plates	Active	2000-07-29

Figure 123. RFQ list

At the present time the RFQ has a status of In Preparation. We first select **Publish the RFQ** which puts the RFQ in Future status and then we select **Activate RFQ**. This puts the RFQ in Active status. Figure 124 shows the state of the RFQ after the above process is complete. At this point we have successfully created an RFQ in the e-Marketplace for the suppliers to provide responses.


Dynasty - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Shop Stop
Bookmarks Location: http://august.itso.ral.ibm.com/emp/standard/DynastySiteHome.html
What's Related
Instant Message WebMail Radio People Yellow Pages Download Calendar Channels
Fredrick K. Jones
Home
Catalog
Catalog Browse
Manage Offerings
My Catalog
Contracts
Requests for Quote
Exchange
Auctions
Orders
Approvals
Reports
Administration
Mailbox
Main Help
Log Off
View Detail
RetractRFQ
CloseRFQ
Back
Document: Done

Change Status of RFQ

RFQ Reference Number 2174

Name	RFQ For Propellers
Tracking Number	2001
Status	Active
Start Date	2000-08-02
End Date	2000-08-08
Closing Rules	Closed at a fixed time
Min Number to Close	
Short Description	Need Propellers
Long Description	
Terms	

Figure 124. RFQ status

2. Review responses and select winners

   An authorized buyer can select winners for an RFQ by following these steps:

   a. The buyer logs on to the e-Marketplace.

   b. The buyer uses the navigation frame to go to Request For Quote and then the My RFQs (buyer) link.

   c. The buyer searches and retrieves a list of RFQs.

   d. The buyer selects an RFQ and clicks **View Details**. The buyer clicks **View Responses** from the View RFQ Information window.

   e. The buyer is presented with the Responses List window. The buyer can select any response and view the details of it.

   f. As mentioned earlier an RFQ can be closed either by a buyer or by the scheduler. We will describe how a buyer performs this task.

   g. The buyer navigates to the Request For Quote window and then the My RFQs (buyer) link and from the list of RFQs selects the RFQ that needs to be closed. The buyer clicks **Change Status**.

h. From the Change Status of RFQ window the buyer selects **Close RFQ**. This action changes the status of the RFQ from Active to Closed.

i. The buyer clicks **Start Evaluation** to change the status of the RFQ from Closed to Evaluation In Progress”. At this stage no responses are accepted for this RFQ and the suppliers see the Being Evaluated status for their responses.

j. The buyer goes to the RFQs for Evaluation link and from the list, selects the RFQ which has a status of Being Evaluated. From the View RFQ Information the buyer clicks **Evaluate**. This action displays a list of all responses and allows the buyer to evaluate each response individually.

k. The buyer clicks **Evaluate** for each response. Buyers can select either Accept, Reject, Accept All or Reject All. The last two options apply to the responses to RFQs with more than one product in them. The buyer can accept responses to some products and not to others. The buyer also can select multiple winners for an RFQ by accepting more than one response.

l. To complete the process the buyer clicks **Accept** on one or more or the responses.

m. Now the buyer needs to change the status of the RFQ to complete.

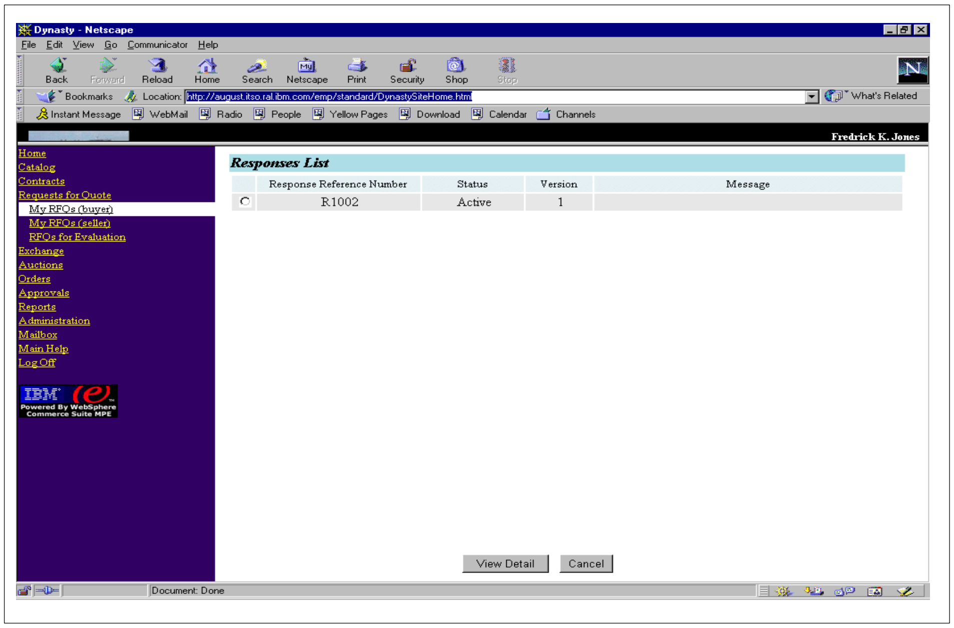
n. The buyer navigates to the Request For Quote and then My RFQs (buyer) link and from the list of RFQs selects the RFQ that has just been evaluated. The buyer clicks **Change Status**.

o. From the Change Status of RFQ window, the buyer clicks **Select Results**. This action puts the RFQ into RFQ Complete state. This action also changes the status of responses from Accepted to Won.

p. To verify the successful completion of this process, the buyer can navigate to the Responses List window and ensure that the accepted responses have a Won status.

   This completes the process of selecting a winner. At this point the buyers can proceed to place orders or start the creation of a contract.

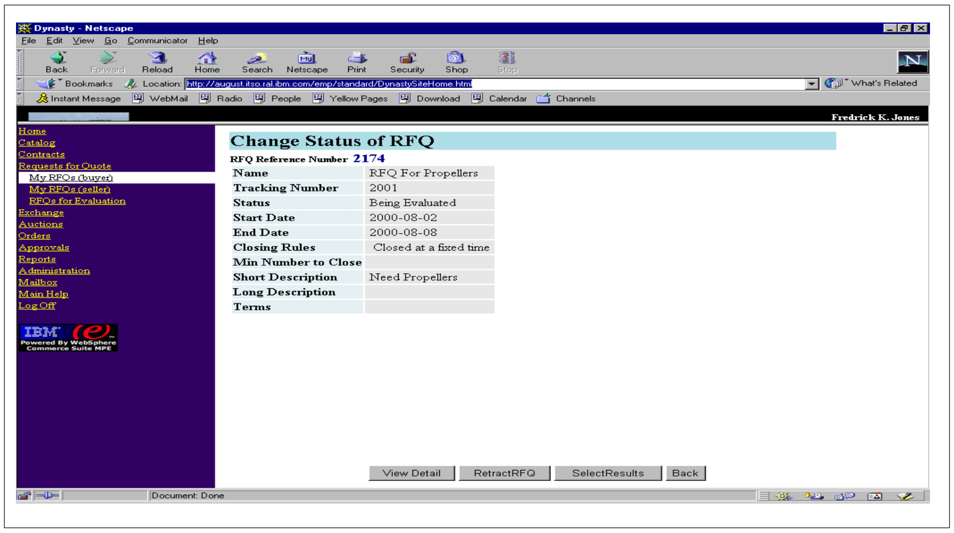
We will now continue with our example we started earlier. We are assuming that a supplier has placed a response to our RFQ. We follow the above instruction to view and select a winning response. Figure 135 shows the Responses List window prior to selecting a winner. We review the response and decide that we don’t need any more responses and we want to close the RFQ. Figure 126 shows the status of the RFQ after we have closed the RFQ. Next we evaluate the response and select it as the winner.


Dynasty - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Shop Stop
Bookmarks Location: http://august.itso.ral.ibm.com/emp/standard/DynastySiteHome.html What's Related
Instant Message WebMail Radio People Yellow Pages Download Calendar Channels
Fredrick K. Jones
Home
Catalog
Contracts
Requests for Quote
My RFQs (buyer)
My RFQs (seller)
RFQs for Evaluation
Exchange
Auctions
Orders
Approvals
Reports
Administration
Mailbox
Main Help
Log Off
View Detail
Cancel
Document: Done

Responses List

	Response Reference Number	Status	Version	Message
○	R1002	Active	1	

Figure 125. Response list prior to evaluation


Dynasty - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Shop Stop
Bookmarks Location: http://august.itso.ral.ibm.com/emp/standard/DynastySiteHome.html What's Related
Instant Message WebMail Radio People Yellow Pages Download Calendar Channels
Fredrick K. Jones
Home
Catalog
Contracts
Requests for Quote
My RFQs (buyer)
My RFQs (seller)
RFQs for Evaluation
Exchange
Auctions
Orders
Approvals
Reports
Administration
Mailbox
Main Help
Log Off
View Detail
RetractRFQ
SelectResults
Back
Document: Done

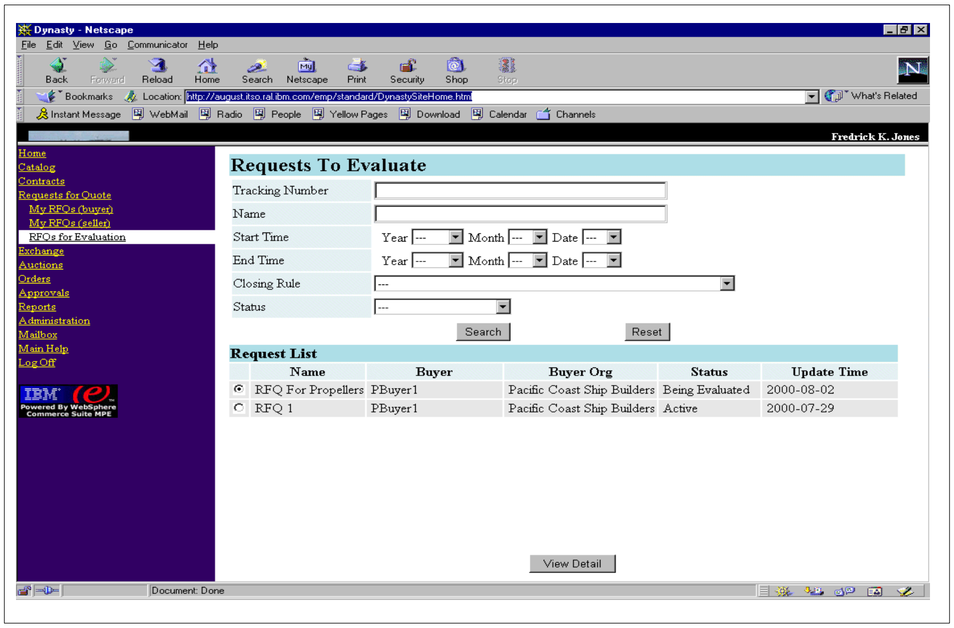
Change Status of RFQ

RFQ Reference Number 2174

Name	RFQ For Propellers
Tracking Number	2001
Status	Being Evaluated
Start Date	2000-08-02
End Date	2000-08-08
Closing Rules	Closed at a fixed time
Min Number to Close	
Short Description	Need Propellers
Long Description	
Terms	

Figure 126. RFQ status after issuing close RFQ

Figure 127 shows the list of RFQ and which ones are in the process of evaluation. We select our RFQ and click **View Details**.


Dynasty - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Shop Stop
Bookmarks Location: http://august.itso.ral.ibm.com/emp/standard/DynastySiteHome.html What's Related
Instant Message WebMail Radio People Yellow Pages Download Calendar Channels
Fredrick K. Jones
Home
Catalog
Contracts
Requests for Quote
My RFQs (buyer)
My RFQs (seller)
RFQs for Evaluation
Exchange
Auctions
Orders
Approvals
Reports
Administration
Mailbox
Main Help
Log Off
Search
Reset
View Detail
Document: Done

Requests To Evaluate

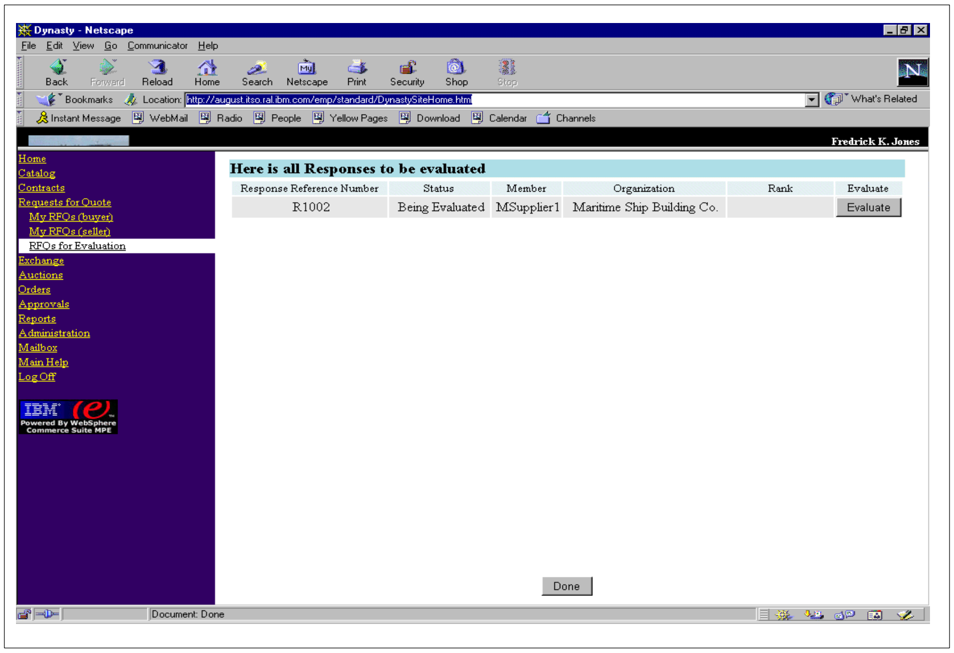
Tracking Number	
Name	
Start Time	Year --- Month --- Date ---
End Time	Year --- Month --- Date ---
Closing Rule	---
Status	---

Request List

	Name	Buyer	Buyer Org	Status	Update Time
●	RFQ For Propellers	PBuyer1	Pacific Coast Ship Builders	Being Evaluated	2000-08-02
○	RFQ 1	PBuyer1	Pacific Coast Ship Builders	Active	2000-07-29

Figure 127. List of RFQs for evaluation

Figure 128 shows all the responses we have received for our RFQ, in this case only one. Note that if there were more than one responses each one would have an Evaluate button at the far right. If there are multiple responses, then from this window we can select each one and accept or reject them as we desire. Figure 129 shows the response evaluation window, where we can accept or reject all or parts of a response. .


Dynasty - Netscape

File Edit View Go Communicator Help

Back Forward Reload Home Search Netscape Print Security Shop Stop

Bookmarks Location: http://august.itso.ral.ibm.com/emp/standard/DynastySiteHome.html What's Related

Instant Message WebMail Radio People Yellow Pages Download Calendar Channels

Fredrick K. Jones

Home
Catalog
Contracts
Requests for Quote
My RFQs (buyer)
My RFQs (seller)
RFQs for Evaluation
Exchange
Auctions
Orders
Approvals
Reports
Administration
Mailbox
Main Help
Log Off

Here is all Responses to be evaluated

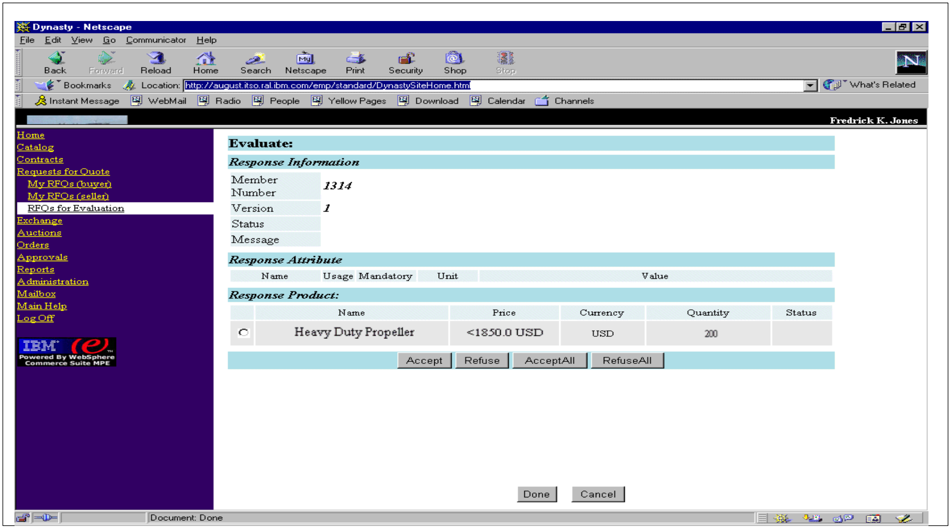
Response Reference Number	Status	Member	Organization	Rank	Evaluate
R1002	Being Evaluated	MSupplier1	Maritime Ship Building Co.		Evaluate

Done

Document: Done

Figure 128. Responses to be evaluated

We click **Evaluate** which displays the response evaluation window as shown in Figure 129. From this window we click **Accept** to accept this response.


Dynasty - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Shop Stop
Bookmarks Location: http://august.itso.ral.ibm.com/emp/standard/DynastySiteHome.html What's Related
Instant Message WebMail Radio People Yellow Pages Download Calendar Channels
Fredrick K. Jones
Home
Catalog
Contracts
Requests for Quote
My RFQs (buyer)
My RFQs (seller)
RFQs for Evaluation
Exchange
Auctions
Orders
Approvals
Reports
Administration
Mailbox
Main Help
Log Off
Powered By WebSphere Commerce Suite MPE
Accept Refuse AcceptAll RefuseAll
Done Cancel
Document: Done

Evaluate:

Response Information

Member Number	*1314*
Version	*1*
Status	
Message	

Response Attribute

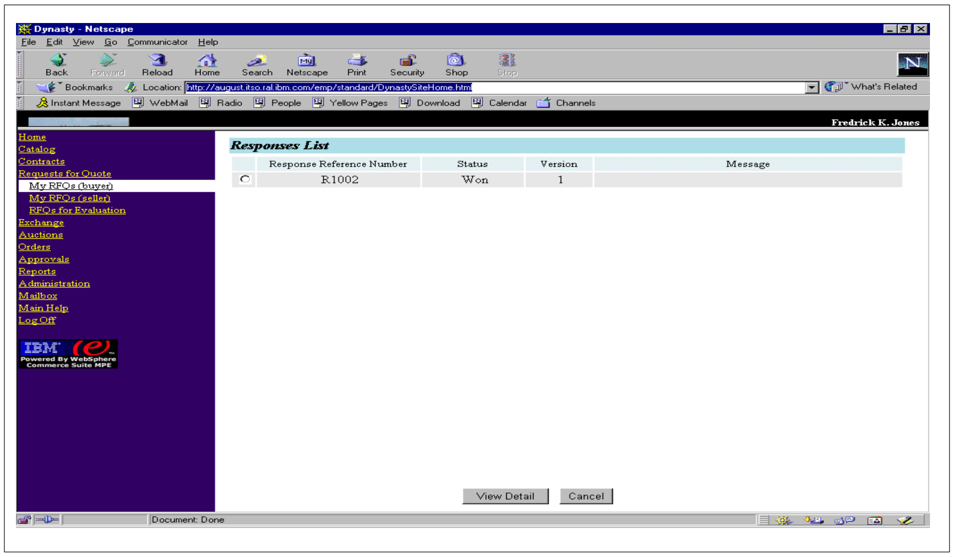
Name	Usage	Mandatory	Unit	Value

Response Product:

	Name	Price	Currency	Quantity	Status
○	Heavy Duty Propeller	<1850.0 USD	USD	200	

Figure 129. Response evaluation window

Now that we have accepted the response we need to change the status of the RFQ to complete. We go to My RFQs (buyer) and select the RFQ and click the **Change Status**. From the status window we complete the RFQ by clicking **Select Results**. This will change the status of RFQ to Complete and the status of the response to Won. Figure 130 shows the winning response to our RFQ.


Dynasty - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Shop Stop
Bookmarks Location: http://august.itso.ral.ibm.com/emp/standard/DynastySiteHome.html
What's Related
Instant Message WebMail Radio People Yellow Pages Download Calendar Channels
Fredrick K. Jones
Home
Catalog
Contracts
Requests for Quote
My RFQs (buyer)
My RFQs (seller)
RFQs for Evaluation
Exchange
Auctions
Orders
Approvals
Reports
Administration
Mailbox
Main Help
Log Off
Powered By WebSphere Commerce Suite MPE
View Detail
Cancel
Document: Done

Responses List

	Response Reference Number	Status	Version	Message
○	R1002	Won	1	

Figure 130. Winning response

3. Place an order from an RFQ

   To place an order from a winning response, the buyer takes the following steps:

   a. The buyer logs on to the e-Marketplace.

   b. The buyer uses the navigation frame to go to the Request For Quote window and then to the My RFQs (buyer) link.

   c. The buyer searches and retrieve a list of RFQs.

   d. The buyer selects a completed RFQ and clicks **View Details**.

   e. The buyer clicks **View Responses** from the View RFQ Information window.

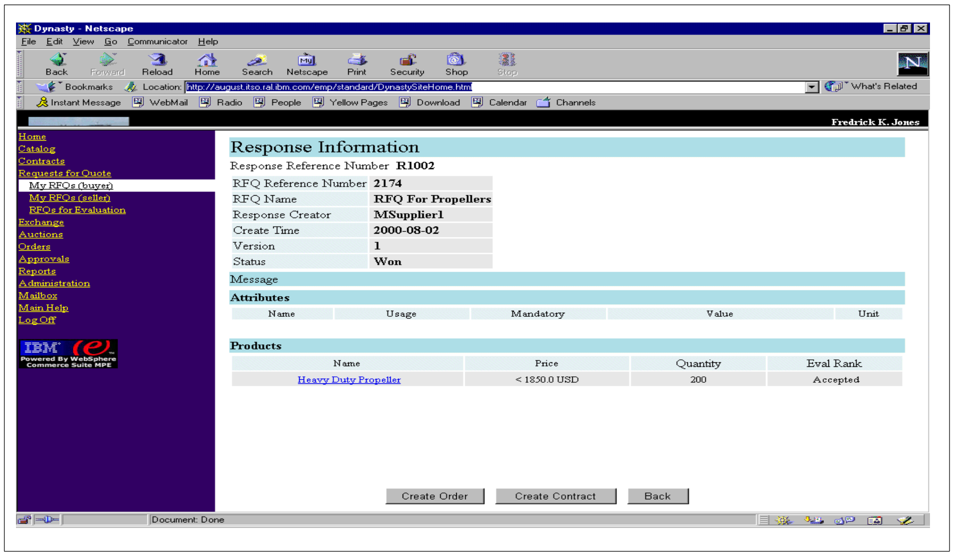
   f. The buyer is presented with the Responses List window containing the winning responses.

   g. The buyer selects a winning response and views its details. From the Response Information window the buyer can either create an order or create a contract.

   h. The buyer clicks **Create Order** and is presented with the Create Order for RFQ window. The buyer specifies the quantity and clicks **Place Order**.

   i. An order is placed and an order confirmation window is shown.

Now we will use the winning response from our example to place an order. Figure 130 shows the list of responses. We select the response and click **View Details**. Figure 131 shows the response information.



Response Information

Response Reference Number **R1002**

RFQ Reference Number	**2174**
RFQ Name	**RFQ For Propellers**
Response Creator	**MSupplier1**
Create Time	**2000-08-02**
Version	**1**
Status	**Won**

Message

Attributes

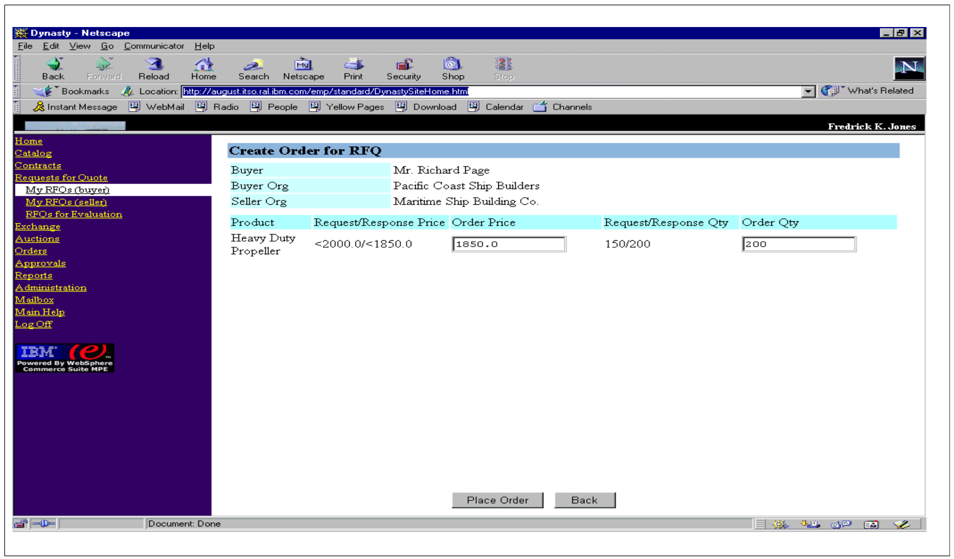
Name	Usage	Mandatory	Value	Unit

Products

Name	Price	Quantity	Eval Rank
Heavy Duty Propeller	< 1850.0 USD	200	Accepted

Figure 131. Response details

We click **Create Order** and are presented with the window shown in Figure 132. We can modify the quantity and click **Place Order** to complete the buying process. An order confirmation is provided at the end of the process.


Dynasty - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Shop Stop
Bookmarks Location: http://august.itso.ral.ibm.com/emp/standard/DynastySiteHome.html
What's Related
Instant Message WebMail Radio People Yellow Pages Download Calendar Channels
Fredrick K. Jones
Home
Catalog
Contracts
Requests for Quote
My RFQs (buyer)
My RFQs (seller)
RFQs for Evaluation
Exchange
Auctions
Orders
Approvals
Reports
Administration
Mailbox
Main Help
Log Off
Powered By WebSphere Commerce Suite MPE
Place Order
Back
Document: Done

Create Order for RFQ

Buyer	Mr. Richard Page
Buyer Org	Pacific Coast Ship Builders
Seller Org	Maritime Ship Building Co.

Product	Request/Response Price	Order Price	Request/Response Qty	Order Qty
Heavy Duty Propeller	<2000.0/<1850.0	1850.0	150/200	200

Figure 132. Order for an RFQ

4. Create a contract from an RFQ

To create a contract from a winning response, the buyer takes the following steps:

a. The buyer logs on to the e-Marketplace.

b. The buyer uses the navigation frame to go to Request For Quote and then to My RFQs (buyer) link.

c. The buyer searches and retrieves a list of RFQs.

d. The buyer selects an RFQ and clicks **View Details**.

e. The buyer clicks **View Responses** from the View RFQ Information window.

f. The buyer is presented with the Responses List window containing the winning responses.

g. The buyer selects a winning response and views the details of it. From the Response Information window, the buyer can either create an order or create a contract.

Note: When we wrote this redbook, the contract creation process had not been implemented, so we are unable to test or describe this in detail.

16.1.4 Example: supplier interaction

Supplier interaction with the RFQs is limited only to providing responses if they see fit. Here we explain how a response is created and use an example to visually demonstrate this process.

1. Create a Response

   An authorized supplier can create a response to an RFQ by following these steps:

   a. The supplier logs on to the e-Marketplace.

   b. From the Navigation frame, the supplier selects **Request For Quote** and the My RFQs (Seller) link. This presents the supplier with the RFQ search and list window. The supplier can search and obtain a list of all available RFQs in the system.

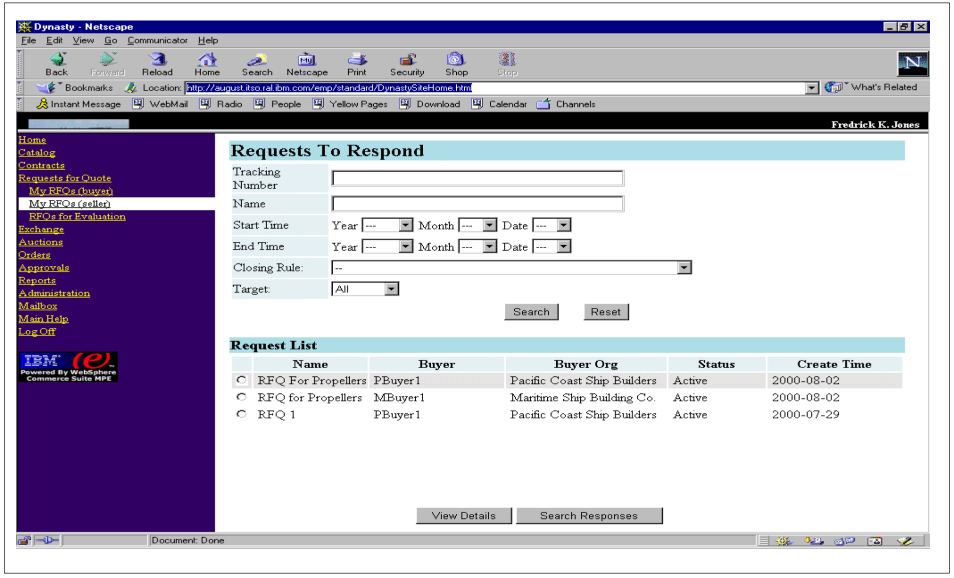
   c. The supplier selects an RFQ and clicks **View Details**.

   d. From the View RFQ Information window, the supplier may choose to download the RFQ file to prepare a response or click **Create Response** to continue.

   e. Create Response navigates the supplier to the Compose Response window. The supplier provides a bid for the requested products and clicks **Done**. This action navigates the supplier to the Response Information window.

   f. At this time the response has a status of In Preparation. From this window the supplier can modify, retract or submit a response. To continue, the supplier clicks **Submit Response**. This will activate the response in the system.

      The following is an example on how a supplier can respond to an RFQ. We will list all RFQs and select the one created in the buyer interaction example. Figure 133 on page 414 shows the list of all available RFQs in the system that our supplier is allowed to work with. We select the RFQ and click **View Details** and then choose to prepare a response for it.


Dynasty - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Shop Stop
Bookmarks Location: http://august.itso.ral.ibm.com/emp/standard/DynastySiteHome.html What's Related
Instant Message WebMail Radio People Yellow Pages Download Calendar Channels
Fredrick K. Jones
Home
Catalog
Contracts
Requests for Quote
My RFQs (buyer)
My RFQs (seller)
RFQs for Evaluation
Exchange
Auctions
Orders
Approvals
Reports
Administration
Mailbox
Main Help
Log Off
Document: Done

Requests To Respond

Tracking Number	
Name	
Start Time	Year --- Month --- Date ---
End Time	Year --- Month --- Date ---
Closing Rule:	--
Target:	All

Search Reset

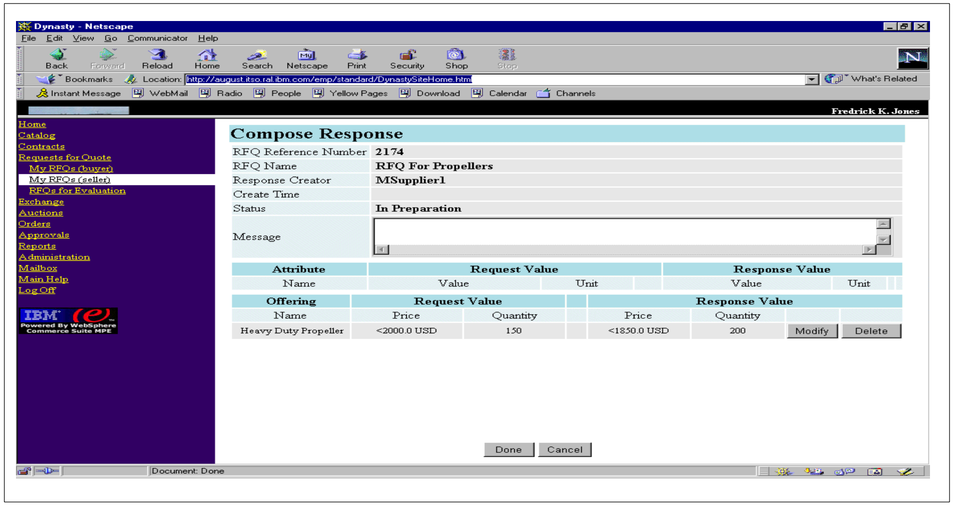
Request List

	Name	Buyer	Buyer Org	Status	Create Time
○	RFQ For Propellers	PBuyer1	Pacific Coast Ship Builders	Active	2000-08-02
○	RFQ for Propellers	MBuyer1	Maritime Ship Building Co.	Active	2000-08-02
○	RFQ 1	PBuyer1	Pacific Coast Ship Builders	Active	2000-07-29

View Details Search Responses

Figure 133. RFQ list for supplier

Figure 134 on page 414 shows the response we created. We then click **Done** and then from the Response Information window we submit the response to activate it in the system.


Dynasty - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Shop Stop
Bookmarks Location: http://august.itso.ral.ibm.com/emp/standard/DynastySiteHome.html What's Related
Instant Message WebMail Radio People Yellow Pages Download Calendar Channels
Fredrick K. Jones
Home
Catalog
Contracts
Requests for Quote
My RFQs (buyer)
My RFQs (seller)
RFQs for Evaluation
Exchange
Auctions
Orders
Approvals
Reports
Administration
Mailbox
Main Help
Log Off
Document: Done

Compose Response

RFQ Reference Number	**2174**
RFQ Name	**RFQ For Propellers**
Response Creator	**MSupplier1**
Create Time	
Status	**In Preparation**
Message	

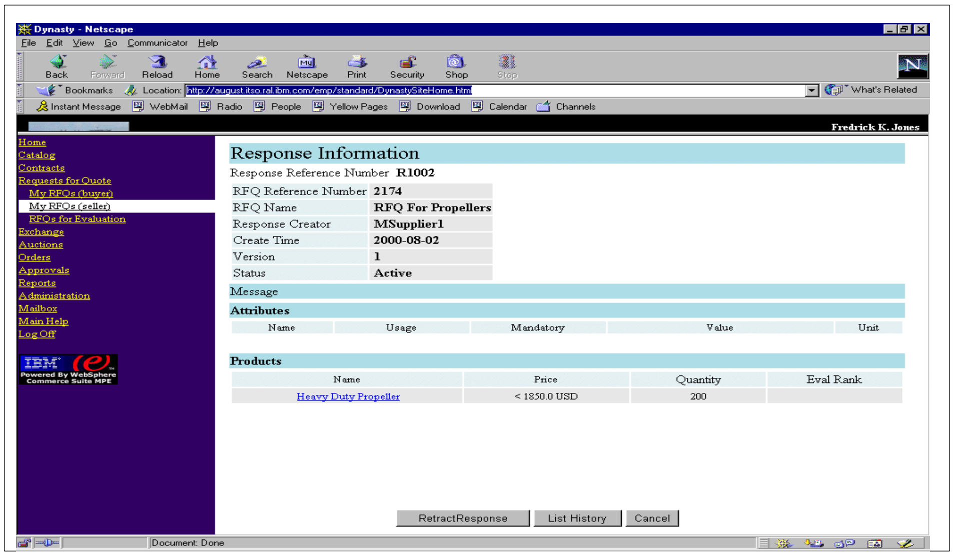
Attribute	Request Value		Response Value	
Name	Value	Unit	Value	Unit

Offering	Request Value		Response Value			
Name	Price	Quantity	Price	Quantity		
Heavy Duty Propeller	<2000.0 USD	150	<1850.0 USD	200	Modify	Delete

Done Cancel

Figure 134. RFQ response

Figure 135 on page 415 shows the final state of our response. The buyer now can view this response and evaluate it when the RFQ is closed. While the RFQ is still Active the supplier can modify this response or retract it.


Dynasty - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Shop Stop
Bookmarks Location: http://august.itso.ral.ibm.com/emp/standard/DynastySiteHome.html What's Related
Instant Message WebMail Radio People Yellow Pages Download Calendar Channels
Fredrick K. Jones
Home
Catalog
Contracts
Requests for Quote
My RFQs (buyer)
My RFQs (seller)
RFQs for Evaluation
Exchange
Auctions
Orders
Approvals
Reports
Administration
Mailbox
Main Help
Log Off
RetractResponse
List History
Cancel
Document: Done

Response Information

Response Reference Number **R1002**

RFQ Reference Number	**2174**
RFQ Name	**RFQ For Propellers**
Response Creator	**MSupplier1**
Create Time	**2000-08-02**
Version	**1**
Status	**Active**

Message

Attributes

Name	Usage	Mandatory	Value	Unit

Products

Name	Price	Quantity	Eval Rank
Heavy Duty Propeller	< 1850.0 USD	200	

Figure 135. Response submit screen

16.1.5 Interaction with other components and subsystems

This section describes the interaction between the RFQ component of the negotiation subsystem and other subsystems.

16.1.5.1 Access Control

The following table lists the access control rules for the various RFQ / response commands.

Table 45. Access control rules

Command	Business role	Object	Object Role
CreateRFQCmd	Buyer		
ModifyRFQCmd	Buyer	RFQ	Creator
ChangeRFQVersionCmd	Buyer	RFQ	Creator
CreateRFQTemplateCmd	Buyer	RFQ	Creator

PublishRFQCmd	Buyer	RFQ	Creator
ActivateRFQCmd	Buyer	RFQ	Creator
CloseRFQCmd	Buyer	RFQ	Creator
StartEvaluationRFQCmd	Buyer	RFQ	Creator
EvaluateResponsesCmd	Evaluator / Buyer	RFQ	Evaluator
SelectResultsCmd	Buyer	RFQ	Creator
RetractRFQCmd	Buyer	RFQ	Creator
CreateResponseCmd	Seller	RFQ	Targeted receiver / Self-service receiver
ModifyResponseCmd	Seller	RFQ	Targetted receiver / Self-service receiver
		Response	Creator
SumbitResponseCmd	Seller	RFQ	Targetted receiver / Self-service receiver
		Response	Creator
RetractResponseCmd	Seller	RFQ	Targetted receiver / Self-service receiver
		Response	Creator
ChangeResponseVersion	Seller	RFQ	Targetted receiver / Self-service receiver
		Response	Creator
CreateContract	Buyer	RFQ	Creator
		Response	Buyer
CreateOrder	Buyer	RFQ	Creator
		Response	Buyer
View / list / Search RFQ ICs	Buyer	RFQ	Creator / Approver / Evaluator
View / list / Search RFQ ICs	Seller	RFQ	Targetted receiver / Self-service receiver

View / list / Search Response ICs	Buyer	RFQ	Creator / Approver / Evaluator
		Response	Buyer / Evaluator
View / list / Search Response ICs	Seller	RFQ	Targetted receiver / Self-service receiver
		Response	Creator / Approver

For ViewRFQ and ViewResponse, the access control should be determined by the policy of each organization.

The roles used by access control in the RFQ subsystem should be the same as the roles used by flex flow.

16.1.5.2 Approvals

The following commands require approval.

- PublishRFQ
- SubminResponse
- ChangeRFQVersionChangeResponseVersion
- SelectResults

The following interaction controllers or commands are needed to support the approval process. The corresponding JSP pages will be provided.

- ViewRFQForApproval
- ViewResponseForApproval

We will use the RFQApprovalObjectHandle object to store the next state and the original state of each command that needs approval, so we don’t need to add intermediate states in the RFQ/Response state machines. An intermediate state will be stored in the database tables. From this intermediate state, no action will be allowed without approval.

16.1.5.3 Flex flow

The RFQ subsystem will rely on flex flow to complete the following tasks:

- Getting allowed actions from the current state and getting to the next state
- Sending messages between RFQ and the response state machines

The RFQ subsystem will not use flex flow to maintain the current state. The current state will be maintained in the database tables.

The RFQ subsystem will not use flex flow to implement access control. The roles used by flex flow should be the same as the roles used by the access control subsystem.

16.1.5.4 Catalog

The following interaction controllers will be registered for the interaction from the catalog to the RFQ subsystem:

- ViewRFQ
- CreateRFQFromCart

The catalog subsystem is responsible for providing the following functions that support the interaction from the RFQ subsystem to the catalog:

- Creating RFQ offerings/offering bundles and modifying offering attributes
- Creating RFQ from catalog and adding offerings from catalog to RFQ via RFQCart

16.1.5.5 Order

The RFQ subsystem will use the order management subsystem to convert win responses to orders. For each winning response, one order can be created.

16.1.5.6 Contract

All data from winning responses that matches the attributes used in a contract will be copied into the draft contract creating form. All fields can be edited. Before the contract is created, new contract offerings will be created in the catalog from RFQ offerings and response product attributes.

16.1.5.7 Legacy WCS

The RFQ subsystem has made some changes to the existing WCS 4.1 tables AUCTINFO and BIDTABLE for sharing database schemas with the auctions subsystem. The RFQ subsystem will use the existing MESSAGES and DISCUSSION tables in the auctions subsystem to complete notification and discussion functions.

16.2 Auctions

Auctions are an alternate method of performing transactions in an e-Marketplace. In order to manage inventory or sell excess products suppliers create auctions. Buyers in an e-Marketplace view and bid on available auctions.

The auction component of the negotiation subsystem is a carryover from the WebSphere Commerce Suite 4.1 (WCS 4.1) with very minor changes. The changes are:

- Database changes

  AUCTINFO table shown in Figure 34 on page 382 and BIDTABLE shown in Figure 37 on page 388 have been modified to allow RFQs (reverse auctions).

- Access control changes

  The WCS 4.1 auctions code for createAuction and ModifyAuction has been changed to add access control rows for each of the catalog items (the auction offering) that are put in an auction. All auctions are public. Since catalog viewing uses access control, a generic “open” access control row has been added for every auction offering.

- Catalog

  The creation of an auction now creates an auction offering in the catalog. The inventory management of catalog has been modified so that it is tied to the offering. The catalog view macros do not change after an auction closes, since the auction is a product offering, and therefore does not have a pre-auction view. A new offering view command (`AuctionOfferingView`) is created, and all auction view commands are routed through this command. The catalog will use this command with appropriate actions such as action=ViewProductDetails, action=ViewAuctionRules, action=ViewAllBids, action=CreateNewBid, action=CreateNewOrderBid, action=ViewMyBids.

- Order processing

  The current `completeorder` command creates entries in the order table, and follows the catalog order processing of WCS 4.1. The auction order processing will continue to work in the same fashion for the WebSphere Commerce Suite, Marketplace Edition for AIX also.

- Approval workflow

  CreateAuction and ModifyAuction are approvable commands in the WebSphere Commerce Suite, Marketplace Edition for AIX.

16.3 Exchanges

The exchange subsystem represents another form of negotiation-based purchasing available in the Marketplace Edition. Its basic functionality is to provide a matching service that matches the products, quantities and maximum price requested by a buyer to products, inventories and actual prices offered by a supplier.

This section explains many of the components and theory of the exchange subsystem. We do not provide a working example due to the fact that, at the time of writing, many of the exchange features were unavailable.

16.3.1 High-level overview

There are three primary components to the exchange subsystem:

- Trading

  The trading component controls the entities interacting with the exchange subsystem. It is responsible for such tasks as capturing and processing of bid and offers, initiating the matching algorithms, searching and displaying the orderbook, notifying participants of complete matches ,and submitting completed matches.

- Orderbook

  The orderbook component is responsible for persisting the bids and offers placed by participants, and manages the submission of multiple bids for a single resource and resource bundles. Basically, buyer's bids and seller's offers are placed in the orderbook.

- Matching

  The matching component provides the logic to match bids and offers via a number of *matching algorithms*. It is also responsible for the dynamic pricing model for the traded products, which can fluctuate based on demand.

The interactions between these components is depicted in Figure 136. We can see from this diagram that the interfaces to the buyer and the suppliers are in a variety of formats, such as XML and HTML. These interactions can be real-time or batch-mode operations.

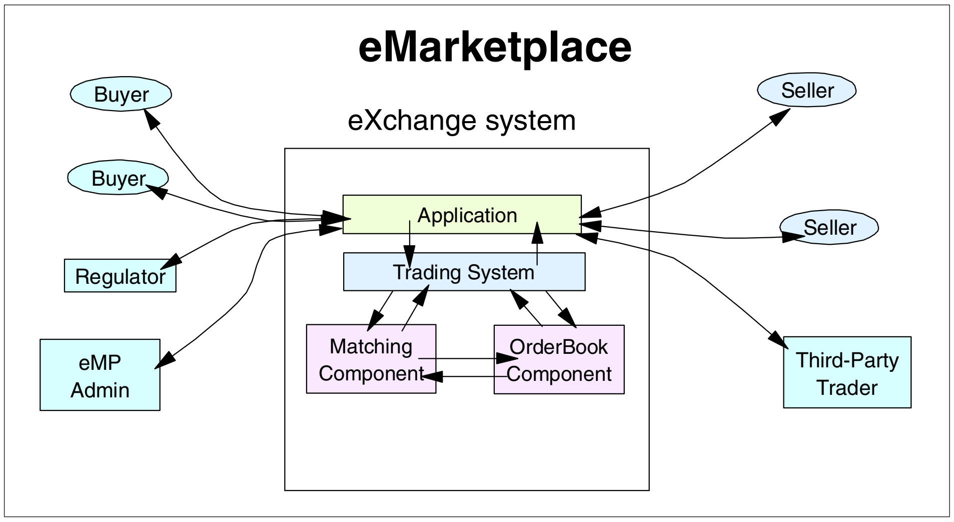


Figure 136. Interactions between the exchange components

When a trade occurs as a result of a successful match, the trade is processed by the order subsystem.

The functionality provided by the exchange subsystem is founded on the following observations on how trading occurs in e-Marketplaces:

- Liquidity

  Liquidity refers to e-Marketplaces that are considered to be "liquid" or dynamic. Examples of these type of e-Marketplaces are financial exchanges, and commodity markets. In order to support these markets, the exchange process must offer real-time matching in addition to manually triggered or event triggered matching processes.

- Anonymity

  Some e-Marketplaces require that bids and offers are submitted anonymously to ensure fairness and equity among the trading participants. This is beneficial, for example, to small businesses who want to compete with larger organizations in the hub, but who don't want to be excluded because of their size.

- Industry-specific e-Marketplace

  This type of e-Marketplace involves trading models centered on specific, and vastly different, industry sectors. For example, the e-Marketplace trading requirements for a financial e-Marketplace are notably different to those required by a steel-trading e-Marketplace.

- Regulation

  Often, stringent regulations must be adhered to in e-Marketplaces, such as those centered on transportation or energy, for example. It is critical that the matching process used in an e-Marketplace exchange enforce any regulatory requirements.

The exchange subsystem in the Marketplace Edition has been implemented with these observations in mind.

16.3.2 Low-level design

In this section, we discuss the attributes of the exchange subsystem design, associated Java programming elements and database tables.

16.3.2.1 Matching component

The matching component relies on matching algorithms to match a buyer's bids to a seller's offers. These matching algorithms can be classified into the buyers-sellers relational groups shown in Table 46.

Table 46. Matching algorithm

Matching type	Description
One to one	Provides for a one-buyer to one-seller trading model.
One to many	Provides for many sellers to one-buyer model where sellers wish only to be matched with a single buyer.
Many to one	Allows only one seller to be matched with many buyers.
Many to many	Provides for multiple buyers to be matched with multiple sellers.

The triggering of matching algorithms is configurable and may be based on timed events, or positions (that isthe submission of a bid or ask), or by manual intervention.

16.3.2.2 Price determination

Because of the need for a dynamic pricing model in the exchange, a number of algorithms are used to “discover” the price for a particular product. The final or discovered price for a product is commonly referred to as its *clearing price*, a term you should be familiar with. This is required because in addition to a fixed base-price, an exchange should offer dynamic pricing based on product supply and demand. The algorithm classifications are shown in Table 47.

Table 47. Pricing algorithms for specific trading models

Pricing algorithm classification	Description
Continuous double action	This pricing model is centered on the highest-paying buyer being matched with the lowest-cost supplier. The clearing price may be set based on the lowest seller price, the highest bid by the buyer or the median of these numbers. For example, if the buyer bids $100, and the seller base-price is $150, the clearing price outcome accepted may be the $100 from the buyer or the agreed median of $125. Once this price is arrived at, the process reiterates - the next lowest paying buyer is matched to the next lowest cost supplier and so on.
Walrasian Tatonnement	This pricing algorithm involves an iterative process where bids are gathered and ranked and the price is set based on demand. As more bids are received, the price fluctuates. In this model, the clearing price is arrived at when the product demand is equal to supply (although other price points can be arrived at). For further reading on Leon Walras' theories on tatonnement processes and his other economic theories and works, visit: `http://www.econ.jhu.edu/People/Fonseca/walras/hardy.htm`
Call market pricing	Call market pricing is based on the private submittal of bids and offers to the central exchange. The clearing price is derived through computation of the price points where demand and supply meet based on predefined objectives such as maximizing the number of trades.
Profit-driven pricing	Similar in theory to call market pricing; however, the object is to maximize the profits of suppliers.
Traditional auction	In traditional auctions the demand and supply are gathered centrally and the clearing prices established. Auctions are a supplier-driven exchange metaphor.

Some of these pricing algorithms are based on complex theories that you do not need to understand fully, however, you should aim to be familiar with the basic principals of each.

16.3.2.3 Positions

Positions, also known as *trading positions*, represent an offer to buy or sell products and are analogous to *bids* and *asks*, if you are more familiar with that terminology. Consequently, a trading position contains some important information, namely:

- Its creator.
- The product that is the target of the exchange.
- Additional attributes specified as name/value pairs.

The Marketplace Edition allows multiple positions of different types to be active for a single product. This enables the product to be concurrently offered to the e-Marketplace under auction, exchange, and fixed price.

Positions can be created by buyers and suppliers using a number of methods, the most common being through browsing the product catalog and selecting a product template to base the position on. This procedure is outlined in 16.3.4, “Supplier interaction” on page 437 and 16.3.5, “Buyer interaction” on page 438. It is also possible to create more complex positions based on compound (or bundled) product offerings. Figure 137 shows the relationships between a trading position, its associated attributes and the attributes of the e-Marketplace product.

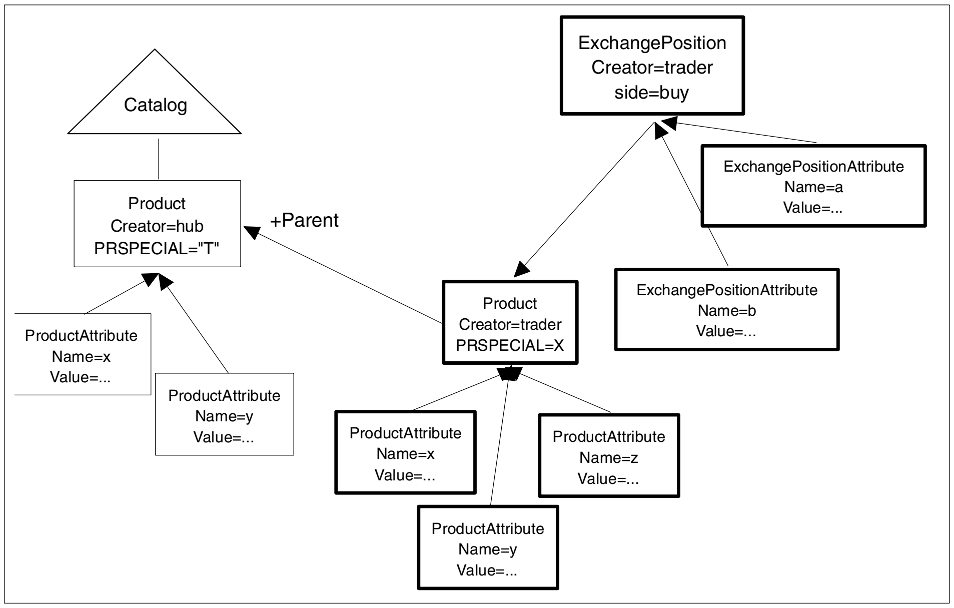


Figure 137. Relationships between a trading position and the product catalog

16.3.2.4 Trading posts

The term trading post refers to the actual processes of conducting the matching logic in the exchange. There can be one or more trading posts within the exchange, each configured with different matching algorithms. However, each trading post only ever has *one* trading position associated with it (this is brought about by the necessity to introduce complex locking mechanisms if multiple trading posts were required to access a single position).

The primary responsibilities of the trading post are:

- Submission of trading positions.
- Triggering and running the matching algorithms and collation of the results.
- Computing the clearing price and publishing this information to the exchange (to enable participants to see the current price in the case of a dynamic pricing model).
- Persisting the matched results and parameters such as price and quantity. This information is subsequently fed to the order subsystem.
- Notifying buyers of successful trades and issuing purchase orders.

In Figure 138, you will see that all other processes, other than the triggering of the trading post process itself, originate from the trading post.

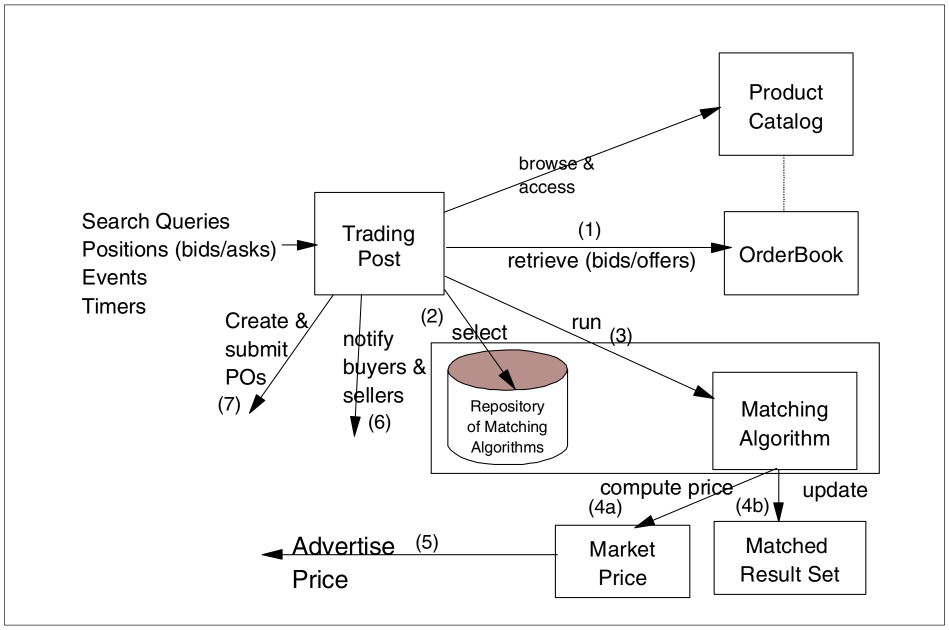


Figure 138. Trading post and associated exchange processes

16.3.2.5 Database tables

This section details the tables related to the exchange subsystem.

Table 48. TRADINGPOSITION

Column	Type	Comments
TradingPositionId	Integer not null	Primary key
TradingPost	Integer	Which trading post is to handle this order. May be null initially when order is being created. Foreign key references column TradingPostid in table TRADINGPOST.
CreatorId	Integer not null	Directory key of creator. Foreign key references column SHRFNBR of table SHOPPER.
CreatorPositionId	Integer not null	Identification number for order supplied by the creator.
ParentPosition	Integer	The parent position from which this position has been derived. May be null if this position was directly submitted by a trader. Foreign key references column TradingPositionId in table TRADINGPOSITION.
RootPosition	Integer	This is a pointer to the original position submitted by the trader from which this position is derived. See note 1. Foreign key references column TradingPositionId in table TRADINGPOSITION.
WhenCreated	Timestamp	
Description	Varchar(254)	Optional, provided by creator.

Column	Type	Comments
ProductId	Integer	Foreign key references column PRRFNBR in table PRODUCT. This is the item that is the subject of this position.
Status	Smallint	Enum: 0 - Inactive 1 - Active Additional states TBD
Side	Smallint	Enum: 0 - Buy 1 - Sell
StartTime	Timestamp	When order processing is to start.
EndTime	Timestamp	When order processing is to end.
Currency	Char(3)	Alphabetic currency codes as per ISO 4217.
Price	NUM(15,2)	
Quantity	NUM(15,5)	
Units	Integer	Enum: TBD

Table 49. TRADINPOSITIONATTRIBUTE

Column	Type	Comments
TradingPositionId	Integer not null	Foreign key references column TradingPositionId of table TRADINGPOSITION.
Name	Varchar(254) not null	
Value	Varchar(254)	
Type	Varchar(254)	TBD.
Constraint	Varchar(254)	TBD. Need a simple constraint language.

Table 50. MATCHRESULTS

Column	Type	Comments
MatchId	Integer not null	Primary key
BuyPosition	Integer	Foreign key references column TradingPositionId of table TRADINGPOSITION.
SellPosition	Integer	Foreign key references column TradingPositionId of table TRADINGPOSITION.
TradingPostId	Integer	Foreign key references column TradingPostId of table TRADINGPOST.
Price	Num(15,2)	
Quantity	Num(15,5)	
Description	Varchar(254)	
WhenMatched	Timestamp	
WhenFilled	Timestamp	

Table 51. MATCHATTRIBUTE

Column	Type	Comments
MatchId	Integer not null	Foreign key references column MatchId of table Match.
Name	Varchar(254) not null	
Value	Varchar(254)	
Type	Varchar(254)	TBD.

Table 52. TRADINGPOST

Column	Type	Comments
TradingPostId	Integer not null	Primary key
Owner	Integer not null	Foreign key references column SHRFNBR in table SHOPPER.
Name	Varchar(254) not null	Name of the TradingPost.
SecurityPolicy	Varchar(254)	TBD
State	Integer	Enum: 0 - inactive 1 - active
TradingPostActions	Long Varchar	Java class
TradingPostSecurity	Long Varchar	Java class
TradingPostStatistics	Long Varchar	Java class
TradingPostPositionManager	Long Varchar	Java class

Table 53. MATCHREPOSITORY

Attribute Name	Type	Description	Length
MatchAlg	Varchar (40)	Name of the algorithm.	40
MatchPol	Varchar (20)	1:1 or 1:M or M:1 or M:N	20
MatchAlgId	Matching Algorithm ID	Integer	11
MatchDLL	Varchar (50)	DLL or JAR file of the matching algorithm	50
MatchObj	Varchar (100)	The objective of the matching algorithm	100

Attribute Name	Type	Description	Length
MatchConstr	Varchar (1000)	The list of constraints for matching	100
MatchSTTime	Time	Start of the Matching Period	
MatchEndTime	Time	End of the Matching Period	
MatchInterval	Time	How often the matching algorithm should be done	
MatchNumber	Integer	Minimum number of buyers/sellers before the matches can take place	11
AccessControl	Integer	Market ID or Group ID	11

16.3.2.6 Commands, data beans and packages

In this section, we provide you with lists of the commands, data beans and packages related to the exchange subsystem.

Commands

AddToMatchReposCmd

ConfigureMatchingCmd

CreatePOFromMatchesCmd

MarketAdminMatchResReportCmd

MarketStatisticsBeanCmd

MatchedResultBeanCmd

MatchedResultReportCmd

MatchingRegistryBeanCmd

MatchPositionsCmd

OrgMatchResReportCmd

QueryMatchedResultCmd

ViewMarketStatisticsCmd

ViewMatchedResultCmd

ViewMatchRepositoryCmd

ViewUserMatchedResultCmd

CancelExchangePositionCmd

CreateExchangeOfferingDetailsCmd

CreateExchangePositionAccessControlCmd

CreateExchangePositionCmd

DisplayAllExchangePositionsCmd

EditExchangeOfferingCmd

ExchangeOfferingCmd

GetAllExchangeOfferingsCmd

GetExchangeOfferingListCmd

GetExchangePositionCmd

GetTradedExchangeOfferingsCmd

UpdateExchangeOfferingDetailsCmd

UpdateExchangePositionCmd

ClearMarketCmd

CreateTradingPostCmd

GetAllTradingPostsCmd

GetTradingPostCmd

GetTradingPostFormCmd

StartTradingPostCmd

StopTradingPostCmd

SubmitExchangePositionCmd

UpdateTradingPostCmd

Data beans

MarketPriceBean

MarketStatBean

MarketStatisticsBean

MatchedResultBean

MatchedResultBeanBeanInfo

MatchedResultInfoBean

MatchingRegistryBean

UserBean

ExchangeOfferingBean

ExchangeOfferingDetailsBean

ExchangePositionAccessControlBean

ExchangePositionAttributeBean

ExchangePositionBean

TradingPostAccessControlBean

TradingPostBean

Java package names

com.ibm.commerce.exchange

com.ibm.commerce.exchange.orderbook

com.ibm.commerce.exchange.matching

com.ibm.commerce.exchange.matchresults

com.ibm.commerce.exchange.pricing

com.ibm.commerce.exchange.markets

com.ibm.commerce.exchange.tradingpost

com.ibm.commerce.exchange.admin

com.ibm.commerce.exchange.util

com.ibm.commerce.exchange.misc

JSPs

DefaultAddNewAttribute.jsp

DefaultCancelExchangePosition.jsp

DefaultClearMarket.jsp

DefaultCreateExchangeOfferingForm.jsp

DefaultCreateExchangePosition.jsp

DefaultCreateExchangePositionForm.jsp

DefaultCreateTradingPostForm.jsp

DefaultDisplayAllExchangeOfferings.jsp

DefaultDisplayExchangeOfferingDetails.jsp

DefaultDisplayMyExchangeOfferingDetails.jsp

DefaultDisplayMyExchangeOfferings.jsp

DefaultDisplayMyExchangePositionDetails.jsp

DefaultDisplayMyExchangePositions.jsp

DefaultDisplayMyExchangePositionsFilter.jsp

DefaultDisplayMyTradingPostDetails.jsp

DefaultDisplayMyTradingPosts.jsp

DefaultDisplayOrderbookExchangePositionDetails.jsp

DefaultDisplayOrderbookExchangePositions.jsp

DefaultDisplayOrderbookExchangePositionsFilter.jsp

DefaultEditExchangeOfferingForm.jsp

DefaultEditExchangePositionForm.jsp

DefaultEditTradingPostForm.jsp

DefaultSubmitExchangePosition.jsp

DefaultUpdateExchangePosition.jsp

MyPositionDetails.jsp

adminReportForm.jsp

displayMatchReposDetails.jsp

editMatchReposDetails.jsp

marketAdminReport.jsp

marketstat.jsp

matchReposStatus.jsp

matchcmdok.jsp

matchedresultall.jsp

matchedresults.jsp

matchrepos.jsp

myReport.jsp

myReportForm.jsp

mymatchedresult.jsp

orgReport.jsp

orgReportForm.jsp

userReport.jsp

userReportForm.jsp

usermatchedresult.jsp

16.3.3 e-Marketplace administrator interaction

The e-Marketplace administrator is responsible for creating exchange offering for products in the e-Marketplace. An offering must be created and assigned to a trading post. During creation of a trading post the type of matching algorithm is selected. This algorithm controls the matching of positions entered into the exchange system by buyers and sellers.

Creating a trading post is the first task an administrator must perform. The following steps describe this process:

1. The e-Marketplace administrator logs on to NCADMIN.
2. The e-Marketplace administrator selects **Exchange**.
3. The e-Marketplace administrator selects **Create Trading Post**.
4. The e-Marketplace administrator completes the Create a Trading Post window. Here, the e-Marketplace administrator decides on the initial state of the trading post (Running or Stopped) and the algorithm which is used for the matching process.
5. From the Create a Trading Post window the e-Marketplace administrator clicks **Create Trading Positions**.
6. A confirmation window is displayed with the detailed information about the newly created trading post.

If the initial status is set to Running, then at this time the trading post is functional and ready to accept creating and assignment of exchange offerings.

If the initial status is set to Stopped, the e-Marketplace administrator can select the **My Trading Posts** link and from the list of trading posts select **Start**.

The e-Marketplace administrator can create exchange offerings from either the NCADMIN or from the e-Marketplace Web site. The following steps describe this process in both cases:

1. From the Exchange link of the NCADMIN, the e-Marketplace administrator selects **Create Offering** or from the e-Marketplace Web site the e-Marketplace administrator selects **Manage Offerings**.
2. The e-Marketplace administrator navigates through the catalog and finds the product description for which an exchange offering needs to be created.
3. The e-Marketplace administrator clicks **Create Exchange Offering**.
4. The Create Exchange Offering window is displayed with the attributes of the selected product description.

5. The e-Marketplace administrator can add new attributes to this offering from the Attribute List section.
6. The e-Marketplace administrator selects a trading post, provides a description for the offering and an initial price, and enables or disables the automatic order processing from the Offering Details section.
7. The e-Marketplace administrator selects **Public** or **Restricted** access for the offering and clicks **Create Offering**.
8. An Exchange Offering Details window is displayed showing the status and the details of the offering.

At this point the offering is available to buyers and sellers to enter their positions (bids and asks).

16.3.4 Supplier interaction

The typical supplier interactions with the exchange subsystem can be summarized as follows:

1. The supplier registers in the e-Marketplace and provides the required information, such as organization, authentication, contact, role, and preferences.
2. The seller browses the product categories.
3. The seller places a product for sale.
4. The seller selects the product to be sold via an exchange.
5. The seller is directed to the exchange area of the site.
6. The seller can do one of the following:
   - Browse the orderbook for the current market conditions for the product, and place an offer on the new product to be sold by the seller.
   - Browse the orderbook and place an offer on an existing product.
   - Choose not to place an offer and exit the exchange.
7. Once the offer is placed, the seller can perform one of the following:
   - Browse the product table.
   - Browse the orderbook (for placing further offers).
   - View the status of the offer (if matched or not).
   - Remove the offer (if needed).

Suppliers also can interact directly by using the Exchange link. Four options are available:

1. My Positions: Lists the supplier's positions, including information on the position and its status. The supplier can view the details of each position.
2. Exchange Offerings: Lists all the exchange offerings in the e-Marketplace that a supplier is allowed to participate in. The supplier can view details, view active positions or create a position.
3. Market Statistics: Displays statistics information pertaining to a supplier.
4. Completed Trades: Displays a list of completed exchanges belonging to a supplier.

16.3.5 Buyer interaction

The typical interactions that occur between a buyer and the exchange subsystem can be summarized as follows:

1. The buyer registers in the e-Marketplace and provides all the necessary organization, authentication, contact and preference information.
2. The buyer browses the product catalog.
3. The buyer selects a product and nominates to purchase it via the exchange mechanism.
4. The buyer is then directed to the exchange area of the e-Marketplace.
5. The buyer can either:
   - Browse the orderbook for that product to obtain current prices and availability and then place a bid.
   - Place a bid (given the current average price of the product).
   - Leave the exchange without participating in the exchange.
6. Once the bid is placed, the buyer can do one of the following:
   - Browse the product catalog further.
   - Observe the status of the bid.
   - Browse the orderbook to place more bids.
   - Remove submitted bids.

Buyers also can interact directly by using the Exchange link. Four options are available:

1. My Positions: Lists the buyer's positions, including information on the position and its status. The buyer can view the details of each position.

2. Exchange Offerings: Lists all the exchange offerings in the e-Marketplace that buyer is allowed to participate in. The buyer can view details, view active positions or create a position.
3. Market Statistics: Displays statistics information pertaining to buyer.
4. Completed Trades: Displays a list of completed exchanges belonging to buyer.

16.3.6 Interaction with other components and subsystems

The primary interactions with the other components of the exchange subsystem are initiated from the trading post and are depicted in Figure 138 on page 426.

These interactions are summarized by the following process, which the trading post performs as a result of receiving messages or timed events.

The trading post:

1. Selects a matching algorithm based on a matching policy (that is one to many, many to one, many to many), matching objective (maximize trades) and pricing policy (single or multiple clearing prices).
2. Retrieves the positions (bids and offers).
3. Matches bids and offers.
4. Computes match prices.
5. Stores the matched results as a list of buyer-sellers pairs.
6. Notifies buyers and sellers.
7. Updates the market statistics table and publishes prices.
8. Creates purchase orders for each buyer-seller pair.

Chapter 17. Example - additional e-Marketplace infrastructure

In this chapter we discuss additional infrastructure provided by the Marketplace Edition to facilitate the operation of an e-Marketplace. We give details of the hub business subsystem of the Marketplace Edition. The hub business subsystem handles all of the reporting for the Marketplace Edition. These reports can be customized by the Marketplace Edition administrator. At the time of writing this redbook the customization of these reports had not been added in to Marketplace Edition. We discuss each one of the reports and what information they provide. We will also discuss the orders process, the approval process and the flex flow system.

17.1 Hub business subsystems

Once a transaction such as an auction, RFQ, exchange or catalog buying is completed, a purchase order (PO) will be created by the Marketplace Edition. When a PO is generated (the order approval is assumed to be finished prior to this.), a PO notification message will be sent to the supplier by e-mail. The supplier can log on to the hub to view, accept or reject the PO. The corresponding acknowledgments will be sent to the buyer when the supplier views, accepts or rejects the PO. The system provides the ability to store POs in XML format ,which allows buyers and suppliers to download the XML file of PO. The purpose of this feature is to loosely integrate the back-end systems of suppliers and buyers with the hub. The downloaded XML file could be pumped into the organization's back-end system for processing. For example, a hub level report for the hub administrator and report for the organizations buyer and supplier will be provided. This also includes a common entry UI for reporting by the transaction type, such as auction, RFQ, exchange and catalog buying. All the reports can be saved in XML file for further processing.

Furthermore, the system provides the supplier with the ability to download RFQ request in XML format.

The hub business subsystem consists of:

- Hub revenue reporting.
- XML reports.
- Table for fee calculation.
- Log business events.
- Common UI provided for requesting reports.
- A set of available reports that can be generated on demand.

© Copyright IBM Corp. 2000

- WCS reporting function.

17.2 e-Marketplace reports

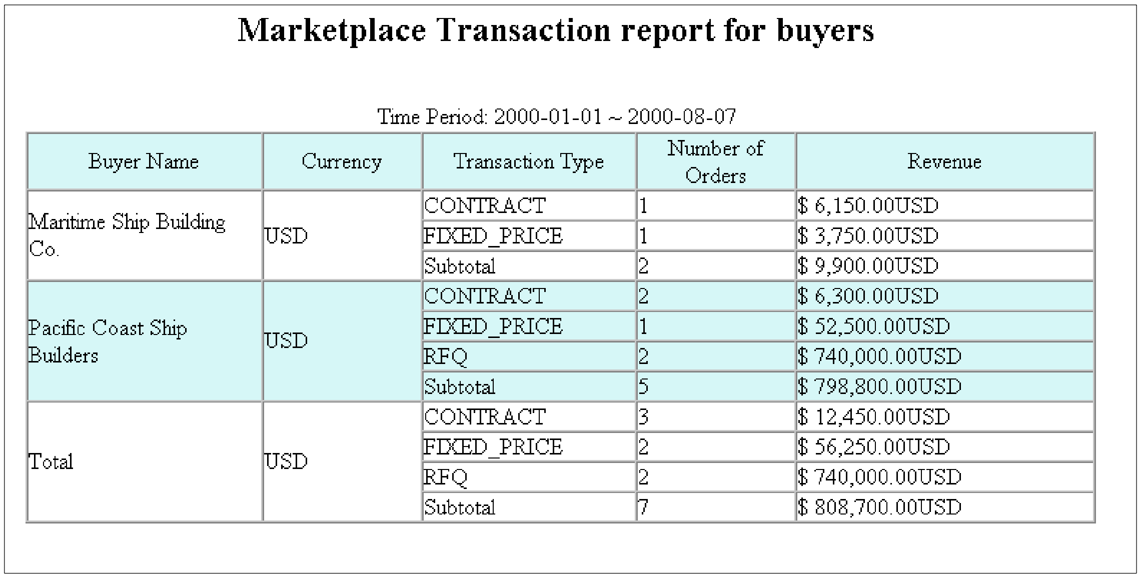
The e-Marketplace reports show supplier transactions, buyer transactions, membership details, catalog and a summary of the complete exchange trades. They are viewable by the hub administrator only and provide a record of all trading activity in an e-Marketplace hub

- Marketplace supplier transaction report

  The marketplace supplier transaction report shows all the supplier transactions. Some of the information in this report is supplier name, the currency used to purchase the product, the transaction type, the number of orders that were placed and the cost of the product.

- Marketplace buyer transaction report

  The marketplace buyer transaction report shows all the buyer transactions. Included in this report is the buyer's name, the currency used for the purchase, the transaction type, the number of orders placed, and the revenue from the transaction. Figure 139 is an example of a buyer transaction report.



Marketplace Transaction report for buyers

Time Period: 2000-01-01 ~ 2000-08-07

Buyer Name	Currency	Transaction Type	Number of Orders	Revenue
Maritime Ship Building Co.	USD	CONTRACT	1	$ 6,150.00USD
		FIXED_PRICE	1	$ 3,750.00USD
		Subtotal	2	$ 9,900.00USD
Pacific Coast Ship Builders	USD	CONTRACT	2	$ 6,300.00USD
		FIXED_PRICE	1	$ 52,500.00USD
		RFQ	2	$ 740,000.00USD
		Subtotal	5	$ 798,800.00USD
Total	USD	CONTRACT	3	$ 12,450.00USD
		FIXED_PRICE	2	$ 56,250.00USD
		RFQ	2	$ 740,000.00USD
		Subtotal	7	$ 808,700.00USD

Figure 139. MarketPlace Buyer transaction report example

- Membership details

The membership detail report will show the details of all the members of the e-Marketplace. At the time of this writing the membership detail report was not available.

- Catalog reports

  The catalog reports show all the details of all the items in the catalog.At the time of this writing the Catalog Report was not available.

- Summary of complete exchange trade report

  The exchange trade summary will provide a complete summary of all of the exchanges that have taken place in the e-Marketplace. At the time of this writing the exchange trade report was not available.

17.3 Organization reports

The organization reports are for certain organization in the e-Marketplace. It focuses on transactions between the organization and specific or all trading partners. These reports are viewable by hub and organization administrators only and provide a record of an organization's trading activities and employee e-Marketplace responsibilities.

There are two type of trading partner (TP) for an organization: seller-side TP and buyer-side TP. Reports for these two types of TP are both available.

- Financial reports

  Orders submitted to suppliers and orders received from buyers.

- RFQ transactions
- Auction transactions
- Exchange transactions
- Contract transactions
- Catalog reports

  Summary statistics of offering types stored in the catalog

- Membership reports

  Summary of members, including roles

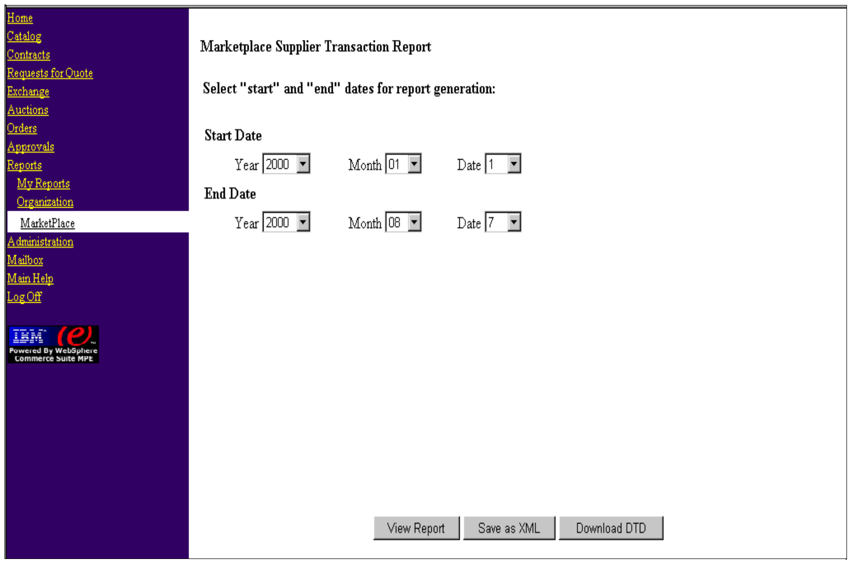
17.4 Member reports

The member reports will give the details of all the members interactions with the e-Marketplace such as the RFQs, the auctions and the exchanges that each one of the members have performed.These reports provide a record of the activities of organization employees (users) responsible for carrying out activities in the e-Marketplace.

- RFQ transaction reports: The RFQ transaction reports will show all of the RFQ transactions that a member has performed in the e-Marketplace.
- Auction transaction reports: The auction transaction report will show all of the auction transactions that have been executed and a detailed summary of each.
- Exchange transaction reports: The exchange transaction report will show all the details of all of the transaction that have take place within the e-Marketplace.

17.5 XML download ability

For offline viewing of all reports the e-Marketplace provides the ability to download the DTDs for the reports and the XML data of the report. This provides the ability for integrating the data with any other business data. At the bottom of the pages of the report, that provides this feature you will see buttons allowing for XML or DTD download as shown in Figure 140.



Marketplace Supplier Transaction Report

Select "start" and "end" dates for report generation:

Start Date

Year 2000 Month 01 Date 1

End Date

Year 2000 Month 08 Date 7

Figure 140. Offline report download page

17.6 Offline reports

The e-Marketplace provides the ability to view report offline, this is accomplished by providing a downloadable in XML format.

17.6.1 Configure offline reports

In order to be able to download the offline reports the user will have to select the offline reporting period. When the user clicks the **Save as XML** button the user will be prompted for a location to download the XML file to. The DTD scheme can also be downloaded for each offline report. The user clicks the **Download DTD** button, and is prompted for a location to download the DTD for the report.

17.7 Order subsystem

The order subsystem in the Marketplace Edition is based on the WebSphere Commerce Suite 4.1 as much as possible. Some of the areas that were changed are as follows:

- Orders can be created by more than one system (the catalog). Orders can also be issued from other sub systems including RFQ, contracts, and exchange.
- Most of the new order functionality is written in Java.
- Orders can be submitted atomically, that is in one shot, and can be collated into a finer grain of categorization.
- Orders in the Marketplace Edition do not have order aggregation.

17.7.1 High-level overview

In the WebSphere Commerce Suite the order subsystem has the ability to review and update information related to inventory and pricing. The WebSphere Commerce Suite, Marketplace Edition for AIX is largely written in Java and WebSphere Commerce Suite is written in C++, so it was difficult to integrate the C++ subsystem logic with the Java access logic. The decision was made to allow a subsystem to choose between integrating into the order system itself or use the order subsystem at arms length through an atomic submission which cannot fail. In the first option, a subsystem might choose to integrate with the orders subsystem in an easy way and benefit from a finer grain of integration where more flexibility is allowed and less up-front work is needed. With the second option, a subsystem might decide that it is simply too hard to replicate logic in the C++ environment. The subsystem then has to do some up-front work: verify all information, create an order and submit it on the spot. Therefore, a user never deals with an order in the pending state. An order goes directly to a completed state after it is created. One very important note to remember is that an order is keyed on the seller. This means that if a buyer selects multiple items from various sources, and those items come from various sellers as well, one order per seller will be created. More importantly, the order subsystem does not do aggregation. Therefore, once placed, there is no remembering that order 1 and 2 came out of the same submission.

17.7.2 Low-level overview

To allow for more flexibility for multiple subsystems, the Marketplace Edition created a new technique to string together overridable functions (OF) in a task. For instance if we want to check inventory for a particular item, a contracted item has its inventory in the contract subsystem, and a regular (fixed price) items has its inventory in the catalog subsystem. A quick solution is to write an OF that determines the type of items and either executes logic for a contract or composes the standard OF for catalog logic. OF composition is a valid technique for the WebSphere Commerce Suite, but it is used primarily for extensions where the logic of both OFs gets executed. Here, the situation is different: the composer acts as a router, and then either executes its business logic or executes another OF's business logic. This is not an extensible proposition; if new types of items come to life, the base OF will need to be changed, or a new OF that does the Route/Compose work will have to be written, thus hard-wiring compositions of OFs. Second, composing an OF, although not as expensive as composing a command, still represents execution time and for an OF that gets called often, this can be an issue.

To solve this problem Marketplace Edition introduces a new technique to allow for multiple mutually exclusive OFs to coexist under one task. The Guarded Chain solution strings together a series of OFs through OF chaining. When an OF is called, it checks for a guard in the environment. If the guard is there, it simply returns true; otherwise, it figures out if the OF applies. If it applies, it executes and sets a guard in the environment; otherwise, it simply returns true. At the end of the chain, a special OF checks for the guard. If no guard is present, the OF returns false indicating that no OF was suitable for the request, thus failing the task. If a guard is found, the OF returns true and the task is successful. This process is illustrated in Figure 141.

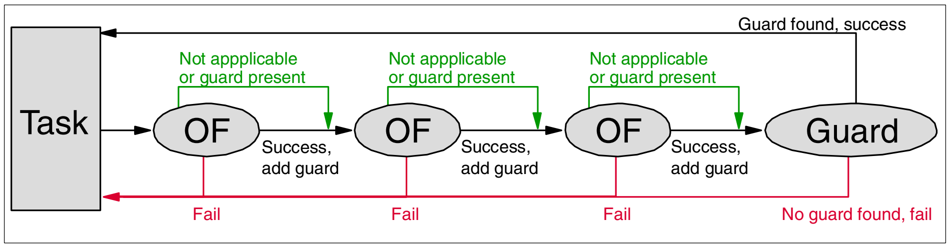


Figure 141. Overridable function interaction diagram

17.8 Approvals subsystem

In this section we will discuss the approval process, which is used to protect commands that participate in the Marketplace Edition. The approvals feature ensures that the action embodied by the command is approved by the right authority. The approval process implemented by the Marketplace Edition provides for:

- One level of approval
- A single approver
- Notification by e-mail
- An API to external workflow engines

In order to provide this functionality the Marketplace Edition has approval administration and configuration tools for:

- Initial mapping of approvable commands
- Registration of commands
- Aassignment of the approver

The approvals process uses this configuration information and provides tools to allow:

- Submitters to query the status of commands they submitted for approval.
- Approvers to view, accept, and reject commands.

17.8.1 High-level overview

Before a command can be used with the approval system, it must be registered as an approvable command. In this section we will give a general overview of the approval process.

All commands requiring approval are automatically replayed in the Marketplace Edition with a standard flex flow state machine as shown in Figure 143 on page 449. In Figure 143 on page 449 the submitRFQ command request is queued, and a pseudo-command “preSubmitRFQ” is executed. This results in the approvals flex flow state machine reaching an intermediate state where it sends out a message <buyer, submitRFQ, RFQ> to the approvals workflow daemon (corresponding to the message template <user, command, businessobject>). The message router routes the message to the approvals workflow daemon process AppWF. The approvals workflow daemon interface process is Boolean. If the command for that business object is approved, the AppWF process issues a message <approve> to the flex flow approval state machine. This results in a transition to state 2. Here the queued original command <buyer, submitRFQ> is valid and is executed. If the request is not approved the <reject> message from the approvals workflow results in the flex flow state machine reverting to the original state, and the command is not executed. The approvals workflow process checks to see if any approval process is registered for approval. If not, a default approval process is kicked off and the following actions will be executed:

- Get approver: Look up approver for <user command> from the approvals registration data. If this is not available, it looks up the parent of <user> in the membership hierarchy.
- Send e-mail (or notification) to approver with a link page showing <user, command,business object> with two buttons, approve and reject.
- Clicking the button results in this default approval process sending the appropriate message to the approval flex flow.

If a workflow process is registered for <user, command> , then it is executed. The only allowed workflow process is a proxy (or interaction controller) that interfaces to a standard workflow engine and uses jFlow to communicate with the workflow engine.

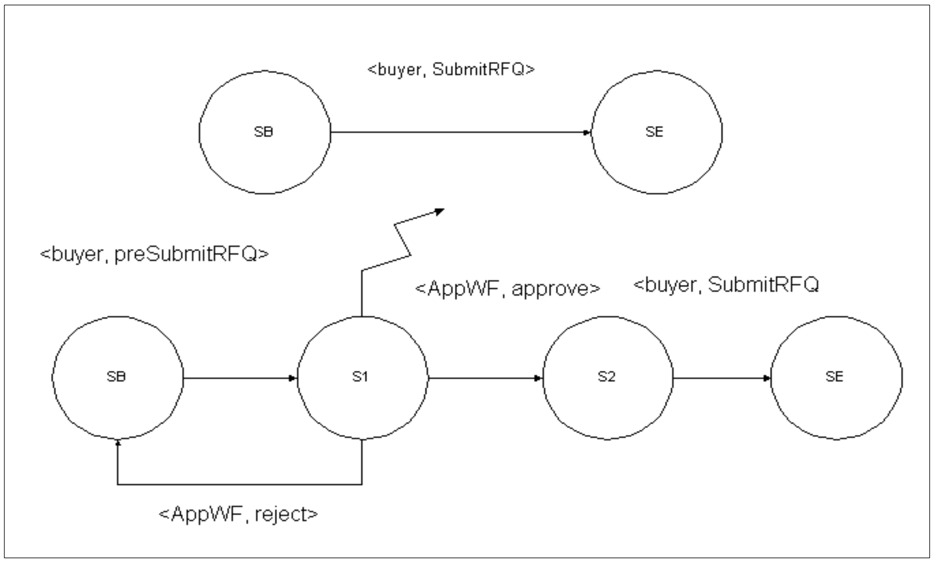
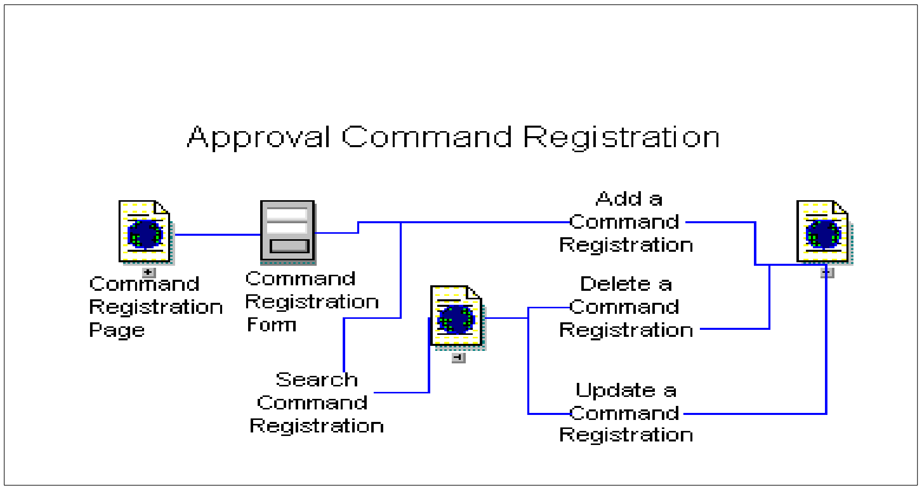


Figure 142. Approval interaction diagram

Figure 143 shows the Approval Command Registration process.

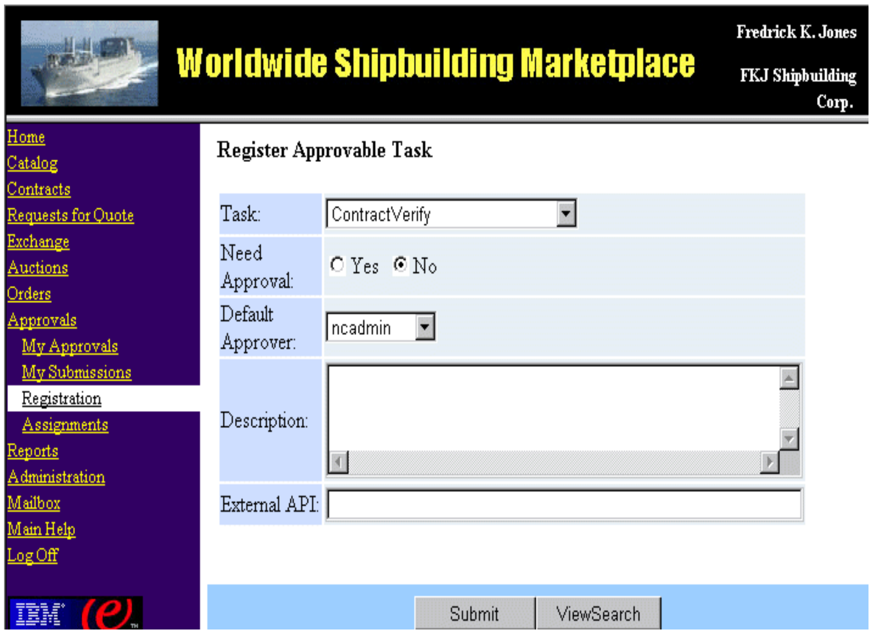


Approval Command Registration
Command Registration Page
Command Registration Form
Add a Command Registration
Delete a Command Registration
Search Command Registration
Update a Command Registration

Figure 143. Approval command registration

Figure 143 shows the flow of the approval process. After a command has been submitted for protection, there are several things that can be done to this command: delete the command if it is no longer needed, update an already active command, or search for commands that have already been registered.

Figure 144 shows and example of the window used in the Marketplace Edition to register a command for approval.



Register Approvable Task

Task:	ContractVerify
Need Approval:	Yes [✓ No]
Default Approver:	ncadmin
Description:	
External API:	

Figure 144. Command registration

17.8.2 Low-level design

The approval subsystem consists of the following interaction controllers:

- ApprovedCommandRegistration

  This IC applies anUI to manage all commands for each subsystem. The administrator can register a command that needs to be approved or register a command that does not need to be approved. The IC will invoke the GetApprovalCommandCmd command and the CommandList.jsp.

- ApproverSpecification

  This IC applies an UI to manage the relationship between registered commands and approvers for each organization. The organization administrators could specify an approver for a specified user or specify “null” approvers for a user, to indicate this user has no approval needs for this command. Even the command is registered as needing approval. The IC will invoke the GetApproved SpecificationCmd command and ApproverSpecifyList.jsp.

- ApprovalProcessApprover

  This IC is used to search and display all submissions which need to be approved by users. It will invoke the GetSubmissionListCmd command and AllSubmissionList.jsp.

- ApprovalProcessDetail

  This IC is invoked when the user clicks the **detail** button or clicks the submission in the submission list. The IC will call PreDisplaySubmissionDetailCmd to get the string used to redirect to the DisplayIC servlet. Each subsystem that will display detailed information about the submission.

- ApprovalProcessApproveOrReject

  This IC is invoked when the approver approves or rejects the submission. The IC will invoke the ApproveOrRejectCmd to note the change of status.

- ApprovalProcessSubmitter

  This IC is used to search and display all submissions of the user based on that user's search. It will invoke the GetSubmissionsListCmd and AllSubmissionList.jsp.

- ApprovalProcessDeleteSubmission

  This IC is used to delete submission for a submitter.

17.9 Flex flow high-level overview

Flex flow is a facility of the Marketplace Edition infrastructure that is used to define and control business process flows within the e-Marketplace. The flows specified can be configured at three levels:

- Business processes
  - Flow between commerce functions
  - Across platforms and trusted boundaries
  - Both data and control flow
- Commerce flows
  - Sets the flow of commands and provides a logical grouping of all commerce activities
  - Shares data and schemas with the business objects

- Commands
    - Used for controlling the flow of tasks and atomic transactions

Figure 145 shows an overview of flex flow and how it interacts with the business process.The large middle circle represents tasks that are being executed.

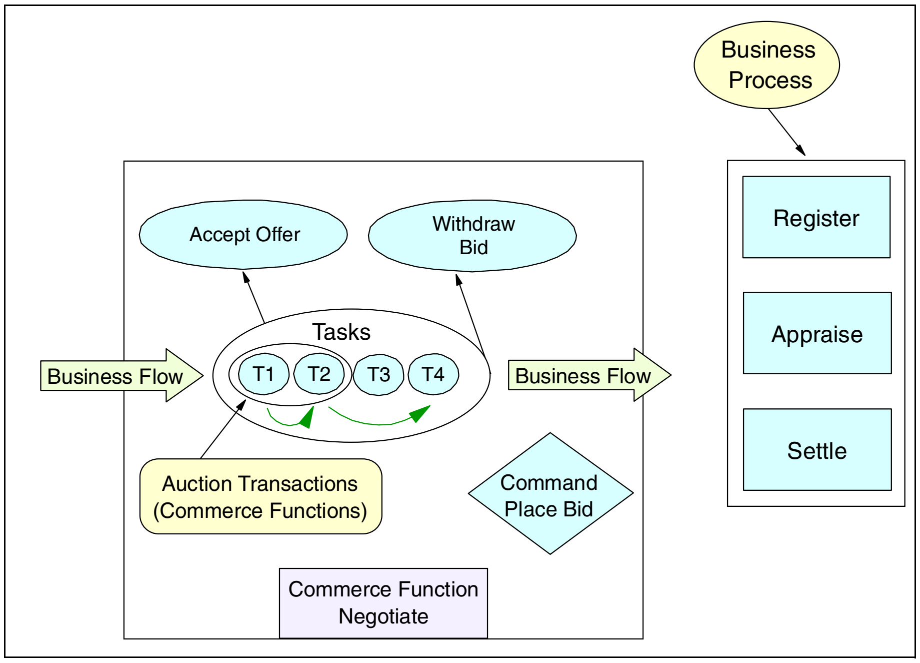


Figure 145. Overview of flex flow and business process

17.9.1 Level one: commands

The command in the flex flow controls the flow through a set of tasks. The commands provide instantaneous completion, uninterruptable execution of commands. Commands will also provide each participant with a single action to be executed. As an example, a consumer submits an offer in a negotiation. There will be tasks that will verify the purchase and credit limit, payment documents, discounts by buyer, and inventory updates. Note that all tasks will not be executed, but only the tasks that are dependent on the conditions that a consumer has submitted, for example purchase amount, buyer loyalty, etc.

17.9.2 Level two: commerce functions

The commerce functions in the flex flow will provide logical grouping of commands that are used for a single commerce activity or function. Commerce flex flow functions will validate the sequence of commands. In the the order process the commerce functions will validate partial or full order commands. The commerce functions are controlled by the finite state machine control. Some of the other features of the commerce functions are the sharing of data schemas and business objects.

17.9.3 Level three: market/business processes

The market/business processes will provide the activation of all commerce functions and will control the data or all control-driven activations. In the market/business process, data and schemas or business objects are not shared. An example of the market/business process is a seller selling an antique. In this case the market/business process will control the registration of the seller, the identify and notification of a potential buyer, the negotiation and the settled sale.

17.10 Flex flow low-level overview

Flex flow uses a state machine that has two parts to control the state of each of the commands that have been executed. In this section we discuss the low-level details of the flex flow state machine. Flex flow has a state machine for processing a command and another state machine for translation of the commands. We discuss both of these state machines in detail and give examples of how they are used in the e-Marketplace.

17.10.1 State machine

There are two major parts to the state machine. One state machine will handle processing and the second state machine will handle the translation of commands. The state machine processing includes code for controlling the transitions within the state machine and code for accessing the DB2 tables used for storing the states and transitions. The state controller keeps track of a session which is persistent (that is, stored in DB2). A state controller can be instructed to resume processing of a previous session. A two phase commit mechanism for transition processing is being used. The process invoking the state controller will be allowed to perform a transition in two steps: process action followed by either a confirm or a rollback. The process action method will check the validity of the requested transition, and if valid the process will be marked as pending by the state machine while tracking what the next state will be upon confirmation of completion. A confirm or rollback will move the machine out of the pending state with confirm completing the transition of the state to the new state. A rollback will return the machine to the state it was before the processAction. There is a method that will dynamically retrieve the next available actions for a particular role in a particular session. This is useful for inserting Web pages in the getAllowedTransition method. All state controller actions are logged in a file specified in the state machine properties file (sm.properties). The state machines translate three different formats. One is for XML, the second is for the GUI creation tool, and the third one is for the DB2 tables.

Appendix A. Marketplace Edition installation guide

This appendix provides some details of the procedure we used to install WebSphere Commerce Suite, Marketplace Edition and the standard sample application. This is not a detailed instruction guide, and we recommend that you carefully read the official Marketplace Edition installation documents. Please also that we expect the installation procedure for the Marketplace Edition to change significantly when the product is released.

A.1 Prerequisites

1. Ensure the WebSphere Commerce Suite is installed.

   Marketplace Edition is based on WebSphere Commerce Suite V4.1, so WCS should be installed and configured before installing MPE.At a bare minimum we suggest you create and test an instance called MSER, but to test WCS's use of servlets and of LDAP it is a good idea to install the WCS demomall sample and configure the instance to use LDAP.
2. Upgrade the Java development kit - at the time of writing IBM JDK 1.1.8 PTF 7 is required by the Marketplace Edition.
3. WebSphere Application Server, Advanced Edition should be upgraded - currently V3.0.2.1 is required.
4. The demomall sample should be installed on WCS41.
5. DB2 extenders should be installed and configured.
6. IBM SecureWay Directory (LDAP) should be installed with its own database LDAPDB2. Be sure that you can access the LDAP administrator's screens using your browser.

Important Note

The ldapxcfg utility will modify the contents of the httpd.conf file used by the IBM HTTP server if you ask it to configure a Web server for directory administration.It adds configuration statements to the Web server file that are needed before you can administer LDAP from a browser.The WebSphere Commerce Suite configuration tool updates the same Web server file when you create, modify or delete an instance. If ldapxcfg is used after an instance has been configured by the WCS configuration tool, LDAP config statements are written in the same block of statements as those from WCS. This has the unfortunate consequence that when the WCS instance is deleted the configuration settings for LDAP are also deleted. We recommend that you either run ldapxcfg before creating a WCS instance or that you edit httpd.conf and move the LDAP configuration settings out of the WCS configuration block.

A.2 Test WebSphere Commerce Suite demomall

To test the demomall on WebSphere Commerce Suite, do the following:

1. Register as a user. Search for some products and add them to the shopping cart. Process the order and ensure you can get the order number.
2. Click **home**. Move down the window and click **IBM Computer Next Generation**. Then select **click here to enter Thinkpads category**. This is equivalent to going to the following URL:

   ```
   http://hostname/webapp/commerce/servlet/CategoryDisplay?merchant_rn=600
   1&cgrfnbr=33
   ```
3. Try the following URL (Replace the `http` with `https`):

   ```
   https://hostname/webapp/commerce/servlet/CategoryDisplay?merchant_rn=60
   01&cgrfnbr=33
   ```

If you can complete the above three steps, WCS 4.1 is installed correctly, and the system is ready for installing the Marketplace Edition (MPE).

A.3 Stop Processes

In the following steps use the root ID:

1. Stop WebSphere Application Server, Advanced Edition.

WAS should always be stopped via the Administrative Console, since the starting process created numerous Java processes that should also be stopped.

If you do stop WAS via the command line, also use the following process to stop all the Java processes:

```
Ps -ef | grep java
Kill -9 pid
```

Where `pid` is the process ID.

Repeat the above command for all the WebSphere related Java processes running.

2. Stop your WCS instance - usually this will be MSER

Note: Do not stop the LDAP and the HTTP Server until you have done the backups.

A.4 Backups

It is always a good idea to back up your work. We suggest that you create and mount a separate backup disk system before you start and do all the backups onto this disk.

Note: The only way the remove MPE currently is to start with a fresh copy of WCS.

1. Back up the WebSphere Commerce Suite database

   This is recommended if you will be doing multiple installs of the Marketplace Edition. The current install scripts for MPE modify the WCS database schema and then add test data to the tables. These scripts work on any database that has a schema in the WCS format, but we found it useful to back up the demomall database and use this as the starting point for Marketplace Edition install The steps to back up demomall are:

   a. Enter `su - db2inst1`
   b. Enter `db2 force applications all`
   c. Enter `db2 backup database WAS to /backup`
   d. Enter `db2 backup database demomall to /backup`

2. Back up the IBM Secureway Directory server (LDAP)

   This step is not essential and the Marketplace Edition install scripts will clean up and refresh test data for the LDAP server, but you may consider this necessary if you have any of your own data in LDAP. The steps to follow are:

- Launch a browser and go to the URL `http://yourhostname/ldap` .
- Log in and select the database folder.
- Select the backup link and enter `/backup/export.ldif`

3. Other software backups

Before continuing with the installation of Marketplace Edition you should consider backing up WebSphere Commerce Suite 4.1 and related software. This will be useful should you decide to remove the Marketplace Edition and revert back to the base WebSphere Commerce Suite.

Make sure that you have ample disk space to continue, because large areas of temporary disk space will be needed. The following instruction assume that backups will be placed in the directory /backup:

a. To back up WebSphere Commerce Suite software:
   1. Enter `cd /usr/lpp`
   2. Enter `tar -cvf  /backup/CommerceSuite.tar CommerceSuite; compress/backup/CommerceSuite.tar`

b. To back up IBM HTTP server:
   1. Enter `cd /usr`
   2. Enter `tar -cvf /backup/HTTPServer.tar HTTPServer; compress /backup/HTTPServer.tar`

c. To back up WebSphere Application Server, Advanced Edition:
   1. Enter `cd /usr`
   2. Enter `tar -cvf /backup/WebSphere.tar WebSphere; compress /backup/WebSphere.tar`

d. To back up LDAP
   1. Enter `cd /usr`
   2. Enter `tar -cvf /backup/ldap.tar ldap; compress /backup/ldap.tar`

A.5 Stop the remaining processes

Once you have completed the backups, you can stop LDAP and the IBM HTTP server. Instructions are as follows:

1. Stop LDAP

Use the browser-based administration tool to stop the LDAP server:

   a. Navigate to `http://<hostname>/ldap`
   b. Log on as cn=root using the admin password defined when LDAP was installed
   c. Navigate to **Server -> Startup/Shutdown**
   d. Submit the server shutdown form by clicking the **Shutdown** button

2. Stop the IBM HTTP server
   a. Change to the IBM HTTP server directory - the default is:

      ```
      cd /usr/HTTPServer/bin/
      ```

   b. Enter `./apachectl stop` to stop the Web server

A.6 Install the Marketplace Edition software

In the pre-release code we used when writing this redbook the Marketplace Edition software was distributed as a jar file with a name such as MPE*mmddx*.jar.

To install the Marketplace Edition software:

1. Copy the .jar file to /usr/lpp/CommerceSuite.
2. Run the scourBeforeJar Script
   a. Enter `cd /usr/lpp/CommerceSuite`
   b. Enter `jar -xvf <your_MPEjar_filename>.jar scourBeforeJar.sh`
   c. Enter `./scourBeforeJar.sh`
3. Extract software from the jar file
   a. Enter `cd /usr/lpp/CommerceSuite`
   b. Enter `jar -xvf MPEmmddx.jar`

      If the extraction from the jar file crashes on `.a` files, just `rm` them one by one (they're in memory) and run the extract again.
4. Extract software from the dynasty.jar:
   a. Enter `cd /usr/lpp/CommerceSuite/classes`
   b. Enter `jar -xvf dynasty.jar`
5. Extract software from the collections.jar
   a. Enter `cd /usr/lpp/CommerceSuite/classes`
   b. Enter `jar -xvf collections.jar`

6. Run the scourAfterJar script
   a. Enter `cd /usr/lpp/CommerceSuite`
   b. Enter `./scourAfterJar.sh`

A.7 Modify the LDAP schema

The Marketplace Edition uses a modified LDAP schema which needs to be applied to the LDAP server. This is a once-only task that should be done in preparation for the first time the Marketplace Edition is installed. A detailed set of instructions are available in the file /usr/lpp/CommerceSuite/emp_schema/singlebyte/membership/readme.txt.

The basic steps required are:

1. Stop the LDAP server
2. Copy the new schema file to the LDAP server

   Enter `cp /usr/lpp/CommerceSuite/LDAP/etc/V3.modifiedschema /usr/ldap/etc`
3. Enter `chown ldap:ldap  /usr/ldap/etc/V3.modifiedschema`
4. Start the LDAP server

A.8 Prepare LDAP server

1. Start the IBM HTTP server and the LDAP server.
2. Log on as the root user ID.
3. Optionally, remove existing LDAP test data.

   Enter

   ```
   /usr/lpp/CommerceSuite/emp_schema/singlebyte/membership/cleanLDAP.sh
   cn=root ldap_pw
   ```
4. Start the Web-based LDAP administration tool. In a Web browser enter the URL:

   `http://<hostname>/ldap`
5. Log on as the LDAP administrator.
6. Click **Suffixes>List Suffixes** and check if the suffix dc=emph,dc=ibm,dc=com already exists.
7. To add the suffix if required, click **Suffixes>Add Suffix,** enter dc=emph,dc=ibm,dc=com and click **Add a new suffix.**

A.9 Check and update WCS configuration files

Several WebSphere Commerce Suite configuration files must be updated in order for the Marketplace Edition to run successfully. The files that should be checked and updated are in the directory /usr/lpp/CommerceSuite/instance/mser/config/.

1. ncommerce.conf

   The entries required are:

```
USE_LDAP_REGISTRY 0
LDAP_VERSION 3
LDAP_HOST <hostname>
LDAP_PORT 389
LDAP_USE_SSL 0
LDAP_ADMIN_DN  cn=root
LDAP_ADMIN_PW 6L01GL/4I0Y=
LDAP_SEARCH_ROOT mid=1,dc=emph,dc=ibm,dc=com
LDAP_DEFAULT_BASE mid=1,dc=emph,dc=ibm,dc=com
LDAP_OCS top person organizationalPerson inetOrgPerson
LDAP_TIMEOUT 0
LDAP_CHARSET

EMP_CONFIG_RESOURCE com.ibm.commerce.emp.Configuration
HUB_ORG_ID 1
CURRENCY_MERCHANTS 1, 1

TRACE_MASK SERVER
TRACE_MASK DB
TRACE_MASK CATALOG
TRACE_MASK EMPAPPROVAL
TRACE_MASK EMPBIZ
TRACE_MASK EMPCOMMON
TRACE_MASK EMPCONTRACTS
TRACE_MASK EMPEXCHANGE
TRACE_MASK EMPFLEXFLOW
TRACE_MASK EMPMEMBERSHIP
TRACE_MASK EMPRFQ
TRACE_MASK EMPAUCTIONS

ACCESS_DEFAULT_POLICIES
/usr/lpp/CommerceSuite/emp_schema/singlebyte/accesscontrol/defaultOrgan
izationPolicies.xml
```

2. scheduler.conf

   The entries required are:

```
HUB_ORG_ID 1
CURRENCY_MERCHANTS 1, 1
EMP_CONFIG_RESOURCE com.ibm.commerce.emp.Configuration
```

3. srvrctrl.conf

   The entries required are:

   ```
   HUB_ORG_ID 1
   ```

A.10 Prepare the Marketplace Edition database schema

To modify the schema of your WCS database to have the changes required by the Marketplace Edition:

1. Log on as root user ID.
2. Set up required the Marketplace Edition environment variables; enter

   ```
   . ./usr/lpp/CommerceSuite/emp_schema/singlebyte/mpesetenv.sh
   ```

3. Change to the database administrator ID; enter `su - db2inst1`
4. Enter `cd /usr/lpp/CommerceSuite/emp_schema/singlebyte`
5. Enter `createSchema.sh demomall` where demomall is the name of the WCS database that you want to use with the Marketplace Edition
6. Enter `createTestData.sh demomall` to create test data in the modified database.

A.11 Create LDAP test data

To install test data in the LDAP server:

1. Log on as root user ID
2. Enter `cd /usr/lpp/CommerceSuite/emp_schema/singlebyte/membership`
3. Enter `./ldapinit.sh cn=root ldap-pw` where `cn=root` is the LDAP administrator ID and `ldap-pw` is the administration password

A.12 Create test data for the Marketplace Edition database

This step is optional. It will create IBM test data in the Marketplace Edition database tables. The examples in this redbook use the IBM-supplied test data. To create the test data:

1. Change to the database administrator ID; enter `su - db2inst1`
2. Enter `cd /usr/lpp/CommerceSuite/emp_schema/singlebyte`

3. Enter `createTestData.sh demomall`

A.13 Install the Eureka search tool

To install the Marketplace Edition Eureka search tool:

1. Enter `cd /usr/lpp/CommerceSuite`
2. Enter `cp -R eureka client`
3. Enter `cp -R pangea client`
4. Enter `cd /usr/lpp/CommerceSuite`
5. Enter `cp -R eureka classes`
6. Enter `cp -R pangea classes`
7. Add the following alias in /usr/HTTPServer/conf/httpd.conf (if it does not already exist):

   ```
   Alias /client/ /usr/lpp/CommerceSuite/client/
   ```

A.14 Copy jndi.jar

Place a copy of the jndi.jar from the LDAP installation in the Marketplace Edition directory structure:

Enter `cp /usr/ldap/java/jndi.jar /usr/lpp/CommerceSuite/classes`

A.15 Define Servlets in WebSphere

To define the Marketplace Edition servlets to WebSphere Application Server, Advanced Edition:

1. Ensure that WAS is started:
   a. Enter `cd /usr/WebSphere/Appserver/bin`
   b. Enter `./startupServer.sh`
   c. WAS is started when the message WebSphere Administration server open for e-business is written to the file /usr/WebSphere/AppServer/logs/tracefile
2. Start the WebSphere Administrative console:
   a. Enter `cd /usr/WebSphere/Appserver/bin`
   b. Enter `./adminclient.sh &`

3. In the WebSphere console, select **WCSServlets**, go to the Advanced tab, type `/usr/lpp/CommerceSuite/html/en_US` in the document root field and click **Apply.**
4. Delete the base servlet; in the WAS Console, select **BaseServlet** (under WCSServlets), right click and select **Remove** from the context menu. This is for the first install only; if MPE is already installed delete EMPBaseServlet.
5. In the console, select **WebSphere Commerce Server** and add the following entries to the classpath:
   a. `/usr/lpp/db2_06_01/java/db2java.zip`
   b. `/usr/lpp/db2_06_01/java/runtime.zip`
6. In the WebSphere console, select **default_host** , select the Advanced tab, and add two aliases:
   a. fully qualified hostname
   b. fully qualified hostname:443
7. Install all the MPE servlets at once:
   a. Enter `cd /usr/lpp/CommerceSuite/emp_schema/singlebyte/xml`
   b. Enter `chmod 755 ZapAndLoadXML.sh`
   c. Enter `./ZapAndLoadXML.sh node` where `node` is the name of your WebSphere node (this is usually the name of the machine without the domain name).

A.16 Configure aliases in IBM HTTP server

The Marketplace Edition requires aliases to be added to the IBM HTTP server configuration so that the correct HTML file can be located. To configure the IBM HTTP server:

1. Enter `cd /usr/HTTPServer/conf`
2. Edit the httpd.conf file and add the following aliases:

```
-Alias   /standard /usr/lpp/CommerceSuite/html/en_US/emp/standard
-Alias   /emp /usr/lpp/CommerceSuite/html/en_US/emp
```

A.17 Load default access policies

The Marketplace Edition provides a set of default access policies that you can use with the sample site and also as a start point for customizing your own policies. To load the default policies:

1. Log on as user ID root
2. Enter `cd /usr/lpp/CommerceSuite/emp_schema/singlebyte/membership`
3. Enter `./testinit.sh demomall cn=root ldap_pw`

A.18 Start all the Marketplace Edition components

Once installation is complete you need to start all the Marketplace Edition components to begin testing. The components to start are:

1. IBM HTTP server
2. The LDAP server
3. WCS instance MSER
4. The WebSphere application server called WebSphere Commerce server

Appendix B. Special notices

This publication is intended to help IT architects and IT specialists to design and deploy e-Marketplace applications. The information in this publication is not intended as the specification of any programming interfaces that are provided by WebSphere Commerce Suite. See the PUBLICATIONS section of the IBM Programming Announcements for IBM WebSphere Commerce Suite V4.1 and WebSphere Commerce Suite, Marketplace Edition for more information about what publications are considered to be product documentation.

References in this publication to IBM products, programs or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only IBM's product, program, or service may be used. Any functionally equivalent program that does not infringe any of IBM's intellectual property rights may be used instead of the IBM product, program or service.

Information in this book was developed in conjunction with use of the equipment specified, and is limited in application to those specific hardware and software products and levels.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact IBM Corporation, Dept. 600A, Mail Drop 1329, Somers, NY 10589 USA.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The information contained in this document has not been submitted to any formal IBM test and is distributed AS IS. The use of this information or the implementation of any of these techniques is a customer responsibility and depends on the customer's ability to evaluate and integrate them into the customer's operational environment. While each item may have been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere. Customers attempting to adapt these techniques to their own environments do so at their own risk.

Any pointers in this publication to external Web sites are provided for convenience only and do not in any manner serve as an endorsement of these Web sites.

The following terms are trademarks of the International Business Machines Corporation in the United States and/or other countries:

e (logo)®	Redbooks
IBM ®	Redbooks Logo
RS/6000	S/390
SecureWay	SP
System/390	VisualAge
WebSphere	Wizard
XT	400
Lotus	Domino
eSuite	

The following terms are trademarks of other companies:

Tivoli, Manage. Anything. Anywhere.,The Power To Manage., Anything. Anywhere.,TME, NetView, Cross-Site, Tivoli Ready, Tivoli Certified, Planet Tivoli, and Tivoli Enterprise are trademarks or registered trademarks of Tivoli Systems Inc., an IBM company, in the United States, other countries, or both. In Denmark, Tivoli is a trademark licensed from Kjøbenhavns Sommer - Tivoli A/S.

C-bus is a trademark of Corollary, Inc. in the United States and/or other countries.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and/or other countries.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States and/or other countries.

PC Direct is a trademark of Ziff Communications Company in the United States and/or other countries and is used by IBM Corporation under license.

ActionMedia, LANDesk, MMX, Pentium and ProShare are trademarks of Intel Corporation in the United States and/or other countries.

UNIX is a registered trademark in the United States and other countries licensed exclusively through The Open Group.

SET, SET Secure Electronic Transaction, and the SET Logo are trademarks owned by SET Secure Electronic Transaction LLC.

Other company, product, and service names may be trademarks or service marks of others.

Appendix C. Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this redbook.

C.1 IBM Redbooks

For information on ordering these publications see "How to get IBM Redbooks" on page 475.

- *Patterns for e-business: User-to-Online Buying Pattern using WebSphere Commerce Suite V4.1*, SG24-6156
- *Servlet and JSP Programming with IBM WebSphere and VisualAge for Java,* SG24-5755-00
- *Application Server Solution Guide, Enterprise Edition: Getting Started*, SG24-5320
- *WebSphere V3 Performance Tuning Guide*, SG24-5657
- *WebSphere Scalability: WLM and Clustering Using WebSphere Application Server Advanced Edition,* SG24-6153
- *IBM WebSphere and VisualAge for Java Database Integration with DB2, Oracle, and SQL Server,* SG24-5471
- *IBM WebSphere Performance Pack: Load Balancing with IBM SecureWay Network Dispatcher*, SG24-5858
- *Design and Implement Servlets, JSPs, and EJBs for IBM WebSphere Application Server*, SG24-5754
- *A Secure Way to Protect Your Network: IBM SecureWay Firewall for AIX V4.1,* SG24-5855
- *Understanding LDAP*, SG24-4986
- *WWW Programming: VisualAge for C++ and ST*, SG24-4734
- *Developing an e-business Application for the IBM WebSphere Application Server*, SG24-5423
- *LDAP Implementation Cookbook*, SG24-5110

© Copyright IBM Corp. 2000

C.2 IBM Redbooks collections

Redbooks are also available on the following CD-ROMs. Click the CD-ROMs button at **ibm.com**/redbooks for information about all the CD-ROMs offered, updates and formats.

CD-ROM Title	Collection Kit Number
IBM System/390 Redbooks Collection	SK2T-2177
IBM Networking Redbooks Collection	SK2T-6022
IBM Transaction Processing and Data Management Redbooks Collection	SK2T-8038
IBM Lotus Redbooks Collection	SK2T-8039
Tivoli Redbooks Collection	SK2T-8044
IBM AS/400 Redbooks Collection	SK2T-2849
IBM Netfinity Hardware and Software Redbooks Collection	SK2T-8046
IBM RS/6000 Redbooks Collection	SK2T-8043
IBM Application Development Redbooks Collection	SK2T-8037
IBM Enterprise Storage and Systems Management Solutions	SK3T-3694

C.3 Other resources

These publications are also relevant as further information sources:

- John Barry et al, *Developing Object-oriented Software - An Experienced-Based Approach*, Prentice Hall, 1997, ISBN 0137372485
- E. Gamma, R. Helm, R. Johnson, J. Vlissides, *Design Patterns - Elements of Reusable Object-Oriented Software*, Addison-Wesley, 1995, ISBN 0201633612
- C. Alexander, S. Ishikawa, M. Silverstein, M. Jacobson, I. Fiksdahl-King, S. Angel, *A Pattern Language*, Oxford University Press, 1977, ISBN 0195019199
- "Enterprise Solutions Structure" in *IBM Systems Journal, Volume 38, No. 1, 1999*, available at `http://www.research.ibm.com/journal/sj38-1.html`
- F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad, M. Stahl, *Pattern-Oriented Software Architecture - A System of Patterns*, Wiley, 1996, ISBN 0471958697
- Flanagan, David, Jim Farley, William Crawford and Kris Magnusson, *Java Enterprise in a Nutshell*, O'Reilly & Associates, Inc. 1999, ISBN 1565924835
- L. Bass, P. Clements, R. Kazman, *Software Architecture in Practice*, Addison Wesley, 1998, ISBN 0201199300

- Booch, Grady, *Object-Oriented Analysis and Design with Applications* (Addison-Wesley Object Technology Series), Addison-Wesley, 1994, ISBN 0805353402
- Jacobson, Ivar, *Object-Oriented Software Engineering; A Use Case Driven Approach,* Addison-Wesley, 1992, ISBN 0201544350
- Rumbaugh, James et al, *Object-Oriented Modeling and Design*, Prentice Hall, 1991, ISBN 0136298419
- Fowler, Martin, Kendall Scott (Contributor) and Ivar Jacobson, *UML Distilled; Applying the Standard Object Modeling Language*, Addison-Wesley, 1997, ISBN 0201325632
- Maggie Archibald and Mike Schlosser, *Designing e-business Solutions for Performance* white paper, available at: `http://www.ibm.com/software/developer/library/patterns/performance.html`
- *The Java HotSpot Performance Engine Architecture* white paper, available at: `http://java.sun.com/products/hotspot/whitepaper.html`
- *IBM Application Framework for e-business* white papers available at: `http://www.ibm.com/software/ebusiness/`
- Flanagan, David, *JavaScript: The Definitive Guide*, Third Edition, O'Reilly & Associates, Inc., 1998, ISBN 1565923928
- Maruyama, Hiroshi, Kent Tamura and Naohiko Uramoto, *XML and Java: Developing Web Applications,* Addison-Wesley, 1999, ISBN 0201485435
- Nagaratnam, Nataraj et al, *Security Overview of IBM WebSphere Standard/Advance 3.02,* IBM white paper, available at: `http://www.ibm.com/software/webservers/appserv/whitepapers.html`
- Shane Claussen and Mike Conner, *Developing Dynamic Web Sites Using the WebSphere Application Server*, available at:

  `http://service2.boulder.ibm.com/devcon/news0399/artpage2.htm`

C.4 Referenced Web sites

These Web sites are also relevant as further information sources:

- `http://www.ibm.com/software/developer/web/patterns` developerWorks: Patterns for e-business
- `http://www.ibm.com/software/ebusiness` IBM Application Framework
- `http://www.research.ibm.com/journal/sj38-1.html/` IBM Systems Journal - Volume 38, No 1

- `http://www.ibm.com/software/webservers/commerce/` IBM Software E-Commerce Overview
- `http://java.sun.com/beans/index.htm` Sun’s JavaBeans page
- `http://www.ibm.com/software/webservers/appserv/doc/v30/se/web/doc/begin_here/index.html` IBM WebSphere Apllication Server V3 Documentation
- `Http://www-4.ibm.com/software/webservers/siteanalyzer/doc/help/sacontents.html` *IBM WebSphere Site Analyzer Guide*
- `http://www.cert.org` CERT Coordination Center
- `http://java.sun.com/products/jsp/` Sun’s JavaServerPages technology page
- `http://www.econ.jhu.edu/People/Fonseca/walras/hardy.htm` Leon Walras and Walrasian Economics
- `http://www.ibm.com/software/webservers/commerce/` IBM Software E-Commerce overview
- `http://www.sun.com/xml/` Sun XML Welcome page
- `http://java.sun.com/products/servlet/` Sun Java Servlet Technology page
- `http://java.sun.com/products/ejb/` Sun Enterprise JavaBeans Technology page

How to get IBM Redbooks

This section explains how both customers and IBM employees can find out about IBM Redbooks, redpieces, and CD-ROMs. A form for ordering books and CD-ROMs by fax or e-mail is also provided.

- **Redbooks Web Site** **ibm.com**/redbooks

  Search for, view, download, or order hardcopy/CD-ROM Redbooks from the Redbooks Web site. Also read redpieces and download additional materials (code samples or diskette/CD-ROM images) from this Redbooks site.

  Redpieces are Redbooks in progress; not all Redbooks become redpieces and sometimes just a few chapters will be published this way. The intent is to get the information out much quicker than the formal publishing process allows.

- **E-mail Orders**

  Send orders by e-mail including information from the IBM Redbooks fax order form to:

	e-mail address
In United States or Canada	pubscan@us.ibm.com
Outside North America	Contact information is in the "How to Order" section at this site: http://www.elink.ibmlink.ibm.com/pbl/pbl

- **Telephone Orders**

United States (toll free)	1-800-879-2755
Canada (toll free)	1-800-IBM-4YOU
Outside North America	Country coordinator phone number is in the "How to Order" section at this site: http://www.elink.ibmlink.ibm.com/pbl/pbl

- **Fax Orders**

United States (toll free)	1-800-445-9269
Canada	1-403-267-4455
Outside North America	Fax phone number is in the "How to Order" section at this site: http://www.elink.ibmlink.ibm.com/pbl/pbl

This information was current at the time of publication, but is continually subject to change. The latest information may be found at the Redbooks Web site.

IBM Intranet for Employees

IBM employees may register for information on workshops, residencies, and Redbooks by accessing the IBM Intranet Web site at http://w3.itso.ibm.com/ and clicking the ITSO Mailing List button. Look in the Materials repository for workshops, presentations, papers, and Web pages developed and written by the ITSO technical professionals; click the Additional Materials button. Employees may access MyNews at http://w3.ibm.com/ for redbook, residency, and workshop announcements.

IBM Redbooks fax order form

Please send me the following:

Title	**Order Number**	**Quantity**

First name	Last name	
Company		
Address		
City	Postal code	Country
Telephone number	Telefax number	VAT number

☐ Invoice to customer number

☐ Credit card number

Credit card expiration date	Card issued to	Signature

We accept American Express, Diners, Eurocard, Master Card, and Visa. Payment by credit card not available in all countries. Signature mandatory for credit card payment.

Abbreviations and acronyms

IBM	International Business Machines Corporation
ITSO	International Technical Support Organization
B2B	Business to Business
B2C	Business to Consumer
RFQ	Request for Quote
RFP	Request for Proposal
PKI	Public Key Infrastructure
DNS	Domain Name Service
SSL	Secure Sockets Layer
ESS	Enterprise Solution Structure
LDAP	Lightweight Directory Access Protocol
DMZ	Demilitarized Zone
WAS	WebSphere Application Server
SLIP	Serial Line Internet Protocol
MTU	Maximum transmission protocol
EGP	Exterior Gateway Protocol
ICMP	Internet Control Message Protocol
PPP	Point to Point Protocol
SMTP	Simple Mail Transfer Protocol
UDP	User Datagram Protocol
CPU	Central Processing Unit.
SSI	Server-side include
HTTP	Hyper Text Transfer Protocol
API	Application Programming Interface
TCP	Transmission Control Protocol
IP	Internet Protocol
ISP	Internet Service Provider
HTTPS	Secure Hypertext Transfer Protocol
TCP/IP	Transmission Control Protocol/Internet Protocol
SMTP	Simple Mail Transfer Protocol
FTP	File Transfer Protocol
IGP	Interior Gateway Protocol
BGP	Border Gateway Protocol
MSS	Maximum Segment Size
IC	Interaction controller
MVC	Model-View-Controller
OF	Overrideable function
IDE	Integrated Development Environment
UML	Unified Modelling Language
SLA	Service Level Agreement
TSM	Tivoli Storage Manager
WTE	WebSphere Test Environment
QoS	Quality of Service

HTTPS	Secure Hypertext Transfer Protocol
WCS	WebSphere Commerce Suite
PO	Purchase Order
XML	Extensible markup language
XSL	eXtensible stylesheet language
DTD	Document type definition
SGML	Standard generalized markup language

Index

© Copyright IBM Corp. 2000

IBM Redbooks review

Your feedback is valued by the Redbook authors. In particular we are interested in situations where a Redbook "made the difference" in a task or problem you encountered. Using one of the following methods, **please review the Redbook, addressing value, subject matter, structure, depth and quality as appropriate.**

- Use the online **Contact us** review redbook form found at **ibm.com**/redbooks
- Fax this form to: USA International Access Code + 1 845 432 8264
- Send your comments in an Internet note to redbook@us.ibm.com

Document Number **Redbook Title**	SG24-6158-00 e-Marketplace Pattern Using WebSphere Commerce Suite, Marketplace Edition Patterns for e-business Series
Review	
What other subjects would you like to see IBM Redbooks address?	
Please rate your overall satisfaction:	O Very Good O Good O Average O Poor
Please identify yourself as belonging to one of the following groups:	O Customer O Business Partner O Solution Developer O IBM, Lotus or Tivoli Employee O None of the above
Your email address: The data you provide here may be used to provide you with information from IBM or our business partners about our products, services or activities.	 O Please do not use the information collected here for future marketing or promotional contacts or other communications beyond the scope of this transaction.
Questions about IBM's privacy policy?	The following link explains how we protect your personal information. **ibm.com**/privacy/yourprivacy/

IBM

Redbooks

e-Marketplace Pattern Using WebSphere Commerce Suite, Marketplace Edition Patterns for e-business Series

(1.0” spine)
0.875”<->1.498”
460 <-> 788 pages

e-Marketplace Pattern Using WebSphere Commerce Suite, Marketplace Edition

Patterns for e-business Series

Understanding the e-Marketplace Pattern

Guidelines for building an e-Marketplace solution

Implementation examples

The Patterns for e-business are a group of proven, reusable assets that can help speed the process of developing applications. The pattern discussed in this book, the Business-to-Business e-Marketplace Pattern, is an emerging pattern that allows the development of e-Marketplace hub applications which bring multiple buyers and sellers together in a way that provides efficient electronic trading of goods and services. Subsets of the application topologies for the Business-to-Business e-Marketplace Pattern are used to describe different parts of the full marketplace topology, and they represent increasing levels of complexity, functionality and integration in the topology, ranging from a simple e-Marketplace to a fully integrated e-Marketplace.

Part 1 of the redbook describes the nature of e-Marketplaces and guides you through the process of choosing an application and runtime topology to deliver the desired market functionality.

Part 2 of the redbook provides a set of guidelines for building your e-Marketplace application. These guidelines include discussion of performance, technology options, application design, application development, systems management, and security.

Part 3 of the redbook describes, using the standard sample application, the functions available in the WebSphere Commerce Suite, Marketplace Edition for AIX.

INTERNATIONAL TECHNICAL SUPPORT ORGANIZATION

BUILDING TECHNICAL INFORMATION BASED ON PRACTICAL EXPERIENCE

IBM Redbooks are developed by the IBM International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.

For more information:
ibm.com/redbooks

SG24-6158-00 ISBN 0738417912