P-1538

SECURE AND FLEXIBLE DATA ACCESS THROUGH ENTERPRISE COLLABORATION VIA WEB SERVICES

By
P.RAGUPATHI
Register No: 71203405011

Of
Kumaraguru college of Technology,
Coimbatore-641006

A PROJECT REPORT
Submitted to the
FACULTY OF INFORMATION AND COMMUNICATION ENGINEERING

In partial fulfillment of the requirements
for the award of the degree

Of
MASTER OF ENGINEERING
IN
COMPUTER SCIENCE AND ENGINEERING

JUNE, 2005

BONAFIDE CERTIFICATE

Certified that this project report titled **Secure and Flexible Data Access through Enterprise Collaboration via Web Services** is the bonafied work of **P.RAGUPATHI** who carried out the research under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form part of any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

[illegible] — Internal Guide
[illegible] — Head of the Department

Submitted for the University Examination held on 22/6/05

[illegible] — Internal Examiner
[illegible] 22/6 — External Examiner

ABSTRACT

Today, the adaptation of the internet and enabling internet based applications has created a world of discrete business applications, which co exists in the same technology space but without interacting with each other. The increasing demands of the industry for enabling Business-to-Business and Inter process application communication has let to a growing requirement for service oriented architecture. Enabling service oriented applications facilitates the exposure of business applications as service components enabled business applications from other organizations to link with these services for application interaction and data sharing without human intervention. By leveraging this architecture, it also enables inter operability between business applications and processes.

By adopting web technologies, the service oriented architecture model facilitates the delivery of services over the internet by leveraging standard technologies such as XML. It uses the platform neutral standards by exposing the underlying application components and making them available to any application, any platform, or any device, and at any location.

With the web services, we can enable Business-to-Business collaboration in a platform independent manner, but the lack of security is the major drawback. This project is to establish a framework to establish plug and play collaboration, which provides secure and flexible data access among Business-to-Business collaboration by combining Distributed Role Based Access Control and Public Key Infrastructure with hierarchical model of certification authorities.

ஆய்வுச் சுருக்கம்

இன்றைய கால கட்டத்தில் இணைய வலைத் தொழில் நுட்பத்தின் வளர்ச்சியும் அதன் சார்ந்த பயன்பாடுகளும், வர்த்தக நிறுவனங்கள் தங்கள் தொழில் சார்ந்த பயன்பாட்டுக்கு இணையத்தை பயன்படுத்திக் கொள்ள வழிவகை செய்கிறது. இருந்த பொழுதிலும் நிறுவனங்களின் செயல்பாடுகள் ஒன்றோடென்று தொடர்பு கொள்வதில்லை மாற்றமடைந்து வரும் வர்த்தக முறைகளால் இன்றைய வர்த்தக நிறுவனங்கள் இணையத்தின் மூலம் பி2பி எனப்படும் கூட்டு வணிக முறையில் ஈடுபடவேண்டிய சூழ்நிலையில் உள்ளனர். இந்தக் கூட்டு வணிக முறையை அமல்படுத்த நாம் இணையத்தில் ஒரு புதிய கட்டமைப்பு உருவாக்க வேண்டியுள்ளது. சேவையை மையமாகக் கொண்ட கட்டமைப்புகளை உருவாக்குவதன் மூலம் நாம் இந்தத் தேவையை பூர்த்தி செய்யலாம். இந்தக் கட்டமைப்புகள் ஒரு பயன்பாட்டுக்கான பிஸினஸ் அப்ளிகேஷன்களை சர்விஸ் காம்பொன்ட்டுகளாக வெளிக்கொணர்கிறது, இதன் மூலம் இரு நிறுவணங்களுக்கு இடையிலான தொடர்பையும் தகவல் பரிமாற்றங்களையும் மனிதத் தலையீடு இல்லாமல் செய்ய முடியும்.

இன்றைய இணையத் தொழில் நுட்பமும் சேவையை சமையமாகக் கொண்ட கட்டமைப்பும இணைத்து உருவானதே வெப் சர்வீஸ் எனப்படுவதாகும். இந்த வெப்சர்வீஸ் XML மற்றும் SOAP ஆகியவற்றை பயன்படுத்தி நிறுவனங்களுக்கு இடையிலான தகவல் தொடர்பை அவை நிறுவியிருக்கும் மென்பொருள் மற்றும் சாதனங்குளுக்கு அப்பாற்பட்டு ஏற்படுத்த முடியும்.

இந்த ஆய்வின் நோக்கமே வெப் சர்வீஸ் பயன்படுத்தி வர்த்தக மாறுதல்களுக்கு தகுந்தவாறு, நிறுவனங்களுக்கு இடையே பி2பி தொடர்பை உருவாக்குவதே ஆகும். இந்தக் கட்டமைப்பு பாதுகாப்பு கொடுப்பதற்கு Distributed Role Based Access Control மற்றும் Public Key Infrastructure ஆகியவற்றை இணைத்து பயன்படுத்துகிறது.

ACKNOWLEDGEMENT

I am extremely thankful to **Dr.K.K.Padmanabhan B.Sc. (Engg.), M.Tech, Ph.D.,** Principal, Kumaraguru College of Technology who has given me his valuable guidance and an efficient infrastructure that aided me during the course of my education.

With profound sense of gratitude and regards, I acknowledge with great pleasure the guidance and support extended by **Prof.S.Thagasamy BE.(Hons), Ph.D.,** head of the Department of Computer science and Engineering for his encouragement and moral support throughout the course of the project.

I would like to express my indebted gratitude and whole hearted thanks to my guide **Mr.R.Dinesh MS,** Assistant Professor Department of Computer science and Engineering, Kumaraguru College of technology, for Constant Encouragement throughout the course of work. I also like to express my special thanks to all the staffs and lab technicians in the department of Computer science and Engineering who helped me for the successful completion of the project.

Finally and most importantly, I would like to thank my **beloved parents, friends and family members** for giving me endless encouragement and necessary help, during the course of my project.

TABLE OF CONTENTS

CHAPTER 1
INTRODUCTION

Recently there has been much industry uproar regarding the dawning of the age of web services. However, regardless of how overrated or underrated, web services currently are in the eye of the technical media. The concept here to say that web services will most likely be the future of inter-enterprise electronic commerce and collaboration. At a high level, the term web services refer to the ability to publish, discover or invoke a set of services in a platform independent manner using XML and standard and web based protocols for transport.

This independence is the missing factor in inter enterprise computing today. Businesses find it hard electronically inter operate in a control, secure manner. Much of this difficulty is caused by differences in platforms and transports, which necessitate tedious conversion of data and functionality in order for two businesses with different architectures to interoperate.

In early days the CORBA is used for inter enterprise computing. Its disadvantage is that it's lack of flexibility and its inability to handle asynchronous messaging in a straight forward manner. Another possible solution would be to invoke a custom servlet published by a quote provider, passing it the necessary data as part of a HTTP post request. This would be a more light weight solution and infact was the way much Business-to-Business electronics commerce was conducted pre-web services. Its disadvantages is that it is completely proprietary, if one person wanted to pragmatically use the services of many external parties; he would need to learn many different proprietary communication protocols or ways to format the http post data he was sending. Similarly he would need to write custom code to contact each service provider again this is not an ideal solution.

What if there was a standard way of transporting data or invoking a service on a remote system. What if the standard was relatively light weight, platform neutral and could be used over just above any transport, including

the most prolific of all transports, HTTP. This would be ideal and it would be a first step towards a world of "hands-off" electronic commerce. This is what web services seek to achieve.

The emergence of web services introduces a new paradigm for enabling the exchange of information across the internet based on open internet standards and technologies using industry standards web services encapsulates application and publish them as services. This services deliver XML based data on the wire and expose it for use on the internet, which can be dynamically located, subscribed, and accessed using a wide range of computing platforms, handheld devices, applications and so on. Due to the flexibility of using open standards and protocols, and also facilitates enterprise application integration, B2B integration and A2A communication across the internet and corporate intranet. In organizations heterogeneous applications and distributed application architectures, the introduction of web services standardizes the communication and enables inter operability of applications based on different programming languages residing on different platforms.

1.1. Web Services

Web services are based on the concept of service oriented architecture. SOA is the latest evolution of distributed computing which enables software components including application functions, objects and processes from different systems, to be exposed as services. According to Gartner research "Web services loosely coupled software components delivered over internet standard technologies". In short, web services are self describing and modular business applications that expose the business logic as services over the internet through programmable interfaces and using internet protocols for the purpose of providing ways to find, subscribe, and invoke those services.

Based on XML standards web services can be developed as loosely coupled application components using any programming languages, any protocols, or any platform. This facilitates delivery business applications as a service accessible to anyone, any time, at any location, and using any platform. Web services enable businesses to communicate, collaborate, and conduct business transactions using light weight infra structure by adopting an XML based data exchange format and industry standard delivery protocols.

The basic characteristics of a web services application model are as follows,

- Web services are based on XML messaging, which means the data exchanged between the web service provider and the user are defined in XML
- Web services provide a cross platform integration of business application over the internet.
- To built web services, developers can use any common programming language such as Java, C, C++.
- Web services are not meant for handling presentations like Html context – it is developed to generate for uniform accessibility through any software application, any platform, and any device.

Traditionally, web applications enable interactions between an end user and a web site, while web services are service oriented and enable application to application communication over the internet and easy accessibility to heterogeneous applications and devices. The following are the major technical reasons for choosing web services over web applications,

- Web services can be invoked through XML based RPC mechanisms across firewalls
- Web services provide a cross platform, cross language solution based on XML messaging
- Web services facilitate ease of application integration using a light weight infrastructure without affecting scalability.
- Web services enable interoperability, heterogeneous applications.

1.2. Web services technologies

There is nothing mystical about web services. They are merely another mechanism for building distributed systems some what more flexible than previous mechanisms. Typically, for any general purpose distributed system to work, it must contain atleast three components,

- A naming service or registry
- A mechanism to describe the interfaces for the service provided
- A methodology for actual transporting the request, along with the interface description to the service provider.

Here is the list of three fundamental technologies in the web service world,

- UDDI, this is a naming service their service providers can advertise their services to prospective clients. UDDI technologies are used as a clearinghouse for interface information, along with other relevant business information regarding providers of web services. A service provider publishes his service in a UDDI registry. A client then looks up the service provider in the registry, enumerates the service, chooses the one it wants, and asks the registry for the location of the interface description (WSDL) that describes the service in question.

- WSDL is the interface description. WSDL's XML based grammar is the mechanism for the service description. It describes the capabilities of a particular web service. The service supports a synchronous request response model; WSDL contains the description of the arguments or parameters for each request, instructions for how to marshal the arguments for transport, and the structure and marshalling information for any response.

- SOAP, the technology is used for the actual invocation of web services. It provides a platform neutral, XML based mechanism to request services or send messages from one application to another. SOAP supports either synchronous or asynchronous messaging.

1.3. Web Services Architecture

Typical web services architectural models consist of three logical components as core building blocks mapping the operational roles and relationships of a web services environment.

Services Container acts as the web services run time environment and hosts the service provider. It defines the service deployment and service administration in addition it also handles the registration of the service description the service registries.
Services registry the services registry hosts the published services and acts as the broker providing the facility to publish and store the description of web services registered by the service providers in addition, it defines a common access mechanisms for the service requestors for locating the registered services.
Services Delivery it acts as the web services client run time environment by looking up the services registries to find the required services and invoking them from the service providers.

Web Services Architecture and its core building blocks

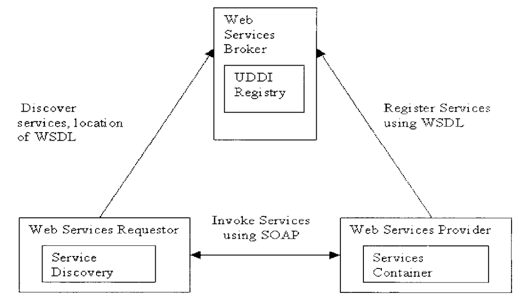


Figure 1.1 – Web Service Architecture

CHAPTER 2
LITERATURE SURVEY

2.1. Authorization and Attribute Certificates for Widely Distributed Access Control -*William Johnston, Srilekha Mudumbai, and Mary Thompson*

In this article the authors describe a system whose purpose is to explore the use of certificates for the distributed management of access rights for resources that have multiple, independent, and geographically dispersed stakeholders. The stakeholders assert their use-conditions in authorization certificates and designate those trusted to attest to the corresponding attributes. These use-conditions implicitly define access groups through their requirement for certain attributes. All use-conditions must be satisfied simultaneously, so the actual access group is the intersection of all of the groups. A policy engine collects the use-condition certificates and attributes certificates when a user attempts to access a particular resource. If all of the use-conditions are met, a capability is generated for the resource. The policy engine can provide several different policy models depending on whether any relationship is established among the use conditions.

2.2. Managing access in extended enterprise networks -*Karl Furst, Thomas Schmidt, and Gerald Wippel, IEEE Internet Computing.*

Plug and play collaboration between independent enterprises with different core competencies will be key for enabling agile responses in a turbulent market environment. Current E-Business technologies use only a first step toward realizing this vision. Successful next generation business to business software must therefore enable effortless and secure creation of agile networks from independent enterprises. Xml based web services allow enterprises to provide internet based application components to authorized business partners, but the lack of security conventions is a major drawback to existing web service approaches.

To integrate components is extended enterprises (group of partners working as one) administrators need tools that go beyond the simple security mechanisms that suffice for general B2B process. In the enterprise cooperation through web services, authentication and authorization should have to be focused mainly, because at the beginning information must be secured against unauthorized access in order to establish trust. This method combines role based access control and public key infrastructure.

2.3. Building Trust in e-Commerce - *Atif, Y, IEEE Internet Computing.*

To increase confidence in commercial transactions over the Web where the transacting parties are invisible to each other, we need not just new protocols but also new transaction processes. One solution is to enlist a third party, referred to here as a trust service provider (TSP), to act as an Internet-based intermediary that assumes responsibility for a smooth transaction. The TSP is known and trusted by both customer and merchant and makes purchases on behalf of the one and conveys the goods on behalf of the other. The article describes a proposal for a trust Web model based on a distributed search algorithm and a network of trusted intermediaries that can establish a trusted channel through which terminal transacting parties deal virtually directly and risk-free with each other.

2.4. Role based access control models

This article introduces a family of reference models for role-based access control (RBAC) in which permissions are associated with roles, and users are made members of appropriate roles. This greatly simplifies management of permissions. Roles are closely related to the concept of user groups in access control. However, a role brings together a set of users on one side and a set of permissions on the other, whereas user groups are typically defined as a set of users only. The basic concepts of RBAC originated with early multi-user computer systems. The resurgence of interest in RBAC has been driven by the need for general-purpose customizable facilities for RBAC and the need to manage the administration of RBAC itself. As a consequence RBAC facilities range from simple to complex. This article describes a novel framework of reference models to

systematically address the diverse components of RBAC, and their interactions. To understand the various dimensions of RBAC authors define a family of four conceptual models. The relationship between these four models is shown in Figure 1(a) and their essential characteristics portrayed in Figure 1(b). RBAC0, the base model, is at the bottom, indicating that it is the minimum requirement for any system that professes to support RBAC. RBAC1 and RBAC2 both include RBAC0, but add independent features to it. They are called advanced models. RBAC1 adds the concept of role hierarchies (situations where roles can inherit permissions from other roles). RBAC2 adds constraints (which impose restrictions on acceptable configurations of the different components of RBAC). RBAC1 and RBAC2 are incomparable to one another. The consolidated model, RBAC3, includes RBAC1 and RBAC2 and, by transitivity, RBAC0.

2.5. Distributed role based access control for Dynamic Coalition Environments- *Eric Frudenthal, Tracy Pesin, and Vijay Karamcheti, IEEE Internet Computing.*

Distributed Role-Based Access Control (DRBAC) is a scalable, decentralized trust management and access control mechanism for systems that span multiple administrative domains. DRBAC utilizes public key infrastructure (PKI) identities to define trust domains, roles to define controlled activities, and role delegation across domains to represent permission to these activities. The mapping of controlled actions to roles enables their namespaces to serve as policy roots. DRBAC combines the advantage of role based access control and trust management systems to create a system that offers both administrative ease and decentralized, scalable implementation. DRBAC distinguishes itself from previous approaches by providing three features: (1) third party delegation of roles from outside a domain's namespaces, relying upon an explicit delegation of assignment; (2) modulation of transferred permissions using scalar valued attributes associated with roles; and (3) continuous monitoring of trust relationships over long-lived interactions.

CHAPTER 3
LINE OF ATTACK

Plug and play collaboration between independent enterprises with different core competencies will be key for enabling agile responses in a turbulent market environment. Current E-Business technologies use only a first step toward realizing this vision. Successful next generation business to business software must therefore enable effortless and secure creation of agile networks from independent enterprises.

Xml based web services allow enterprises to provide internet based application components to authorized business partners, but the lack of security conventions is a major drawback to existing web service approaches. To integrate components is extended enterprises (group of partners working as one) administrators need tools that go beyond the simple security mechanisms that suffice for general B2B process.

In the enterprise cooperation through web services, authentication and authorization should have to be focused mainly, because at the beginning information must be secured against unauthorized access in order to establish trust. The project is to develop an authentication and authorization system in a secure and flexible manner. This method combines distributed role based access control and public key infrastructure.

The vulnerabilities when attempting to secure a web services application.

- The transport between client and server is done via HTTP and requests and responses are encoded using SOAP / XML which is easily viewable or changeable with a common text editor.
- The WSDL Meta data necessary to invoke the service is usually available to the general public, either as an adjunct to the web services container or as part of a reference from a UDDI directory entry.
- Propagation of security identity or credentials between the client and server is not and can be quite problematic, especially in work flow

based architecture. There are emerging standards for credential propagation message integrity and confidentiality, but these are still disparate and somewhat vendor – specific.

The following are some aspects of web service security.

- Authentication with web services
- Securing the client/server connection
- Connecting web services via severe tunneling over SSL
- Implementing declarative authorization for web services.
- Implementing programmatic authorization for web services.
- Confidentially and integrity of payload information.

3.1. DRBAC – EE Approach:

This extended enterprise approach uses attribute based authentication with a PMI and uses XML to handle the data needed. The necessary attributes are included in the header of the SOAP message.

- Enterprises that will act as a business partners.
- Projects to be carried out in the scoop of the extended enterprise.
- Legally defined shared and project specific roles.

The extended enterprise project administrator centrally controls project in the security domain.

3.2. Requirements for extended enterprise:

Extended enterprises unit project and opportunity based business that must react quickly to changes in the market or the available technology.

- The collaborating partners must reach general agreement on a project frame work and one. Enterprise must lead the network.
- This leader (often describes simply as an enterprise whose expanded boundaries incorporate its business partners) – provide specific resources for operating the extended enterprise.
- As a matter of course any one of the partners can be involved in multiple extended enterprises for example two enterprises might simultaneously cooperate in one business segment and compete in another.

To develop information technology systems that can automate extended enterprise business process, a system must be both secure and flexible.

Security is the most important element in business networks, particularly in distributed environments where data is exchanged over the internet. Although there are manifold security issues that effect extended enterprises. I focus on authorization and authentication measures because at the beginning information must be secured against unauthorized access in order to establish trust, an essential element in B2B transactions. Unfortunately the security requirements conflict with the need for flexibility.

Flexibility enables business networks to react quickly to opportunities. That requires fast network setup and simple mechanisms for incorporating new partners as needed. To foster highly automated business process, each enterprise must have access to all the internet application components it requires.

3.3. Security approaches:

The authentication and authorization system combines three security approaches with modifications and enhancements. RBAC, PKI and PMI

(I). Role based access control:

RBAC provides a suitable means for simplifying privilege administration the core idea is not to directly grand specific users the right to access resources but to introduces an abstraction layer in which roles, such as sales person are each assigned privileges. (A sales person can access the customer data base for example) the administrator then Assigns roles to individual users. There are different kinds of RBAC related activities.

- Assigning roles to roles to establish a hierarchy.
- Assigning privileges to roles
- Assigning users to roles

In the context I use distributed role based access control which would allow management to delegate administration to individual enterprises in an

extended enterprise. DRBAC is a scaleable, decentralized trust management and access control mechanism for systems that spanned multiple administrative domains. DRBAC utilizes PKI identities to define trust domains: roles to define controlled activities and role delegation across domains to represent permissions to these activities. The mapping of controlled actions to roles enables their name spaces to serve as policy roots.

DRBAC distinguishes itself from previous approaches by providing three features

- Third party delegation of roles from outside a domains name space, relying upon an explicit delegation of assignment.
- Modulation of transferred permissions using scalar valued attributes associated with roles.
- Continuous monitoring of trust relationship over long lived interactions.

DRBAC combines the advantages of role based access control and trust management system. To create a system that offers both administrative ease and a decentralized scalable implementation. DRBAC does not distinguish between owners of resources protected by the system and principles attempting to access them, both are termed entities and represented by unique PKI public identity

(II). PKI and PMI:

PKI is often used to establish security in distributed environments. PKI recovers both a private key, which is kept secret and a public key, which must be distributed. Another central element of PKI is the certification authority, which can verify the identity of a specific user or entity.

The privilege management infrastructure lets the system control authorization using attribute certificates to enable privileges, roles and delegation mechanisms. PMI defines an attribute authority which is similar to

but logically independent of the CA because the creation and maintenance of an identity can often should be separated from authorization.

3.4. Administration:

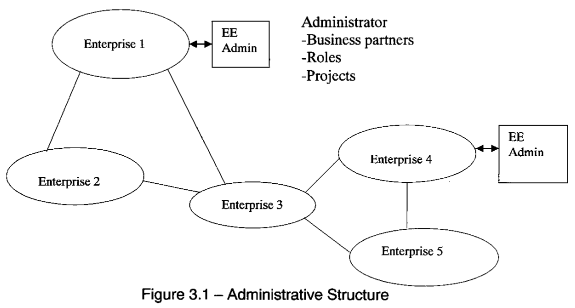


Figure 3.1 – Administrative Structure

Figure shows to extended enterprises each with a single leader that controls resources and operations. Automated role translation between internal and extended enterprise ontologies is extremely important. So the project has to develop an extended enterprise role translation technique that simplifies administration and increases flexibility. The key is to distinguish internal from extended enterprise roles and distribute administrative responsibilities accordingly.

- The extended enterprise project coordinator identifies the project members, selects the extended enterprise can have roles and defines new roles as needed.
- The project managers of all participating enterprises assign internal users and roles to the extended enterprise
- The security administrators of all participating enterprises setup access authorization to services needed for extended enterprise roles.

3.5. Access management:

The access management component is central to our approach. It is used to route internal resource access to external resources and to handle external calls for internal resources. The components main responsibilities are

- Verifying privileges for internal and extended enterprise roles.
- Translating ontologies for roles and processes
- Logging requests for documentation, data mining and legal actions.
- Billing

3.6. Prototype design:

Each enterprise consists of

- Set of components to run an application
- An access management component
- Certification authority to ensure identity
- Attribute authority to ensure roles

I implement an authentication and authorization system in an extended enterprise which consists of two enterprises involving in a distributed project development.

In this infrastructure each enterprise contains its own certification authority to define the identities of users and an attribute authority to define the roles of the participating entities. There is an extended enterprise certification authority to define the identity of each enterprises participating in the collaboration.

The collaboration diagram shows the inter enterprise collaboration between two enterprises

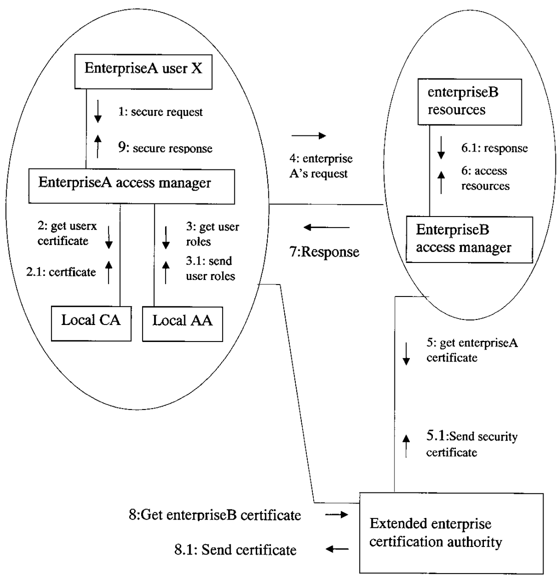


Figure 3.2 – Collaboration of Enterprises

CHAPTER 4

SAMPLE APPLICATION

To demonstrate a typical business-to-business process application, I have implemented a simple supply chain management operation for electronic goods selling which is running between the retailer system and the manufacturing system. The following document presents a high level definition of a Supply Chain Management (SCM) application in the form of a set of Use Cases. The application being modeled is that of a Retailer offering Consumer electronic goods to Consumers; a typical B2C model. To fulfill orders the Retailer has to manage stock levels in warehouses. When an item in stock falls below a certain threshold, the Retailer must restock the item from the relevant Manufacturer's inventory (a typical B2B model). In order to fulfill a Retailer's request a Manufacturer may have to execute a production run to build the finished goods. In the real world, a Manufacturer would have to order the component parts from its suppliers. For simplicity in this application, I assume this is a manual process.

4.1. Overall Non-functional Requirements and Assumptions

In order to simplify the design, facilitate delivery of a demonstration application and allow the Working Group to concentrate on Web services and the implementation, the following requirements and assumptions have been defined:

1. A Retailer will have more than one warehouse that it owns.
2. There will be exactly three manufacturers (Brand1, Brand2, and Brand3).
3. Each manufacturer supplies exactly three products (TV, DVD, video camera). Hence there are nine valid products to be offered by a Retailer.
4. The warehouse stocks all nine products.
5. For demo purposes there will be one invalid product Brand4 TV. The product will be visible in the Retailer's catalog but is not stocked (or recognized) by the warehouse.

6. An order may contain multiple line items, where each line item relates to a specific product and quantity required. A product shall not appear more than once in an order.
7. There are no minimum order quantities, and quantities express units of one (true for both Consumer to Retailer and Retailer to manufacturer).
8. Partial shipments of a single product are not supported; either the required quantity of a product in a line item can be fulfilled in full or none are.
9. The requested quantity of a product must be shipped by a single warehouse, or none are shipped i.e. it is not possible to split the shipment of a product across warehouses.
10. Back orders are not supported; either the required quantity of product can be fulfilled in full by a single warehouse (points 7 and 8) or that line item is rejected.
11. The Consumer's information (payment details, address, etc.) are known to the Retailer system via an implicit logon when the demo starts.
12. Payment is not demonstrated, it is assumed that a Consumer has pre-registered credit card details and billing happens out of band.
13. The start of each purchase use case assumes state is set back to predefined values i.e. predefined stock levels, min/max levels, etc.
14. It is assumed that all implementers will implement all use cases in the Retailer and Manufacturing Systems i.e. 1 Retailer with three warehouses (A, B, and C), and three Manufacturers (Brand1, Brand2, and Brand3.)
15. Only implementation team sanctioned implementations can be configured in this demo i.e. the use cases and demo system do not provide a means for third parties to plug in their implementations.
16. A manufacturer will always ship the requested number of a product to a warehouse i.e. we assume it can always manufacture the required amount.
17. When a purchase request brings a warehouse quantity to below a certain level, the warehouse makes a request of the appropriate manufacturer for more goods.

4.2. Use case diagram

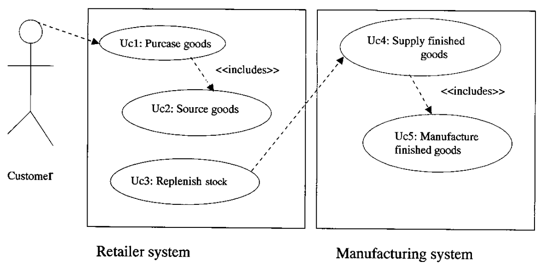


Figure 4.1 Use Cases

4.2.1. UC1: Purchase Goods

Goal of Use Case:	A Consumer goes to the Retailer web site with the intent of purchasing Consumer electronic products.
Preconditions:	**1**. Product Catalog Exists **2**. All state (warehouse levels etc) set back to predefined values **3**. Payment and address details for Consumer are known
Success Post Conditions:	**1**. At least one product is shipped **2**. The Consumer is returned a Confirmation page outlining which products will be shipped **3**. The Retailer has requested the warehouses to ship the available goods. **4**. Payment from the Consumer's credit card is triggered.
Failed Post Conditions:	The Consumer is returned an error stating that none of the items in the order can be fulfilled.
Actors:	Retailer System, Demo System, Consumer

Triggers:	This process is started by the Consumer (human interaction)

4.2.2. UC2: Source Goods

Goal of Use Case:	To locate ordered goods in a warehouse and request shipment
Preconditions	: none
Success Post Conditions:	1. For each line item in the order, a warehouse is selected that has the available quantity and that warehouse ships the goods. 2. For line items that are accepted, the inventory levels in the shipping warehouse for that product are decreased by the quantity in the line item.
Failed Post Conditions	: There is no stock availability in any Warehouse for all of the line items in the order.
Actors	: Retailer System
Triggers	: Receipt of order from Consumer.

4.2.3. UC3: Replenish Stock

Goal of Use Case	: The Retailer System orders goods from a manufacturer to replenish stock for a particular product in a particular warehouse.
Preconditions	: The inventory level of a product in a particular warehouse has fallen below its minimum level
Success Post Conditions:	The inventory level of the product in a particular warehouse is at the maximum level.
Failed Post Conditions	: The inventory level of the product in a particular warehouse is not updated and remains under stocked.
Actors	: Retailer System, Manufacturing System

Triggers	: Triggered internally in the Retailer System for each warehouse that detects the precondition.

4.2.4. UC4: Supply Finished Goods

Goal of Use Case	: A manufacturer processes a purchase order from a warehouse.
Preconditions	: Minimum < Manufacturer's Finished Goods Inventory Level < Maximum.
Success Post Conditions:	The purchase order is fulfilled and finished goods shipped to Retailer's warehouse.
Failed Post Conditions	: The purchase order is not fulfilled.
Actors	: Manufacturing System, Retailer System
Triggers	: Receipt of a purchase order.

4.2.5. UC5: Manufacture Finished Goods

Goal of Use Case	: The goal of this use case is to initiate a production run for the purposes of replenishing the stock levels of a specified product.
Preconditions	: Stock levels for the manufactured product are not sufficient to meet a purchase request or stock levels have fallen below the minimum level for the product. The necessary parts and their quantities for a production run are available.
Success Post Conditions:	The stock level for the manufactured product will be at the maximum level.
Failed Post Conditions	: Stock levels will be left unchanged.
Actors	: Manufacturing System
Triggers	: Manufacturer is requested to supply finished goods

4.3. Supply Chain Management Architecture

The sample application described herein was modeled after the "simple" supply chain management (SCM) application outlined in the SCM Use Cases; it is not intended to exhibit all of the characteristics of a real world SCM design and implementation. Rather, it serves to document and demonstrate web services might be designed, implemented and deployed.

4.3.1. Retailer System Architecture

This section pertains to the technical design and implementation of the Sample Application. In particular it defines the Retailer part of the system defined in the SCM use case. The domain model and architecture described in this section relates to the retailer system and its related use cases as defined in the *SCM Use Cases*.

4.3.1.1. Retailer System Overview

The Retailer system web service provides a façade onto the Retailer System, providing operations to access the catalog of products and to place orders. Within the Retailer System there are three instances of the warehouse web services, one for each of warehouse A, B and C defined in the Use Case document. These warehouses will in turn call out to the three Manufacturing systems, whose architecture is defined. To facilitate this interaction the warehouse has to provide a callback interface – these and the retailer web service are the only external entry points into the Retailer system.

4.3.1.2. Deployment Diagram

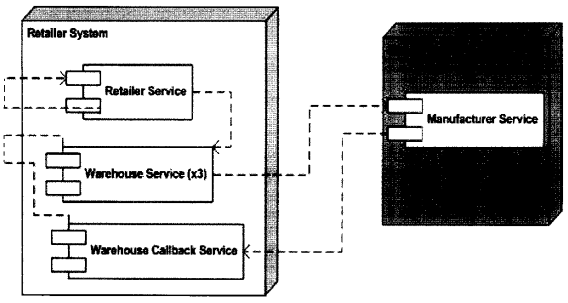


Figure 4.2 Retailer System Architecture

4.3.1.3. Retailer System Web services

The operations, message types and other relevant information for each Web service are outlined in the sections below. These sections specify key details to be defined in the associated WSDL files. The exact structure of these files is to be resolved as part of the implementation of this sample application.

The Retailer System will contain the following web services:

- Retailer Service
- Warehouse Service
- Warehouse Callback Service.

In addition, the Retailer System depends on two other services external to itself and defined elsewhere:

Logging Service - defined in the Demo System section.
Manufacturer Service – defined in the Manufacturing System section.

4.3.1.3.1. Retailer Service

Operations/Message Types

The following operations/message types should be supported, and shall follow the synchronous request/reply scenario:

Operation	Msg. Type	Message	Parameters	Type
submitOrder	Input	submitOrderRequest	PartsOrder	PartsOrderType
			Customer Details	CustomerDetailsType
			Configuration Header	Configuration
	Output	submitOrderResponse	Response	PartsOrderResponseType
	Fault	BadOrder		Client
	Fault	InvalidProductCode		Client
	Fault	ConfigurationFault		Client

Operation	Msg. Type	Message	Parameters	Type
getCatalog	Input	getCatalogRequest		
	Output	getCatalogResponse	return	CatalogType

I. SubmitOrder

Description

Refer to UC (Use Case)1 and UC2. The consumer submits an order and receives a response indicating which goods will be shipped. To determine which goods can be shipped Warehouse A,B, and C are asked in turn (see warehouse service).A BadOrder fault is returned if the order is malformed and cannot be interpreted, or contains no line items. An order is rejected, with an InvalidProductCode fault, if it contains a line item with an invalid (i.e. unknown) product code. A ConfigurationFault fault is returned if there is a problem in the configuration header.

Scenario

The scenario used will be Synchronous Request/Response using SOAP over HTTP.

Message Style

The rpc/literal message style will be used.

II. GetCatalog

Description

Refer to UC1. A call to this operation results in the Retailer sending over the list of products that it sells, this list would need to be rendered in order to present to a user. There are no input parameters as there is no choice of catalogs or parts of a catalog.

Scenario

The scenario used will be Synchronous Request/Response using SOAP over HTTP.

Message Style

The rpc/literal message style will be used.

4.3.1.3.2. Warehouse Service
Operations/Message Types

The following operations/message types should be supported, and all shall follow the synchronous request/reply scenario:

Operation	Msg. Type	Message	Parameters	Type
ShipGoods	Input	ShipGoodsRequest	ItemList	ItemList
			Customer	CustomerReferenceType
			Configuration Header	Configuration
	Output	ShipGoodsResponse	Response	ItemShippingStatusList
	Fault	ConfigurationFault		Server

I. ShipGoods

Description

Refer to UC 1 and UC2. A call to this service results in the warehouse checking each line item in the order. If for each line item the required quantity is in stock, the warehouse will ship that line item (and hence reduce the stock level). It will record which ones it has shipped and which ones it does not have enough stock for and hence cannot ship; the response will contain this list. A ConfigurationFault fault is returned if there is a problem in the configuration header. The reduction of the stock level below the minimum level will trigger the warehouse to call on the Supply service of the relevant manufacturer.

4.3.1.3.3. Warehouse Callback Service
Operations/Message Types

The following operations/message types should be supported:

Operation	Msg. Type	Message	Parameters	Type
SubmitSN	Input	SNSubmit	shippingNotice	doc
			ConfigurationHeader	Configuration
			CallbackHeader	CallbackHeader
	Output	ackSN	Response	boolean
	Fault	ConfigurationFault		Client
	Fault	CallbackFault		Server

Operation	Msg. Type	Message	Parameters	Type
ErrorPO	Input	ProcessPOFault	ProcessPOFault	SubmitPOFault
	Output	AckPO	Repsonse	boolean
	Fault	ConfigurationFault		Client
	Fault	CallbackFault		Server

I. SubmitSN

Description

Refer to UC3 and UC4. A call to this service indicates that the manufacturer has finished
Processing an order. If the manufacturer's processing has been successful, the manufacturer will submit the shipping notice using the SubmitSN operation. If the shipping notice can be correlated to an order placed with a manufacturer, a positive acknowledgement is sent in the reply; otherwise a callbackfault is returned. In this version of the sample application, all responses will cause the Manufacturer to consider
the order request complete i.e. no further processing of the order will take place.

Scenario

The scenario used will be the reply/callback portion of the Basic Callback Scenario using SOAP over HTTP.

Message Style

The doc/literal message style will be used.

II. ErrorPO

Description

Refer to Use cases 3 and 4. A call to this service indicates that the manufacturer has finished processing an order but there had been an error in doing so. If the Manufacturer's processing of the order has not been successful, the Manufacturer will
provide a reason using the ErrorPO operation. For any PO, a Manufacturer may only invoke one of submitSN or ErrorPO.

Scenario

The scenario used will be the reply/callback portion of the Basic Callback Scenario using SOAP over HTTP.

Message Style

The doc/literal message style will be used.

RETAILER.WSDL

The Retailer Service is defined by two schemas and one WSDL document. The schemas are imported into the *types* section of the WSDL.

```
  <?xml version="1.0" encoding="utf-8" ?>
- <wsdl:definitions xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
    xmlns:cfg="http://www.ws-
    i.org/SampleApplications/SupplyChainManagement/Configuration.xsd"
    xmlns:cfgw="http://www.ws-
    i.org/SampleApplications/SupplyChainManagement/Configuration.wsdl"
    xmlns:cat="http://www.ws-
    i.org/SampleApplications/SupplyChainManagement/RetailCatalog.xsd"
    xmlns:order="http://www.ws-
    i.org/SampleApplications/SupplyChainManagement/RetailOrder.xsd"
    xmlns:tns="http://www.ws-
    i.org/SampleApplications/SupplyChainManagement/Retailer.wsdl"
    targetNamespace="http://www.ws-
    i.org/SampleApplications/SupplyChainManagement/Retailer.wsdl"
    xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
    xmlns="http://schemas.xmlsoap.org/wsdl/" xmlns:wsi="http://ws-
    i.org/schemas/conformanceClaim/">
 <wsdl:documentation>
 This WSDL document describes the Retailer service for the WS-I Basic Sample
    Application. This service is part of a supply chain management system. It is
    used to demonstrate a web service that is conformant with the Basic Profile
    and to show how different web service platforms can interoperate
 .</wsdl:documentation>
 <wsdl:import namespace="http://www.ws-
    i.org/SampleApplications/SupplyChainManagement/Configuration.wsdl"
    location="Configuration.wsdl" />
- <wsdl:types>
- <xs:schema>
 <xs:import namespace="http://www.ws-
    i.org/SampleApplications/SupplyChainManagement/RetailCatalog.xsd"
    schemaLocation="RetailCatalog.xsd" />
 <xs:import namespace="http://www.ws-
    i.org/SampleApplications/SupplyChainManagement/RetailOrder.xsd"
    schemaLocation="RetailOrder.xsd" />
 <xs:import namespace="http://www.ws-
    i.org/SampleApplications/SupplyChainManagement/Configuration.xsd"
    schemaLocation="Configuration.xsd" />
   </xs:schema>
   </wsdl:types>
 <wsdl:message name="getCatalogRequest" />
- <wsdl:message name="getCatalogResponse">
- <wsdl:part name="return" type="cat:CatalogType">
 <wsdl:documentation>the product catalog</wsdl:documentation>
   </wsdl:part>
   </wsdl:message>
- <wsdl:message name="submitOrderRequest">
- <wsdl:part name="PartsOrder" type="order:PartsOrderType">
```

```
 <wsdl:documentation>XML structure holding product/quantity
    pairs</wsdl:documentation>
   </wsdl:part>
 <wsdl:part name="CustomerDetails" type="order:CustomerDetailsType" />
 <wsdl:part name="ConfigurationHeader" element="cfg:Configuration" />
   </wsdl:message>
- <wsdl:message name="submitOrderResponse">
- <wsdl:part name="return" type="order:PartsOrderResponseType">
 <wsdl:documentation>XML structure holding product/quantity ordered
    pairs, with optional failure message</wsdl:documentation>
   </wsdl:part>
   </wsdl:message>
- <wsdl:message name="BadOrderFault">
 <wsdl:part name="Reason" element="order:BadOrderReason" />
   </wsdl:message>
- <wsdl:message name="InvalidProductCodeFault">
 <wsdl:part name="InvalidProductCode" element="order:InvalidProductCode" />
   </wsdl:message>
- <wsdl:portType name="RetailerPortType">
- <wsdl:operation name="getCatalog">
 <wsdl:documentation>returns a product catalog</wsdl:documentation>
 <wsdl:input message="tns:getCatalogRequest" name="getCatalogRequest" />
 <wsdl:output message="tns:getCatalogResponse"
    name="getCatalogResponse" />
   </wsdl:operation>
- <wsdl:operation name="submitOrder">
 <wsdl:documentation>Accept an order for quantities of multiple
    products</wsdl:documentation>
 <wsdl:input message="tns:submitOrderRequest" name="submitOrderRequest"
    />
 <wsdl:output message="tns:submitOrderResponse"
    name="submitOrderResponse" />
 <wsdl:fault name="BadOrder" message="tns:BadOrderFault" />
```

```
 <wsdl:fault name="InvalidProductCode"
    message="tns:InvalidProductCodeFault" />
 <wsdl:fault name="ConfigurationFault"
    message="cfgw:ConfigurationFaultMessage" />
   </wsdl:operation>
   </wsdl:portType>
- <wsdl:binding name="RetailerSoapBinding" type="tns:RetailerPortType">
- <wsdl:documentation>
 <wsi:Claim conformsTo="http://ws-i.org/profiles/basic1.0/" />
   </wsdl:documentation>
 <soap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http" />
- <wsdl:operation name="getCatalog">
 <soap:operation soapAction="" style="rpc" />
- <wsdl:input>
 <soap:body use="literal" namespace="http://www.ws-
    i.org/SampleApplications/SupplyChainManagement/Retailer.wsdl" />
   </wsdl:input>
- <wsdl:output>
 <soap:body use="literal" namespace="http://www.ws-
    i.org/SampleApplications/SupplyChainManagement/Retailer.wsdl" />
   </wsdl:output>
   </wsdl:operation>
- <wsdl:operation name="submitOrder">
 <soap:operation soapAction="" style="rpc" />
- <wsdl:input>
 <soap:body use="literal" namespace="http://www.ws-
    i.org/SampleApplications/SupplyChainManagement/Retailer.wsdl"
    parts="PartsOrder CustomerDetails" />
- <soap:header message="tns:submitOrderRequest" part="ConfigurationHeader"
    use="literal" wsdl:required="true">
 <soap:headerfault message="cfgw:ConfigurationFaultMessage"
    part="ConfigurationFault" use="literal" />
   </soap:header>
   </wsdl:input>
```

```
- <wsdl:output>
 <soap:body use="literal" namespace="http://www.ws-
    i.org/SampleApplications/SupplyChainManagement/Retailer.wsdl" />
   </wsdl:output>
- <wsdl:fault name="BadOrder">
 <soap:fault name="BadOrder" use="literal" />
   </wsdl:fault>
- <wsdl:fault name="InvalidProductCode">
 <soap:fault name="InvalidProductCode" use="literal" />
   </wsdl:fault>
   </wsdl:operation>
   </wsdl:binding>
   </wsdl:definitions>
     - <!--
      The following is an example of a getCatalog SOAP request message
     compliant with the above WSDL:

     <SOAP-ENV:Envelope xmlns:SOAP-
     ENV="http://schemas.xmlsoap.org/soap/envelope/">
     <SOAP-ENV:Body>
     <ns1:getCatalog xmlns:ns1="http://www.ws-
     i.org/SampleApplications/SupplyChainManagement/Retailer.wsdl">
     </ns1:getCatalog>
     </SOAP-ENV:Body>
     </SOAP-ENV:Envelope>

     The following is an example of a getCatalog SOAP response
     message compliant with the above WSDL:

     <SOAP-ENV:Envelope xmlns:SOAP-
     ENV="http://schemas.xmlsoap.org/soap/envelope/">
     <SOAP-ENV:Body>
     <ns1:getCatalogResponse xmlns:ns1="http://www.ws-
     i.org/SampleApplications/SupplyChainManagement/Retailer.wsdl">
```

```
<return>
  <Item xmlns="http://www.ws-
i.org/SampleApplications/SupplyChainManagement/RetailCatalog.xsd">
    <name>TV,Brand1</name>
    <description>24", Color, Advanced Velocit Scan Modular</description>
    <productNumber>605001</productNumber>
    <category>TV</category>
    <brand>Brand1</brand>
    <price>299.95</price>
  </Item>
  <Item xmlns="http://www.ws-
i.org/SampleApplications/SupplyChainManagement/RetailCatalog.xsd">
    <name>TV, Brand2</name>
    <description>32", Super Slim Flat Panel Plasma</description>
    <productNumber>605002</productNumber>
    <category>TV</category>
    <brand>Brand2</brand>
    <price>1499.99</price>
  </Item>
</return>
</ns1:getCatalogResponse>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

The following is an example of a submitOrder SOAP request message compliant with the above WSDL:

```
<SOAP-ENV:Envelope xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelope/">
<SOAP-ENV:Header>
<h:Configuration
        xmlns:h="http://www.ws-
i.org/SampleApplications/SupplyChainManagement/Configuration.xsd">
  <h:UserId>griddell@bowstreet.com</h:UserId>
```

```
  <h:ServiceUrl
Role="LoggingFacility">http://example1/SampleApp/SCM/LoggingFacility<
/h:ServiceUrl>
  <h:ServiceUrl
Role="Retailer">http://example2/wsi/soaprpc/wsi/RetailerImpl</h:ServiceU
rl>
  <h:ServiceUrl
Role="WarehouseA">http://example3/wsi/soaprpc/wsi/WarehouseAImpl</
h:ServiceUrl>
  <h:ServiceUrl
Role="WarehouseB">http://example4/wsi/soaprpc/wsi/WarehouseBImpl</
h:ServiceUrl>
  <h:ServiceUrl
Role="WarehouseC">http://example5/wsi/soaprpc/wsi/WarehouseCImpl</
h:ServiceUrl>
  <h:ServiceUrl Role="ManufacturerA">http://example6/ws-
i_sample/ManufacturerA</h:ServiceUrl>
  <h:ServiceUrl Role="ManufacturerB">http://example7/ws-
i_sample/ManufacturerB</h:ServiceUrl>
  <h:ServiceUrl Role="ManufacturerC">http://example8/ws-
i_sample/ManufacturerC</h:ServiceUrl>
</h:Configuration>
</SOAP-ENV:Header>
<SOAP-ENV:Body>
<ns1:submitOrder xmlns:ns1="http://www.ws-
i.org/SampleApplications/SupplyChainManagement/Retailer.wsdl">
<PartsOrder xmlns:p="http://www.ws-
i.org/SampleApplications/SupplyChainManagement/RetailOrder.xsd">
  <p:Item>
    <p:productNumber>605006</p:productNumber>
    <p:quantity>182</p:quantity>
    <p:price>3.99</p:price>
  </p:Item>
  <p:Item><p:productNumber>605002</p:productNumber>
```

```
    <p:quantity>4</p:quantity>
    <p:price>3.99</p:price>
    </p:Item>
  <p:Item>
    <p:productNumber>605003</p:productNumber>
    <p:quantity>82</p:quantity>
    <p:price>7.99</p:price>
  </p:Item>
</PartsOrder>
<CustomerDetails xmlns:c="http://www.ws-
i.org/SampleApplications/SupplyChainManagement/RetailOrder.xsd">
  <c:custnbr>ABCD999999999EFG</c:custnbr>
  <c:name>Joe Bloggs</c:name>
  <c:street1 />
  <c:city />
  <c:state>NH</c:state>
  <c:zip>03870</c:zip>
  <c:country>USA</c:country>
</CustomerDetails>
</ns1:submitOrder>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

The following is an example of a submitOrder SOAP response message that is compliant with the WSDL:

```
<SOAP-ENV:Envelope xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelope/">
<SOAP-ENV:Body>
<ns1:submitOrderResponse xmlns:ns1="http://www.ws-
i.org/SampleApplications/SupplyChainManagement/Retailer.wsdl">
<return xmlns:ns2="http://www.ws-
i.org/SampleApplications/SupplyChainManagement/RetailOrder.xsd">
  <ns2:Item>
```

```
    <ns2:productNumber>605002</ns2:productNumber>
    <ns2:quantity>4</ns2:quantity>
    <ns2:price>3.99</ns2:price>
    <ns2:comment>in stock from WarehouseA</ns2:comment>
  </ns2:Item>
  <ns2:Item>
    <ns2:productNumber>605006</ns2:productNumber>
    <ns2:quantity>0</ns2:quantity>
    <ns2:price>0</ns2:price>
    <ns2:comment>insufficient stock</ns2:comment>
  </ns2:Item>
  <ns2:Item>
    <ns2:productNumber>605003</ns2:productNumber>
    <ns2:quantity>0</ns2:quantity>
    <ns2:price>0</ns2:price>
    <ns2:comment>insufficient stock</ns2:comment>
  </ns2:Item>
</return>
</ns1:submitOrderResponse>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

4.3.2 Manufacturing System Architecture

This section pertains to the technical design and implementation of the Manufacturer System of the Basic Profile Sample Application as specified in the *SCM Use case*. The domain model described in this section relates to the following use cases outlined in the *SCM Use case,* and as such should be read in conjunction with that document:

- UC4: Supply Finished Goods
- UC5: Manufacture Finished Goods

The various types of diagrams (class, sequence, deployment, etc.) included in this section each provide a different view of the application, including its interfaces, roles and responsibilities, business rules, processes as well as the information collected and maintained. In general this model is implementation independent except where technical constraints are given as part of the requirements.

4.3.2.1. Manufacturing System Overview

The Manufacturing system supplies finished goods to warehouses. Requests for finished goods may be fulfilled by the Manufacturer by supplying from internal stock or, if the required quantity is not available, by scheduling a production run. Since there could be a considerable time delay between receiving the order and informing the warehouse of shipment of goods, an asynchronous processing model is used. This allows a warehouse to proceed on other business, and allows the Manufacturer to callback to the Warehouse once the order has been fulfilled.

4.3.2.2. Deployment Diagram

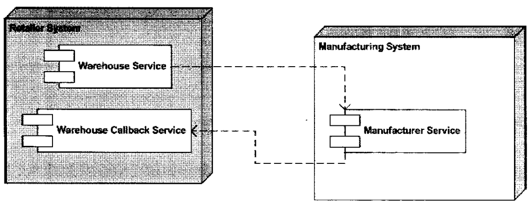


Figure 4.3 Manufacturing System Architecture

4.3.2.3. Manufacturing System Web services

The operations, message types and other relevant information for each Web service are outlined in the sections below. These sections specify key details to be defined in the associated WSDL files. The exact structure of these files is to be resolved as part of the implementation of the sample application. The Manufacturer System portion of the sample application will contain the following web services:

• Manufacturer Service

4.3.2.3.1. Manufacturer Service Operations/Message Types

The following operations/message types should be supported:

Operation	Msg. Type	Message	Parameters	Type
submitPO	Input	POSubmit	PurchaseOrder	doc
			ConfigurationHeader	Configuration
			StartHeader	StartHeader
	Output	ackPO	Response	boolean
	Fault	POFault		Client
	Fault	ConfigurationFault		Client

I. SubmitPO

Description

The purpose of this operation is to place a purchase order with the manufacturer for finished goods. PurchaseOrder is a set of line items describing ordered finished goods.

Manufacturer.WSDL

The Manufacturer is defined by a schema and [part of] a WSDL document. The schema is imported into the *types* section of the WSDL. As mentioned above, the Manufacturer WSDL document contains two port types and bindings which together describe a single Web service type. The ManufacturerPortType and associated messages and bindings apply to the Manufacturer; the other applies to the Warehouse.

Following XML document WSDL description of manufacturing system.

Manufacturer.wsdl

```
<?xml version="1.0" encoding="UTF-8" ?>
<wsdl:definitions xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" xmlns:cb="http://www.ws-
  i.org/SampleApplications/SupplyChainManagement/Manufacturer/CallBack"
```

```
  xmlns:ct="http://www.ws-
  i.org/SampleApplications/SupplyChainManagement/Configuration.xsd"
  xmlns:cfg="http://www.ws-
  i.org/SampleApplications/SupplyChainManagement/Configuration.wsdl"
  xmlns:po="http://www.ws-
  i.org/SampleApplications/SupplyChainManagement/ManufacturerPO.xsd"
  xmlns:sn="http://www.ws-
  i.org/SampleApplications/SupplyChainManagement/ManufacturerSN.xsd"
  xmlns:tns="http://www.ws-
  i.org/SampleApplications/SupplyChainManagement/Manufacturer.wsdl"
  targetNamespace="http://www.ws-
  i.org/SampleApplications/SupplyChainManagement/Manufacturer.wsdl"
  xmlns:wsi="http://ws-i.org/schemas/conformanceClaim/">
<wsdl:documentation>This WSDL document describes the Manufacturer
  service for the WS-I Basic Sample Application. This service is part of a
  supply chain management system. It is used to demonstrate a web
  service that is conformant with the Basic Profile and to show how
  different web service platforms can interoperate.</wsdl:documentation>
<wsdl:import namespace="http://www.ws-
  i.org/SampleApplications/SupplyChainManagement/Configuration.wsdl"
  location="../Configuration.wsdl" />
- <wsdl:types>
- <xs:schema elementFormDefault="qualified"
  attributeFormDefault="unqualified">
<xs:import namespace="http://www.ws-
  i.org/SampleApplications/SupplyChainManagement/Configuration.xsd"
  schemaLocation="../Configuration.xsd" />
<xs:import namespace="http://www.ws-
  i.org/SampleApplications/SupplyChainManagement/ManufacturerPO.xsd"
  schemaLocation="ManufacturerPO.xsd" />
<xs:import namespace="http://www.ws-
  i.org/SampleApplications/SupplyChainManagement/ManufacturerSN.xsd"
  schemaLocation="ManufacturerSN.xsd" />
 </xs:schema>
```

```
- <xs:schema targetNamespace="http://www.ws-
  i.org/SampleApplications/SupplyChainManagement/Manufacturer/CallBack"
  elementFormDefault="qualified" attributeFormDefault="unqualified">
 <xs:element name="StartHeader" type="cb:StartHeaderType" />
 <xs:element name="CallbackHeader" type="cb:CallbackHeaderType" />
 <xs:element name="CallbackFault" type="cb:CallbackFaultType" />
- <xs:complexType name="StartHeaderType">
- <xs:sequence>
 <xs:element minOccurs="1" maxOccurs="1" name="conversationID"
  type="xs:string" />
 <xs:element minOccurs="1" maxOccurs="1" name="callbackLocation"
  type="xs:string" />
  </xs:sequence>
  </xs:complexType>
- <xs:complexType name="CallbackHeaderType">
- <xs:sequence>
 <xs:element minOccurs="1" maxOccurs="1" name="conversationID"
  type="xs:string" />
  </xs:sequence>
  </xs:complexType>
- <xs:complexType name="CallbackFaultType">
- <xs:sequence>
- <xs:element name="Reason">
- <xs:simpleType>
- <xs:restriction base="xs:NMTOKEN">
 <xs:enumeration value="notFound" />
  </xs:restriction>
  </xs:simpleType>
  </xs:element>
 <xs:element minOccurs="1" maxOccurs="1" name="conversationID"
  type="xs:string" />
  </xs:sequence>
  </xs:complexType>
  </xs:schema>
```

```
    </wsdl:types>
- <wsdl:message name="POSubmit">
  <wsdl:documentation>A purchase order.</wsdl:documentation>
  <wsdl:part name="PurchaseOrder" element="po:PurchaseOrder" />
  <wsdl:part name="ConfigurationHeader" element="ct:Configuration" />
  <wsdl:part name="StartHeader" element="cb:StartHeader" />
    </wsdl:message>
- <wsdl:message name="ackPO">
  <wsdl:documentation>A response of true indicates the purchase order has
      been accepted for processing.</wsdl:documentation>
  <wsdl:part name="Response" element="po:ackPO" />
    </wsdl:message>
- <wsdl:message name="submitPOFault">
  <wsdl:part name="POFault" element="po:submitPOFault" />
    </wsdl:message>
- <wsdl:message name="SNSubmit">
  <wsdl:documentation>A shipment notification.</wsdl:documentation>
  <wsdl:part name="ShipmentNotice" element="sn:ShipmentNotice" />
  <wsdl:part name="ConfigurationHeader" element="ct:Configuration" />
  <wsdl:part name="CallbackHeader" element="cb:CallbackHeader" />
    </wsdl:message>
- <wsdl:message name="processPOFault">
  <wsdl:documentation>Alternative to SNSubmit, indicates a reason for the
      rejection of a given PO after having been acknowledged by the
      Manufacturer. Contains callback information with which to find the
      original replenishment request.</wsdl:documentation>
  <wsdl:part name="processPOFault" element="po:submitPOFault" />
  <wsdl:part name="ConfigurationHeader" element="ct:Configuration" />
  <wsdl:part name="CallbackHeader" element="cb:CallbackHeader" />
    </wsdl:message>
- <wsdl:message name="ackSN">
  <wsdl:documentation>A response of true indicates the shipment notice has
      been accepted for processing.</wsdl:documentation>
  <wsdl:part name="Response" element="sn:ackSN" />
```

```
    </wsdl:message>
- <wsdl:message name="Callback">
  <wsdl:documentation>To be used in SOAP headers for relating two req/resp
      message pairs sent asynchronously. The CallbackFault indicates a
      reason for the rejection of the second req/resp
      pair.</wsdl:documentation>
  <wsdl:part name="CallbackFault" element="cb:CallbackFault" />
    </wsdl:message>
- <wsdl:portType name="ManufacturerPortType">
- <wsdl:operation name="submitPO">
  <wsdl:documentation>Submit a purchase order for specified items to the
      manufacturer.</wsdl:documentation>
  <wsdl:input message="tns:POSubmit" />
  <wsdl:output message="tns:ackPO" />
  <wsdl:fault name="POFault" message="tns:submitPOFault" />
  <wsdl:fault name="ConfigurationFault"
      message="cfg:ConfigurationFaultMessage" />
    </wsdl:operation>
    </wsdl:portType>
- <wsdl:portType name="WarehouseCallbackPortType">
- <wsdl:operation name="submitSN">
  <wsdl:documentation>Submit a shipment notice for specified items to the
      retailer.</wsdl:documentation>
  <wsdl:input message="tns:SNSubmit" />
  <wsdl:output message="tns:ackSN" />
  <wsdl:fault name="ConfigurationFault"
      message="cfg:ConfigurationFaultMessage" />
  <wsdl:fault name="CallbackFault" message="tns:Callback" />
    </wsdl:operation>
- <wsdl:operation name="errorPO">
  <wsdl:documentation>Notify warehouse there was an error in processing a
      submitted PO.</wsdl:documentation>
  <wsdl:input message="tns:processPOFault" />
  <wsdl:output message="tns:ackPO" />
```

```
  <wsdl:fault name="ConfigurationFault"
      message="cfg:ConfigurationFaultMessage" />
  <wsdl:fault name="CallbackFault" message="tns:Callback" />
    </wsdl:operation>
    </wsdl:portType>
- <wsdl:binding name="ManufacturerSoapBinding"
      type="tns:ManufacturerPortType">
- <wsdl:documentation>
  <wsi:Claim conformsTo="http://ws-i.org/profiles/basic1.0/" />
    </wsdl:documentation>
  <soap:binding style="document"
      transport="http://schemas.xmlsoap.org/soap/http" />
- <wsdl:operation name="submitPO">
  <soap:operation />
- <wsdl:input>
  <soap:body parts="PurchaseOrder" use="literal" />
- <soap:header message="tns:POSubmit" part="ConfigurationHeader"
      use="literal">
  <soap:headerfault message="cfg:ConfigurationFaultMessage"
      part="ConfigurationFault" use="literal" />
    </soap:header>
  <soap:header message="tns:POSubmit" part="StartHeader" use="literal" />
    </wsdl:input>
- <wsdl:output>
  <soap:body use="literal" />
    </wsdl:output>
- <wsdl:fault name="POFault">
  <soap:fault name="POFault" use="literal" />
    </wsdl:fault>
    </wsdl:operation>
    </wsdl:binding>
- <wsdl:binding name="WarehouseCallbackSoapBinding"
      type="tns:WarehouseCallbackPortType">
```

```
  <soap:binding style="document"
      transport="http://schemas.xmlsoap.org/soap/http" />
- <wsdl:operation name="submitSN">
  <soap:operation />
- <wsdl:input>
  <soap:body parts="ShipmentNotice" use="literal" />
- <soap:header message="tns:SNSubmit" part="ConfigurationHeader"
      use="literal">
  <soap:headerfault message="cfg:ConfigurationFaultMessage"
      part="ConfigurationFault" use="literal" />
    </soap:header>
- <soap:header message="tns:SNSubmit" part="CallbackHeader" use="literal">
  <soap:headerfault message="tns:Callback" part="CallbackFault" use="literal" />
    </soap:header>
    </wsdl:input>
- <wsdl:output>
  <soap:body parts="Response" use="literal" />
    </wsdl:output>
    </wsdl:operation>
- <wsdl:operation name="errorPO">
  <soap:operation />
- <wsdl:input>
  <soap:body parts="processPOFault" use="literal" />
- <soap:header message="tns:processPOFault" part="CallbackHeader"
      use="literal">
  <soap:headerfault message="tns:Callback" part="CallbackFault" use="literal" />
    </soap:header>
    </wsdl:input>
- <wsdl:output>
  <soap:body parts="Response" use="literal" />
    </wsdl:output>
    </wsdl:operation>
    </wsdl:binding>
    </wsdl:definitions>
```

- <!--

The following is an example of a SOAP request message compliant with the above WSDL:

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
        xmlns:xsd="http://www.w3.org/XMLSchema"
        xmlns:xsi="http://www.w3.org/XMLSchema-instance">
  <soapenv:Header>
    <ns1:Configuration xmlns:ns1="http://www.ws-
i.org/SampleApplications/SupplyChainManagement/Configuration.xsd">
      <ns1:UserId>Some user ID</ns1:UserId>
      <ns1:ServiceUrl Role="LoggingFacility">http://localhost:999/WS-
I_Sample/services/LoggingFacility</ns1:ServiceUrl>
      <ns1:ServiceUrl Role="Retailer">http://localhost:999/WS-
I_Sample/services/Retailer</ns1:ServiceUrl>
      <ns1:ServiceUrl Role="WarehouseA">http://localhost:999/WS-
I_Sample/services/WarehouseA</ns1:ServiceUrl>
      <ns1:ServiceUrl Role="WarehouseB">http://localhost:999/WS-
I_Sample/services/WarehouseB</ns1:ServiceUrl>
      <ns1:ServiceUrl Role="WarehouseC">http://localhost:999/WS-
I_Sample/services/WarehouseC</ns1:ServiceUrl>
      <ns1:ServiceUrl Role="ManufacturerA">http://localhost:999/WS-
I_Sample/services/ManufacturerA</ns1:ServiceUrl>
      <ns1:ServiceUrl Role="ManufacturerB">http://localhost:999/WS-
I_Sample/services/ManufacturerB</ns1:ServiceUrl>
      <ns1:ServiceUrl Role="ManufacturerC">http://localhost:999/WS-
I_Sample/services/ManufacturerC</ns1:ServiceUrl>
    </ns1:Configuration>
    <ns2:StartHeader xmlns:ns2="http://www.ws-
i.org/SampleApplications/SupplyChainManagement/Manufacturer/CallBac
k">
      <ns2:conversationID>1</ns2:conversationID>
```

```
      <ns2:callbackLocation>http://localhost:999/WS-
I_Sample/services/WarehouseCallBack</ns2:callbackLocation>
    </ns2:StartHeader>
  </soapenv:Header>
  <soapenv:Body>
    <PurchaseOrder xmlns="http://www.ws-
i.org/SampleApplications/SupplyChainManagement/ManufacturerPO.xsd"
>
      <orderNum>1</orderNum>
      <customerRef>ABCD999999999EFG</customerRef>
      <items>
        <Item>
          <ID>605002</ID>
          <qty>18</qty>
          <price>100.0</price>
        </Item>
      </items>
      <total>0.0</total>
    </PurchaseOrder>
  </soapenv:Body>
</soapenv:Envelope>
```

and the reply from the Manufacturer:

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
        xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Body>
    <ackPO xmlns="http://www.ws-
i.org/SampleApplications/SupplyChainManagement/ManufacturerPO.xsd"
>true</ackPO>
  </soapenv:Body>
```

```
</soapenv:Envelope>
```

and the callback request:

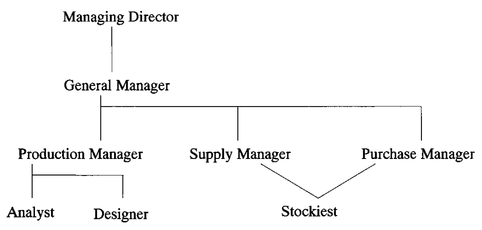
```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
        xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Header>
    <ns1:CallbackHeader xmlns:ns1="http://www.ws-
i.org/SampleApplications/SupplyChainManagement/Manufacturer/CallBac
k">
      <ns1:conversationID>1</ns1:conversationID>
    </ns1:CallbackHeader>
    <ns2:Configuration xmlns:ns2="http://www.ws-
i.org/SampleApplications/SupplyChainManagement/Configuration.xsd">
      <ns2:UserId>Some user ID</ns2:UserId>
      <ns2:ServiceUrl Role="LoggingFacility">http://localhost:999/WS-
I_Sample/services/LoggingFacility</ns2:ServiceUrl>
      <ns2:ServiceUrl Role="Retailer">http://localhost:999/WS-
I_Sample/services/Retailer</ns2:ServiceUrl>
      <ns2:ServiceUrl Role="WarehouseA">http://localhost:999/WS-
I_Sample/services/WarehouseA</ns2:ServiceUrl>
      <ns2:ServiceUrl Role="WarehouseB">http://localhost:999/WS-
I_Sample/services/WarehouseB</ns2:ServiceUrl>
      <ns2:ServiceUrl Role="WarehouseC">http://localhost:999/WS-
I_Sample/services/WarehouseC</ns2:ServiceUrl>
      <ns2:ServiceUrl Role="ManufacturerA">http://localhost:999/WS-
I_Sample/services/ManufacturerA</ns2:ServiceUrl>
      <ns2:ServiceUrl Role="ManufacturerB">http://localhost:999/WS-
I_Sample/services/ManufacturerB</ns2:ServiceUrl>
      <ns2:ServiceUrl Role="ManufacturerC">http://localhost:999/WS-
I_Sample/services/ManufacturerC</ns2:ServiceUrl>
```

```
    </ns2:Configuration>
  </soapenv:Header>
  <soapenv:Body>
    <ShipmentNotice xmlns="http://www.ws-
i.org/SampleApplications/SupplyChainManagement/ManufacturerSN.xsd"
>
      <shipNum>1</shipNum>
      <orderNum>1</orderNum>
      <customerRef>ABCD999999999EFG</customerRef>
      <items>
        <Item>
          <ID>605002</ID>
          <qty>18</qty>
          <price>100.0</price>
        </Item>
      </items>
      <total>0.0</total>
    </ShipmentNotice>
  </soapenv:Body>
</soapenv:Envelope>
```

and the Callback reply:

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
        xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Body>
    <ackSN xmlns="http://www.ws-
i.org/SampleApplications/SupplyChainManagement/ManufacturerSN.xsd"
>true</ackSN>
  </soapenv:Body>
</soapenv:Envelope>
```

In this project I used consolidated RBAC model. Allocation of roles in this sample application has been given below.



4.4.2. Public key Infrastructure

Here I use RSA public key algorithm for Authentication. The RSA algorithm has given below. It consists of two operations 1. Key generation and 2. Encryption.

Key generation

In order to choose the public and private keys, we must do

1. Choose two large prime numbers, p and q.
2. Compute n=pq and z= (p-1) (q-1).
3. Choose a number, e, less than n, which has no common factors with z.
4. Find a number, d, such that ed-1 is exactly divisible by z.
5. Now the public key that available to the world is a pair of numbers (n, e); and the private key is the pair of (n, d).

The encryption and decryption process is given below

Cipher text c can be obtained with the following operation

Cipher text c = m^e mod n.

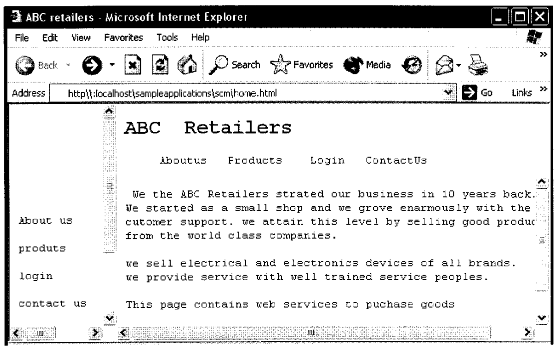
Then the original message can be retrieved by the following operation called decryption

Original message m = c^d mod n.

In this project I use RSA algorithm to encrypt simple integer identity of enterprises called retailer system and manufacturing system.

4.5. Results

The following snapshots will show the user interfaces involved in the application.


ABC retailers - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back
Search
Favorites
Media
Address
Go
Links

http\\:localhost\sampleapplications\scm\home.html

ABC Retailers

Aboutus Products Login ContactUs

About us
produts
login
contact us

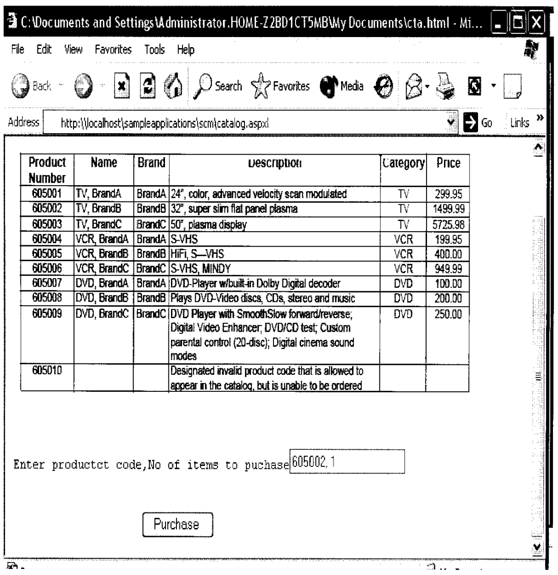
We the ABC Retailers strated our business in 10 years back. We started as a small shop and we grove enarmously with the cutomer support. we attain this level by selling good produc from the world class companies.

we sell electrical and electronics devices of all brands. we provide service with well trained service peoples.

This page contains web services to puchase goods

Figure 4.4 – Retailer Home Page

The above figure shows the home page of a retailer. Customer can buy goods by navigating this page.


C:\Documents and Settings\Administrator.HOME-Z2BD1CT5MB\My Documents\cta.html - Mi...
File Edit View Favorites Tools Help
Back
Search
Favorites
Media
Address
Go
Links

http:\\localhost\sampleapplications\scm\catalog.aspx\

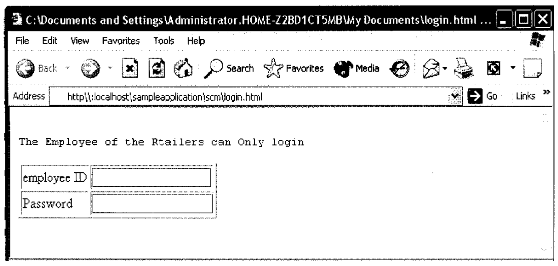
Product Number	Name	Brand	Description	Category	Price
605001	TV, BrandA	BrandA	24", color, advanced velocity scan modulated	TV	299.95
605002	TV, BrandB	BrandB	32", super slim flat panel plasma	TV	1499.99
605003	TV, BrandC	BrandC	50", plasma display	TV	5725.98
605004	VCR, BrandA	BrandA	S-VHS	VCR	199.95
605005	VCR, BrandB	BrandB	HiFi, S—VHS	VCR	400.00
605006	VCR, BrandC	BrandC	S-VHS, MINDY	VCR	949.99
605007	DVD, BrandA	BrandA	DVD-Player w/built-in Dolby Digital decoder	DVD	100.00
605008	DVD, BrandB	BrandB	Plays DVD-Video discs, CDs, stereo and music	DVD	200.00
605009	DVD, BrandC	BrandC	DVD Player with SmoothSlow forward/reverse; Digital Video Enhancer; DVD/CD test; Custom parental control (20-disc); Digital cinema sound modes	DVD	250.00
605010			Designated invalid product code that is allowed to appear in the catalog, but is unable to be ordered		

Enter productct code,No of items to puchase 605002, 1

Purchase

Figure 4.5 – Product Catalog

The above figure shows the product catalog which can be viewed by the user and can get the knowledge about the product.


C:\Documents and Settings\Administrator.HOME-Z2BD1CT5MB\My Documents\login.html ...
File Edit View Favorites Tools Help
Back
Search
Favorites
Media
Address
Go
Links

http\\:localhost\sampleapplication\scm\login.html

The Employee of the Rtailers can Only login

employee ID	
Password	

Figure 4.6 – Login Service

The above figure shows the login facility to for the employees to access the resources .the role based access control system makes use of this login information.

CHAPTER 5
CONCLUSION AND FUTURE WORK

With this proposed Extended Enterprise approach we can establish plug and play collaboration between independent enterprises with different core competencies through web services. The Extended Enterprise approach provides secure and flexible data access in enterprise collaboration via web services by combining distributed Role Based Access Control with Public Key Infrastructure. I implement a simple supply chain management application for electrical and electronics goods selling operations, which uses Role Based Access Control as an authorization mechanism and RSA encryption algorithm to provide authentication, as an example of Business – to-Business application to show the feasibility of our approach.

In our experience with extended enterprises, we have found that the main difficulty is not in provisioning the service but provisioning a reliable infrastructure for secure and flexible access control. Our efforts to develop such a foundation began with the DRBAC-EE approach described in this article.

Our long term goal is to create a software system that allows trading partners to design, manufacture, and sell products commonly-irrespective of enterprise borders.

REFERENCE

[1]."Authorization and attribute certificates for widely Distributed Access Control"-W. Johnson, S. Mudumbai, and M.Thompson. *7th international workshop enabling Technologies: Infrastructure for collaborative enterprises.* 1998

[2]. "Building trust in e-commerce"- y.Atif, *IEEE Internet Computing*, vol 6, no 1, Jan/feb 2002.

[3]."Distributed role based access control for Dynamic Coalition Environments- *Eric Frudenthal, Tracy Pesin, and Vijay Karamcheti, IEEE Internet Computing, 2002.*

[4]. "Managing Access in Extended Enterprises" – Karl Furst, Thomas Schmit, and Gerald Wippel. *IEEE Internet Computing* sep/oct 2002

[5]. "Role Based Access Control Models" – R. S. Sandhu and E.J. Coyne, *Computer*, vol 32, no 2, Feb 1999.

[6]. "Cryptography and network security" –William stalling

[7]. "Authentication and its privacy effects", IEEE Internet Computing, Jun 2002.

[8]. "Public Key Infrastructure that satisfy Security Goals", IEEE Internet Computing, Jun 2002.