IBM® WebSphere Commerce®

Access Control Guide

Version 5.4

IBM® WebSphere Commerce®

Access Control Guide

Version 5.4

Note:
Before using this information and the product it supports, be sure to read the information in the Notices section.

First Edition, Third Revision (December 2002)

Order publications through your IBM representative or the IBM branch office serving your locality. Publications are not stocked at the address given below.

IBM welcomes your comments. You can send your comments by any one of the following methods:

1. Electronically to the network ID listed below. Be sure to include your entire network address if you would like a reply.

   `Internet: torrcf@ca.ibm.com`

2. By mail to the following address:

   IBM Canada Ltd. Laboratory
   B3/KB7/8200/MKM
   8200 Warden Avenue
   Markham, Ontario, Canada L6G 1C7

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

Where to find information

WebSphere Commerce™ has online and hardcopy information describing the complete e-commerce solution. In addition, the software products that are bundled with WebSphere Commerce provide further information, describing the specific features and functions of the software. This section provides a quick overview of where to locate the various types of information.

WebSphere Commerce publications

- *IBM™® WebSphere Commerce Fundamentals, Version 5.4*
- *IBM™ WebSphere Commerce Programmer's Guide, Version 5.4*
- *IBM™ WebSphere Commerce for Windows NT™® and Windows™® 2000, Quick Beginnings, Version 5.4*
- *IBM™ WebSphere Commerce Studio Business Developer Edition for Windows NT™ and Windows™ 2000 Installation Guide, Version 5.4*
- *IBM™ WebSphere Commerce Migration Guide, Version 5.4*

For updates to these publications, refer to the following Web address:
`http://www.ibm.com/software/webservers/commerce/wc_be/lit-tech-general.html`

WebSphere Commerce online help

The WebSphere Commerce online help consists of online information that can be viewed using a Web browser. Extracts from the online information have also been compiled into related subject area PDF (Portable Document Format) documents.

The online help can be accessed from a Web browser that runs on Internet Explorer, Version 5.5, or higher using the following address:

`http://`*`host_name`*`/wchelp/`, where *`host_name`* is the name of your WebSphere Commerce machine.

In addition, on Windows, the help can be accessed from the Start menu as follows:

Start – > Programs – > IBM® WebSphere Commerce – > Documentation

Further information on the Web

Support

To find support information, including newsgroups, FAQs, technical notes, troubleshooting information and downloads, visit the following Web address:

▶ Business

`http://www.ibm.com/software/webservers/commerce/wc_be/support.html`

▶ Professional

`http://www.ibm.com/software/webservers/commerce/wc_pe/support.html`

Software partners

There are many software partners that offer products and services to enhance WebSphere Commerce. For information about these partners, visit the following Web address: `http://www.ibm.com/software/webservers/commerce/community` and click the Software Developers link.

Redbooks™™

To find more advanced technical information, visit the Redbooks Web site, which is located at `http://www.ibm.com/redbooks` and search for WebSphere Commerce.

Before you begin

The *IBM WebSphere Commerce, Version 5.4 Access Control Guide* is intended for site administrators who want to manage access to their WebSphere Commerce site. Store administrators can perform limited access management for the organizational entity for which they play their role.

This guide provides an introduction to access management, including an overview of organizations and users, access control policies, their hierarchy and relationships, and the default policies that are packaged with the product. This guide also provides a wide range of scenarios to help site administrators who want to make basic customizations to their existing policies, as well as guidelines for testing modified policies, and making performance considerations.

This book is divided into the following sections:

Chapter 1: Overview A brief overview of the key features of WebSphere Commerce's access control system, as well as a description of what has changed since the previous release of WebSphere Commerce.

Chapter 2: Getting started An introduction to access management, including how to define organizations and users, how organizations and users are related to access control policies, the basic structure of an access control policy, and how to read and identify the key parts of a policy in the WebSphere Commerce Administration Console, and in XML.

Chapter 3: Access control concepts Conceptual information on the structure of an organization and its sub-organizations, how users are granted access to a system, descriptions of default roles, and related terminology.

Chapter 4: Customizing your access control policies An in-depth look at resource-level and role-based policies, their relationship and hierarchy.

Chapter 5: Access control scenarios A variety of scenarios to show you how to make basic modifications to the default access control policies shipped with WebSphere Commerce.

Chapter 6: Using XML files to customize access control policies A look at customizing the parts of an access control policy using XML. Includes step-by-step procedures for loading policy information from the XML files into the access control database tables, and for extracting policy information from the access control database tables into the XML files.

Appendix: Table of default access control policies A complete list of all of the default access control policies loaded into your system at the time of installation.

Assumptions

This guide assumes that you have successfully installed and configured IBM WebSphere Commerce, Version 5.4 on your site, and that you have Site Administrator access to the WebSphere Commerce Administration Console tool. Store Administrators are able to manage the access control policies for their organizational entity using the WebSphere Commerce Administration Console tool, but are not able to manage the components of the policies, such as action groups and resource groups, since these are system-wide entities.

This guide also assumes that your system meets all of the software and hardware requirements for running WebSphere Commerce. For more information on installing WebSphere Commerce, including prerequisites, refer to the *IBM WebSphere Commerce, Version 5.4 Installation Guide*.

Conventions used in this book

This book uses the following conventions:

Boldface type indicates graphical user interface (GUI) controls such as names of fields, buttons, or menu choices.

`Monospaced type` indicates examples of text you enter exactly as shown, as well as directory paths.

Italic type is used for emphasis and variables for which you substitute your own values.

indicates additional information that can help you complete a task.

NT indicates information that is specific to WebSphere Commerce for Windows NT ®.

2000 indicates information that is specific to WebSphere Commerce for Windows ® 2000.

AIX indicates information that is specific to WebSphere Commerce for AIX® ®.

Solaris indicates information that is specific to WebSphere Commerce for Solaris Operating Environment software.

Linux indicates information that is specific to WebSphere Commerce for Linux.

400 indicates information specific to WebSphere Commerce for the IBM Eserver iSeries™™ 400® ® (formerly called AS/400® ®)

Professional indicates information specific to WebSphere Commerce Professional Edition.

Business indicates information specific to WebSphere Commerce Business Edition.

Contents

Chapter 1. An introduction to access control

The role of e-commerce has not only changed the way companies are doing business, but it has dramatically increased the kinds of relationships that they can expect to have with their customers and business partners. The Web is a key factor in delivering improved value to your existing customers, and paving the way for new customers eager to benefit from the power and increased efficiency of the Internet. Along with the clear advantages of doing business on the Web and the tremendous potential for increasing your customer base, comes the challenge of managing your business flows and trading patterns while maintaining a highly secure environment, authorizing appropriate transactions, and streamlining your work processes.

The hallmark of access control is the ability to oversee these work processes by managing the ways in which users participate in your system, based on their activities, and their business relationship to your products and services. For example, you might only want customers that have registered with your site to be able to view products for auctions in your store, and to place bids on them. Likewise, you might authorize graphic designers to customize your store pages, but you might restrict them from managing the actual content in your product catalog.

WebSphere Commerce provides you with the right tools for access management, by including more than 200 default access control policies that are automatically loaded into your system at the time of instance creation. These policies have been designed to address many of the typical access control requirements that your business needs, and can even be customized to suit your own e-commerce solution.

Managing access to activities in your electronic marketplace is an integral part of protecting your company's financial assets and resources, for ensuring secure business transactions between approved members of your site, and validating the legitimacy of your online operations. Access control becomes especially crucial in the context of e-commerce, where the entry to your business is largely affected by customer relationships that begin over the Web.

What's new in WebSphere Commerce Version 5.4?

For a list of other new features and enhancements added to WebSphere Commerce, refer to the *IBM WebSphere Commerce, Version 5.4 What's New Guide*.

Enhanced user interface

In addition to the policy editing pages accessible from the Access Management menu of the Administration Console, WebSphere Commerce now also provides additional viewer pages for viewing policies, and their related action groups, access groups, and resource groups. The policy viewing pages are seamlessly integrated into the Administration Console user interface and can be accessed using the buttons added to the existing policy editing pages.

Fine-grained control

The previous release of WebSphere Commerce Suite provided you with ″coarse-grained″ access control, which enabled you to define who could invoke what functions in the system. For example, in the previous release of WebSphere® Commerce Suite, you might have used coarse-grained access control to permit buyers to cancel orders by invoking the `cancel order` function.

Now in WebSphere Commerce, you are also provided with the ability for "fine-grained" access control, by defining who can invoke what functions against which business object instances (also referred to as `resources`). In the same example, you are not only able to permit buyers to cancel orders, but limit buyers to invoke the cancel order function only against their own orders, not the orders of other users.

The added power of fine-grained access control combined with coarse-grain access control, allows you a greater range of access management and the ability to fine-tune the activities that users are permitted to do on your site.

Separately administrated component

In the previous release of WebSphere Commerce Suite, fine-grained access control was built into the system code, which required changes to the code to institute customization of policies at the resource level.

Now, WebSphere Commerce externalizes both coarse-grained and fine-grained access control by codifying the access control policies in XML files that can be modified using the policy viewer interface included with the Administration Console tools, or by using a standard text editor.

Since both coarse-grained and fine-grained access control policies are now available separate from the product code, adapting access management for your business needs requires making changes to the information contained in the XML files, not to the product code.

Adaptable to new business processes

In today's constantly changing market, the ability to quickly customize your business environment plays an important role in remaining competitive, adjusting to market changes, and adapting to new business processes. By externalizing both coarse-grained and fine-grained policies, changes you want to make to the levels of access to your system can be done quickly and easily by modifying the policies, and not by customizing the code. More importantly, by exposing fine-grained policies that were previously only available to an engagement services team, your organization can now do many of the basic modifications to the policies yourself, and reduce the added costs of customizing WebSphere Commerce for your Web site.

Scalability

As your organization changes and grows over time, access to your system must accommodate those changes as well. As new employees join, change roles, and responsibilities, their levels of access must change accordingly to permit them to perform the activities they are required to do. Yet the task of keeping track of each individual user's activities can be time-consuming, difficult, and even impractical.

However, with WebSphere Commerce, granting access to your system can be managed implicitly by using access groups whose membership is defined by a shared set of *attributes*, rather than by their identities. Users are assigned roles, and given access according those roles. For example, Users A, B, and C might be assigned a `Buyer` role, and all buyers can be given the ability to cancel orders that have not been shipped, using the appropriate access control policy. If User A leaves the organization, the role information for User A can be deleted, whereas the access control policy associating buyer roles with canceling orders remains untouched for Users B and C.

The ability to grant users access to your system implicitly is a powerful method for managing activities, and requires far less time and effort. In addition, the effort required to manage access control becomes a factor of the number of policies you want to change, not the size of your system, the number of users belonging to your organization, or the level of business activities you conduct. The access control policies that run on your system can be applied to both small and large organizations alike. As a result, the scalability of the access control policies that run on WebSphere Commerce allows your company to continue to change and grow without hindering the structure or efficiency of your operations.

What access control means for you

Access control enables you to manage your business workflows and ensure that users only carry out those activities that are appropriate with their roles and responsibilities. Not only does WebSphere Commerce provide you with default policies that are ready to use ″out of the box,″ but it also provides you with the tools and capacity to customize the policies to suit your business needs.

The following table outlines just a few examples of how simple modifications can customize access to your business environment.

What users are allowed to do by default	**What users are allowed to do after customization**
Customers can self-register.	Only seller administrators can register new customers.
Buyers can display RFQs that they created.	Only sellers can display RFQs if the RFQ resulted in a contract.
Only customers can cancel orders they created if the order is in pending state.	Customer Service Representatives can also cancel orders in pending state, if the total product price is less than $1000.
An order can be modified by the person who created it.	Only a user from the buyer organization with the role of `purchaser` can modify an order that has been created.
Account representatives can display all accounts.	Account Representatives can only display active accounts.
Employees with the Logistics Manager role can create and modify fulfillment centers.	Employees with the Logistics Manager role can create but not modify fulfillment centers.

In the next chapter, we will take a look at how to create organizations and users, and at an access control policy in detail.

Chapter 2. Getting started

In the previous chapter, you learned about the important role that access control plays in e-commerce, and its key benefits for improving the efficiency and reliability of doing business over the Web.

In this chapter, we will discuss the basics of access management in WebSphere Commerce, such as defining organizations and users, and how access control policies are used to manage the activities that these organizations and their users perform across your system. After briefly outlining the steps you would take to set up your organizations and users, we will take a closer look at access control policies, their role in WebSphere Commerce, and study one in detail.

The chapter is divided into the following sections:
- Defining organizations and users
- Understanding access control
- How do I get started using access control?

Defining organizations and users

For site administrators, one of your first tasks after installing and configuring WebSphere Commerce is setting up and managing access to your e-commerce site. This involves creating organizations that will participate at your site, as well as defining the users that will be members of those organizations.

In some cases, the organizations joining your site may be buyer organizations, or else, you may have customers registering at your site who are engaged in a business-to-consumer relationship with your business. Regardless if you are administering a business-to-business or business-to-consumer site, defining the organizational structure of site is an important step in managing the types of access that members have to your system.

In this section, we will provide the high level steps that you need to take to define the structure of your site. If you have already set up your organizations and users, you can skip ahead to the next section on access control. Otherwise, use this section as a guideline for planning ahead.

For more details on creating organizations, users, and roles, refer to the online help, which is available from the Technical Library page:

Business

`http://www.ibm.com/software/webservers/commerce/wc_be/lit-tech-general.html`

Professional

`http://www.ibm.com/software/webservers/commerce/wc_pe/lit-tech-general.html`

We also recommend that you look at the *IBM WebSphere Commerce Fundamentals Guide, Version 5.4*.

Defining a seller organization

Typically, the seller organization is the organization that owns one or more stores on a WebSphere Commerce site. The seller organization can also have sub-organizations, or divisions, which in turn can have one or more stores. For example, the sample store, InFashion, which sells fashion merchandise, may have a women's division and a men's division, each with separate, online stores.

For now, let us assume that you are setting up a seller organization that does not have any sub-organizations. To set up the seller organization, here is a broad outline of what you will need to do:

1. Create a new organization. When you create a new organization, you will create a profile for that organization, which includes the organization's name, description, address, and contact person, as well as the organization type.
2. (Optional) Define which tasks within the seller organization require approval, such as order processing or user registration. This step is only required for a business-to-business site. Refer to the product online help for approvals documentation.
3. Assign roles to the new organization. An organization can only take on roles that have been assigned to its parent organization. Since Root Organization is an ancestor of all other organizations, it must be assigned all possible roles. WebSphere Commerce provides a set of default roles that you can start using right away. Since you are creating a seller organization, typical roles that you might assign include Seller Administrator, Store Administrator, Store Developer, Seller, and so on. See "Roles" on page 11 for a list of default roles.
4. Create users. Like organizations, you will create a profile for each user that includes the user's name, contact information, and the role assigned to that user. When assigning roles, you will select them from the list of roles you assigned to the organization in the previous step.

All of the steps outlined above can be done from the Access Management menu in the Administration Console, by a Site Administrator.

Note: In WebSphere Commerce Professional Edition, there can be only one seller organization.

Defining a buyer organization

If you are running a business-to-business site, there can be one or more buyer organizations belonging to your site. (If you are running a business-to-consumer site, you will have individual buyers registered to the Default Organization instead). After you have established which businesses will participate in a buying relationship with your site, you will have to create a buyer organization for each business. You can have as many buyer organizations as you need.

Buyer organizations are structurally similar to seller organizations. Like seller organizations, buyer organizations can also have sub-organizations or divisions, that represent different buying activities for the organization.

For now, let us assume that your buyer organizations do not have any sub-organizations. To set up a buyer organization, here is an outline of what you need to do:

1. As you did when creating the seller organization, create a new organization and define approvable tasks if needed. Again, defining approvable tasks is only required for business-to-business sites.

2. Assign roles to the new buyer organization. Since you are now creating a buyer organization, typical roles that you might assign include Buyer Administrator, Buyer (buy-side), Buyer Approver, and so on.
3. Create users and assign them roles. When assigning roles, you will select them from the list of roles you assigned to the buyer organization in the previous step.
4. Repeat the entire procedure for each buyer organization you want to add to your site.

Again, all of the steps outlined above are done from the Access Management menu in the Administration Console.

Note: In WebSphere Commerce Professional Edition, all customers belong to the Default Organization.

Understanding access control

After you have finished defining the organizations and users that will participate in your e-commerce site, you can now manage their activities through a set of policies, a process referred to as *access control*. In this next section, we will take a look at access control policies and their basic structure.

What is an access control policy?

An access control policy is a rule that describes which group of users is authorized to perform particular activities on your site. These activities can range from registration, to managing auctions, to updating the product catalog, and granting approvals on orders, as well as any of the hundreds of other activities that are required to operate and maintain an e-commerce site.

The policies are what grant users access to your site. Unless they are authorized to perform their responsibilities through one or more access control policies, users have no access to any of your site’s functions.

How does an access control policy work?

Access control policies consist of four parts: an access group, an action group, a resource group, and an optional relationship.

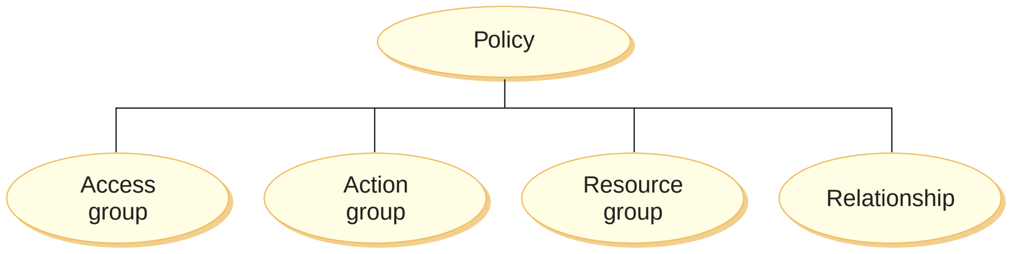
An *access group* is a group of users that share common access to a set of functions on your site. An access group generally includes users that share common attributes, such as the same role, department or skill set.

An *action group* refers to a group of actions that can be acted on the same resource. In general, action groups include actions that are associated with a common business area, or a related set of activities on your site.

A *resource group* includes the resources controlled by the policy. A resource group may include business objects such as a contract, or a set of related commands.

In some cases, a resource may only be acted on by a user that has a *relationship* to that resource. For example, only those users that create a contract might be allowed to modify it.

Figure 1. The four parts of an access control policy



Together, these four parts define a policy in WebSphere Commerce by specifying the users, the actions they can take, the business object or set of commands on which their actions are taken, and optionally, the relationship that the users have to the resource group.

For more detailed information on access groups, action groups, resource groups, and relationships, see Chapter 3, "Access control concepts" on page 9.

How do I get started using access control?

In some cases, you have to do nothing at all! This is because the default policies in WebSphere Commerce are designed to provide a basic structure of access control based on typical users in your system, and the activities they perform that are associated with their roles in an organization. The policies cover a wide range of common business activities, including membership, order creation and processing, work flow approvals, and trading, such as auctions, request for quotes and contracts. After you have defined your organizations and users, the default policies can be used as provided or customized to meet your company's individual needs.

However, before you can decide whether you want to use the default policies, or customize them, it is important understand what they look like in WebSphere Commerce. For a closer look at a default policy in detail, see "Looking at a policy in detail" on page 30.

Chapter 3. Access control concepts

WebSphere Commerce views access control as the process of verifying that users or applications have sufficient authority to access a resource. This section describes the details of several aspects of WebSphere Commerce access control.

Access control in WebSphere Commerce is accomplished using access control policies. An access control policy is a rule that describes which group of users can perform a set of actions on a set of resources. WebSphere Commerce provides a set of default access control policies. These default access control policies are specified in XML format and are designed to address many of the typical access control requirements that an e-commerce site needs. In order to understand the access control component of WebSphere Commerce, you must first understand the typical organizational hierarchy of an e-commerce site.

Organizational hierarchy

Users and organizational entities within the WebSphere Commerce member subsystem are organized into a hierarchy. This hierarchy emulates a typical organizational hierarchy, with entries for organizations and organizational units, and entries for users in the leaf nodes. The hierarchy includes an artificial organizational entity called a *root organization* at the top. All other organizational entities and users are descendants of this root organization. Under the root organization there can be one seller organization and several buyer organizations; all these organizations can have one or more sub-organizations under them. Buyer or Seller Administrators are the heads of the organizations, and they are responsible for maintaining their organizations. On the seller organization side, each sub-organization can have one or more stores within it. Store Administrators are responsible for maintaining the stores. The following diagram shows the organizational hierarchy of a business-to-business e-commerce site.

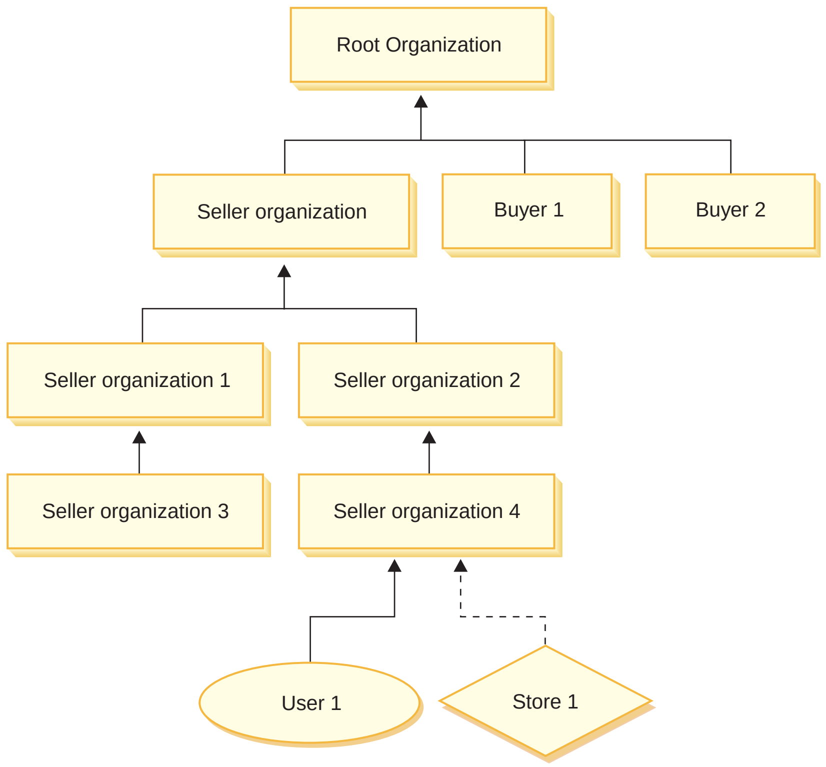


Figure 2. Organizational hierarchy of a business-to-business site

Root organization

The root organization is at the top of the organizational hierarchy. A Site Administrator has super-user access to perform any operation within WebSphere Commerce. The Site Administrator installs, configures, and maintains WebSphere Commerce and its associated software and hardware. This role typically controls access and authorization (that is, creating and assigning members to the appropriate role) and manages the Web site. The Site Administrator can assign roles to users and specify the organization(s) for which the user plays the role. The Site Administrator must assign a password to each administrator to ensure that only authorized parties can access confidential information. This provides a way to control key responsibilities, such as updating a catalog or approving a request for quotation (RFQ).

Note: It is possible for a user to play roles in an organization other than their parent organization.

In a WebSphere Commerce site, there is one seller organization. In a business-to-business site, there are also one or more buyer organizations. The Site Administrator may define both the access control policies of the seller organization (that owns the store) as well as the access control policies of each organization that buys from the store. In a business-to-consumer site, there are no buyer organizations. Business-to-consumer customers are modeled as members of the default organization.

Organizations (seller)

Both in business-to-business and business-to-consumer sites, the Site Administrator creates one top-level seller. Underneath this seller organization, other sub-organizations or organization units can be created. Any of these sell-side organizational entities can own one or more stores. The Site Administrator then defines any special access control policies for a seller organization, and assigns a Seller Administrator to manage that organization. The Seller Administrator registers users, assigns them different roles to fit the organization's business needs according to the access control policies pertaining to that organization.

The Seller Administrator's responsibilities are summarized as follows:

- Create sub-organizations that can own stores. Optionally, define which processes within the organization require approval. This step is only required in a business-to-business site.
- Assign roles to the sub-organizations.
- Create users.
- Assign roles to users.

Organizations (buyer)

In a business-to-business site, the Site Administrator creates one or more buyer organizations, depending on the business needs. The Site Administrator then defines any special access control policies for a buyer organization and assigns a Buyer Administrator to manage the buyer organization. The Buyer Administrator registers users and assigns them different roles to fit the organization's business needs, according to the access control policies pertaining to that organization.

The Buyer Administrator's responsibilities are summarized as follows:

- Create and administer the sub-organizations within the buyer organization. Optionally, define which processes within the organization require approval. This step is only required in a business-to-business site.
- Assign roles to the sub-organizations.
- Create users.
- Assign roles to users.

Note: The Site Administrator can modify and manage the access control policies of the buyer organization if appropriate. For more information on the Site Administrator's tasks, see "Site Administrator" on page 12.

Roles

As mentioned above, WebSphere Commerce provides default sets of roles. The Site Administrator must assign specific roles to every organization before assigning users to those roles. An organization can only take on roles that have been assigned to it's parent organization. Similarly, a user can only take on roles that have been assigned to their parent organization .

All roles in WebSphere Commerce are scoped to an organization. For example, a user plays the Product Manager role for Organization X. The parent organization of this user must also be assigned the Product Manger role for itself. The access control policies could then be setup such that this user can only perform product management operations within the context of Organization X and its sub-organizations.

Note: Assigning roles to users and organizations is done in the MBRROLE table.

The default roles that come with WebSphere Commerce can be grouped into the following categories:

- Site operations
- Site and content development
- Marketing management
- Product management
- Sales management
- Logistics and operations management
- Organizational management

Site operations

The following technical operations roles are supported by WebSphere Commerce:

- Site Administrator
- Store Administrator

Site Administrator

The Site Administrator installs, configures, and maintains WebSphere Commerce and the associated software and hardware. The Administrator responds to system warnings, alerts, and errors, and diagnoses and resolves system problems. This role typically controls access and authorization (creating and assigning members to the appropriate role), manages the Web site, monitors performance, and manages load balancing tasks. The Site Administrator may also be responsible for establishing and maintaining several server configurations for different stages of development such as testing, staging, and production. This role also handles critical system backups and resolves performance problems.

Store Administrator

The Store Administrator manages the store assets, and updates and publishes changes to taxes, shipping, and store information. The Store Administrator can also manage the access control policies for the organization. The Store Administrator, usually the lead on the store development team, is the only role on the team with the authority to publish a store archive (the Site Administrator can also publish a store archive). The Store Administrator is usually Web-literate and has a thorough knowledge of the store's business procedures.

Site and content development

WebSphere Commerce supports the Store Developer site and content development role.

Store Developer

Store Developers create Java™ Server Pages files and any necessary customized code and can modify any of the standard functionality included with WebSphere Commerce. Once a store archive has been created, Store Developers have the authority to make changes to it manually or by using the Store Profile notebook and Tax and Shipping notebooks. They do not have the authority to publish the store archive to the WebSphere Commerce Server.

Logistics and operations

WebSphere Commerce supports the following logistics and operations management roles:

- Logistics Manager

- Operations Manager
- Receiver
- Returns Administrator
- Pick Packer

Logistics Manager

Business The Logistics Manager, sometimes called the Shipping Manager, manages and negotiates bulk freight or shipping from carriers to warehouse, and to individual customers. This role is responsible for ensuring the company uses the best shippers at the best costs to meet company strategy. Shipping is an important aspect of customer service and may be a key success factor for the online business.

Operations Manager

B2C This role manages order processing, ensuring that orders are properly fulfilled, payment is received, and orders are shipped. The Operations Manager can search for customer orders, view details, manage order information, and create and edit returns.

Pick Packer

The Pick Packer picks products from fulfillment centers and packs the products for shipping to customers. The Pick Packer also manages pick tickets and packing slips, which are used to confirm shipment of products during order fulfillment.

Receiver

The Receiver receives inventory at the fulfillment center, tracks expected inventory records and ad hoc receipts for ordered products, and receives returned products as a result of customer returns.

Returns Administrator

The Returns Administrator manages the disposition of returned products.

- List returns
- Lists returned products
- Dispositions returned products

Product management

The following product management roles are supported by WebSphere Commerce:

- Buyer (seller side)
- Category Manager
- Product Manager or Merchandising Manager

Buyer (seller-side)

The buyer purchases merchandise for sale. The buyer handles relations with vendors or suppliers and negotiates to obtain the desired product with favorable terms for such things as delivery and payment options. The buyer may set prices. Inventory is managed by the buyer in order to determine the quantities to buy and ensure that stock is properly replenished.

Category Manager

The category manager manages the category hierarchy by creating, modifying, and deleting categories. The category hierarchy organizes products or services offered by the store. The Category Manager also manages products, expected inventory records, vendor information, inventory, and return reasons.

Product Manager/Merchandising Manager

The ▶ Business Merchandising or ▶ B2C Product Manager traces customer purchases, suggests discounts, and determines the best way to display, price, and sell products in the online store

- Performs all Category manager tasks
- Performs all Marketing manager tasks

Sales management

The following business relationship management roles are supported by WebSphere Commerce:

- Sales Manager
- Account Representative
- Customer Service Supervisor
- Customer Service Representative

Sales Manager

Sales Managers acquire and retain customers, meet sales forecasts, provide incentives for increased customer business, contract management, set pricing terms, work with product manager to establish inventory forecasts, and work with the Marketing Manager for promotions

Account Representative

Account representatives work with individual accounts to build relationships, and manage customer service issues. They may be authorized to change contract pricing, negotiate contracts, profiles, and analyze profitability by account category.

Customer Service Supervisor

This role has access to all customer service tasks. The Customer Service Supervisor manages customer inquiries (such as customer registration, orders, returns, and auctions) and has authority to complete tasks that cannot be accessed by a Customer Service Representative, such as approving system-denied returns records, and contacting customers regarding payment exceptions (such as credit card authorization failures).

Customer Service Representative

No matter how well an online business is designed to provide a customer with self-service features, there will be some types of customers or some occasions when even the most web-literate customer will require personal contact. Most online businesses provide an e-mail, fax or contact number for the customer to obtain direct service. It is the responsibility of the customer service representative to handle all inquiries from the customer.

Marketing management

WebSphere Commerce supports the marketing management role of Marketing Manager.

Marketing Manager

The Marketing Manager communicates the market strategy and brand messages to the customers. This role monitors, analyzes, and understands customer behavior. In addition, the marketing manager creates or modifies customer profiles for targeted selling, and creates and manages campaigns and promotions. Campaign event planning can be handled by a team comprising the Merchant, Marketing Manager, and Merchandising Manager.

Organizational management

WebSphere Commerce supports the following organizational management roles:

- Seller Administrator
- Buyer Administrator
- Buyer Approver

Seller Administrator

The Seller Administrator manages the information for the selling organization. Seller administrators create and administer the sub-organizations within the selling organization and the various users in the selling organization, including the assignment of the appropriate business roles.

Buyer Administrator

The Buyer Administrator manages the information for the buying organization. They create and administer the sub-organizations within the buying organization and manage the various users including approving users as buyers. Other buy-side roles such as buyer approvers and additional buyer organization administrators may be created and managed.

Buyer Approver

A Buyer Approver is an individual in the buying organization who approves orders made by buyers before the order is submitted for purchase with the seller.

Access control policy

An access control policy authorizes a group of users to perform a set of actions on a set of resources within WebSphere Commerce. Unless authorized through one or more access control policies, users have no access to any functions of the system. To understand access control policies you need to understand four main concepts: users, actions, resources, and relationships. Users are the people who use the system. Resources are objects in the system that need to be protected. Actions are the activities that users can perform on the resources. Relationships are optional conditions that exist between users and resources.

Elements of an access control policy

An access control policy consists of four elements:

Access group
The group of users to which the policy applies.

Action Group
A group of actions performed by the user on resources.

Resource group
The resources controlled by the policy. A resource group may include business objects like `contract` or `order`, or a set of related commands such as, all the commands that users of a particular role can perform.

Relationship (optional)
Each resource class can have a set of relationships associated with it. Each resource can have a set of users that fulfill each relationship. For example, a policy could specify that only the creator of an order can modify it. In this case, the relationship would be `creator` , and it is between the user and the order resource.

Access control policy concepts

Access control policies grant users access to your site. Unless they are authorized to perform their responsibilities through one or more access control policies, users have no access to any of your site's functions.

Each access control policy takes the following form:

```
AccessControlPolicy [AccessGroup,ActionGroup,ResourceGroup,Relationship]
```

The elements in the access control policy specify that a user belonging to a specific access group is permitted to perform actions in the specified action group on resources belonging to the specified resource group, as long as the user satisfies a particular relationship with respect to the resource. The relationship is only specified when needed. For example, `[AllUsers,UpdateDoc,doc,creator]` specifies that all users can update a document, if they are the creator of the document.

The following sections describe conceptual information and terminology associated with access control.

Member groups

The Member subsystem in WebSphere Commerce allows you to create member groups, which are groups of users categorized for various business reasons. The groupings can be used for many purposes, for example, access control purposes, approval purposes, as well as for marketing purposes such as calculating discounts and prices, and displaying products. A member group of type Access Group (-2) is for access control purposes, while a member group of type User Group (-1) is for general use. A member group is associated with member group types in the MBRGRPUSG table.

Access groups: A member group of type Access Group (-2) is for grouping users for access control purposes. An access group is one element of an access control policy, and is defined as a group of users defined specifically for access control purposes. The criteria for membership in an access group is usually based on roles, the organization to which the user belongs, or the user's registration status. For example, the access group called `Buyer Administrators` is a group whose users play the role of Buyer Administrator.

WebSphere Commerce includes a number of default roles, and corresponding to each role is a default access group that implicitly references that role. Roles can be used as attributes to add users to an access group based on the type of activities they perform in the site. For example, by default there is a role called Seller Administrator and a corresponding access group called `Seller Administrators`. A Site Administrator uses the WebSphere Commerce Administration Console to create, maintain, and delete access groups for a site. A Buyer Administrator or a Seller Administrator uses the WebSphere Commerce Organization Administration Console to assign roles to users or to explicitly assign users to access groups. Access groups can be implicit, explicit or both.

Implicit access group: An implicit access group is defined by a set of criteria. Anyone who satisfies the criteria is a member of the group. The criteria are usually based on a user's roles, parent organization, or registration status. The implicit conditions that define membership in a member group are in the `CONDITIONS` column of the `MBRGRP` table. Using implicit access groups that specify the attributes of users, makes it easy to authorize access to similar users without having to explicitly assign and unassign individual users. It also eliminates the need to update the members of a group when a user's attributes change. A simple criterion for an access group is to include everyone that has been assigned a specific role, regardless for which organization the user plays the role. A more complex criterion would be to specify that only users that play one of a possible set of roles for a particular organization would belong to the access group.

Explicit access group: It is possible to explicitly add or remove a user from a member group. Both of these explicit specifications can be done using the `MBRGRPMBR` table. An explicit access group contains explicitly assigned users who may or may not share common attributes. This also allows you to exclude individuals that satisfy the conditions for inclusion in an implicitly defined group, but that you want excluded anyway.

User groups: A member group of type User Group (-1) is a collection of users defined by the merchant, who share a common interest. User groups are similar to clubs that are offered by large stores for their frequent or preferred customers. Being part of a user group can entitle customers to discounts or other bonuses for purchasing products. For example, if market research shows that senior customers repeatedly purchase travel books and luggage, you can assign these customers to a member group called `Seniors' Travel Club`. Likewise, you can create a user group to reward frequent customers for their business.

Actions

Generally, an action is an operation that is performed on a resource. In role-based policies for controller commands, the action is `Execute` and the resource is the command being executed. In role-based policies for Views, the action is the name of the view, and the resource is `com.ibm.commerce.commands.ViewCommand`. For resource-level access control, actions typically map to WebSphere Commerce commands, and the resource is usually the remote interface of a protected EJB (Enterprise Java Bean). For example, the controller command `com.ibm.commerce.order.commands.OrderCancelCmd` operates on the `com.ibm.commerce.order.objects.Order` resource. Lastly, the `Display` action is used to activate databean resources.

The WebSphere Commerce Administration Console can be used by a Site Administrator to associate existing actions with action groups, but not for creating new actions. New actions can be created by defining them in an XML file and then loading them to the database. Actions are stored in the `ACACTION` table.

Action groups

Action groups are groups of related actions. An example of an action group is the `AccountManage` group that includes the following commands:

- `com.ibm.commerce.account.commands.AccountDeleteCmd`
- `com.ibm.commerce.account.commands.AccountSaveCmd`

Only the Site Administrator can create, update, and delete action groups. This can be done from the WebSphere Commerce Administration Console and through XML. Action groups are stored in the `ACACTGRP` table. Actions are associated with action groups in the `ACACTACTGP` table.

Resource category

Resource category refers to a class of resources that need to be protected by access control. Resources must implement the Protectable interface information. Resource categories are Java classes such as order, RFQ, and auction. Resources are the instances of these classes. For example, Auction1 created by Auction Administrator A is one resource; Auction2 created by Auction Administrator B is another resource. These two resources belong to the resource category: auction.

Note: For more information on the Protectable interface, see the *IBM WebSphere Commerce Programmer's Guide*.

Resource categories are defined in the `ACRESCGRY` table, and for convenience, are sometimes referred to as resources. A Site Administrator can associate existing resource categories with resource groups, using the WebSphere Commerce Administration Console. New resource categories can be created using XML.

Resources

Resources are any objects in the system that need to be protected. For example, RFQs, auctions, users, and orders are some of the resources in WebSphere Commerce which need to be protected. Each resource has an owner. The ownership of the resource is used to determine which access control policies apply to it. Access control policies have an owner, which is an organizational entity. A policy is only applied to resources that are owned by the same organizational entity that owns the policy. Policies that are owned by ancestor organizational entities are also applied to the resource.

Controller command resources: For role-based access control for controller commands, the policy is structured such that the `Execute` action is being performed on the controller command resource. These policies are intended to restrict the execution of controller commands to users with a specified role. The access group for these policies is usually those with a single role, for example, Product Managers (those with the Product Manager role). Then, the resource group would be the set of controller commands that a product manager can execute.

While enforcing role-based access control on a controller command, the owner of the command must be determined. This is done by calling the `getOwner()` method on the command if it has been implemented. Usually this method is not implemented, so WebSphere Commerce Runtime will evaluate it by doing one of the following:

- Use the organization that owns the store that is currently in the command context.
- If there is no store in the command context, use the Root Organization as the owner.

Data bean resources: Not all data beans require protection. Within the existing WebSphere Commerce application, data beans that require protection already implement the required access control. The question of what to protect comes into play when you create new data beans. Deciding which resources to protect depends upon your application. A data bean should be protected (directly or indirectly), if the information to be displayed is not sufficiently protected by the role-based access control on the view, that corresponds to the JSP (Java Server Page) that contains the data bean.

If a data bean needs to be protected and can exist on its own, it should be directly protected. If the existence of a data bean depends upon the existence of another data bean, then it should delegate to the other data bean for protection. An example of a data bean that would be directly protected is the `Order` data bean. An example of a data bean that would be indirectly protected is the `OrderItem` data bean, as it cannot exist without `Order` data bean. Refer to the *WebSphere Commerce 5.4 Programmer's Guide* for more information on how to protect the data bean resource.

Data resources: Data resources refer to business objects that can be manipulated such as, auctions, orders, RFQs, and users. These are usually protected at the enterprise bean level, but it is possible to protect any class, as long as it implements the Protectable interface. Data resources are protected using resoure-level access control checks. The common way of doing this by returning data resources in the `getResources()` method of a controller or task command. For more information see the *WebSphere Commerce 5.4 Programmer's Guide* .

Resource groups

A resource group identifies a set of related resources. A resource group can include business objects such as a contract or a set of related commands. In access control, resource groups specify the resources to which the access control policy authorizes access.

Resource groups are defined in the `ACRESGRP` table. Site Administrators can manage resource groups and associate resources with resource groups using the WebSphere Commerce Administration Console, or by using XML.

Implicit resource groups: Implicit resource groups define resources that match a certain set of attributes. One of these attributes must be the Java class name. Other attributes may include status, store ID, price, etc. For example, you could create an implicit resource group that includes all orders that have pending status (`ORDERS.STATUS=P`). Implicit resource groups are usually used for grouping resources that will be used in resource-level policies, when the resources share a common attribute beyond the Java class name.

Implicit resource groups are defined using the `CONDITIONS` column of the `ACRESGRP` table. Simple implicit resource groups can be created using the WebSphere Commerce Administration Console. Increasingly complex groups can be created using XML.

Explicit resource groups: Explicit resource groups are specified by associating one or more resource categories to a resource group. This association is done in the `ACRESGPRES` table. Adding a resource category to a group explicitly, by listing its Java class name, lets you group individual resources that might not necessarily share common attributes.

Relationships

Each resource may have some kind of relationship associated with it, and a set of members that fulfill each relationship. For example, all resources have a relationship of *owner*, which is fulfilled by the owner of the resource. Other relationships can include recipients of documents and the creator of an order. These resource relationships are important in determining who can perform certain actions on a particular instance of a resource. For example, the creator of a document may not be able to delete it, but perhaps an auditor may. Similarly, a reviewer may only be able to read and approve a document, but not forward it or perform other operations.

Relationships are stored in the ACRELATION table, and are optionally specified in an access control policy, using the ACRELATION_ID column of the ACPOLICY table. When evaluating a policy that requires the fulfillment of a relationship between the user and the resource, the fulfills(`Long Member, String relationship`) method on the resource will be called to evaluate it. When comparing these relationships to relationship groups, these relationships are sometimes referred to as simple relationships.

Relationship groups: Access control policies can specify that a user must fulfill a particular relationship with respect to the resource being accessed, or they can specify that a user must fulfill the conditions specified in a relationship group. In most cases, a relationship is sufficient. However, if more complex relationships are needed, a relationship group can be used instead. A relationship group allows you to specify multiple relationships and also a chain of relationships. Both of these are done using a relationship chain construct. A relationship chain is a construct that can express a simple relationship (directly between a user and the resource), but can also be used to express a series of relationships between the user and the resource. For example, in order to express that a user must have a role in an organization that has a relationship (other than the owner relationship) with the resource, one must use a relationship group. In this example, there is a role relationship between the user and the organization, and a relationship between the organization and the resource.

Comparing relationships and relationship groups: In most cases, using a relationship should satisfy the access control requirements for your application since, conceptually, most relationships are directly between a user and the resource. For example, the policy states that the user must be the creator of the resource. If however, you need to specify multiple relationships, a relationship group should be used. For example, the policy states that the user must be the creator or the submittor of the resource.

Relationship groups are also needed to express a chain of relationships between a user and the resource. In a chain of relationships, there is no direct relationship between the user and the resource for example, a user belongs to the buying organization specified by an order. In this case, the user has a child relationship with the organization, and that organization has a buying relationship with the order.

Relationship chains: Each relationship group consists of one or more RELATIONSHIP_CHAIN open conditions, grouped by `andListCondition` or `orListCondition` elements. A relationship chain is a series of one or more relationships. The length of a relationship chain is determined by the number of relationships it consists of. This can be determined by examining the number of `<parameter name= "X" value="Y"/>` entries in the XML representation of the relationship chain. The following is an example of a relationship chain with a length of one.

```
<openCondition name="RELATIONSHIP_CHAIN">
<parameter name="RELATIONSHIP"
value="aValue"/>
</openCondition>
```

For relationship chains of length one, the `<parameter name="Relationship" value="something">` element specifies a direct relationshp between the user and the resource. The value attribute is the string representing the relationship between the user and the resource. It must also correspond to the relationship parameter of the `fulfills()` method on the protectable resource.

When a relationship chain has a length of two, it is a series of two relationships. The first ,`<parameter name= "X" value="Y"/>`, element is between a user and an organizational entity. The last, `<parmeter name= "X" value="Y"/>`, element is between that organizational entity and the resource. The following is an example of a relationship chain with a length of two.

```
<openCondition name=RELATIONSHIP_CHAIN">
<parameter name="aValue1" value="aValue2"/>
<parameter name="RELATIONSHIP" value="aValue3"/>
</openCondition>
```

The `aValue1` possible values include `HIERARCHY` and `ROLE`. `HIERARCHY` specifies that there is a hierarchical relationship between the user and the organizational entity in the membership hierarchy. `ROLE` specifies that the user plays a role in the organizational entity.

If the value of `aValue1` is `HIERARCHY`, the possible values include `child`, which returns the organizational entity for which the user is a direct child in the member hierarchy. If the value of `aValue1` is `ROLE`, possible values include any valid entries in the NAME column of the ROLE table which return all of the organizational entities for which the current user plays this role.

The `aValue3` entry, is a string representing the relationship between one or more organizational entities retrieved from evaluating the first parameter and the resource. This value corresponds to the relationship parameter of the `fulfills()` method on the protectable resource. If more than one organizational entity was returned by evaluating parameter `aValue1` , this part of the RELATIONSHIP_CHAIN is satisfied if at least one of these organizational entities satisfies the relationship specified by parameter `aValue2`.

Note: A relationship group that consists of a single relationship chain with a single parameter element, is functionally equivalent to a simple relationship. In this case, it is easier to use relationship instead of relationship group in the policy. For more information on defining relationship groups, see “Defining relationship groups” on page 85.

Resource and policy ownership

All policies are owned by an organizational entity. All access control resources also have an owner that is usually an organizational entity; for example, an order is owned by the organization that owns the store where the order was placed. Users can also own resources, for example a registered user owns his own user registration information. Ownership of resources and access control policies is important when determining which policies to apply to a certain resource. For a given resource, the policies that belong to its owning organizational entity and that owner’s ancestor organizational entities are applied.

Types of access control policies

There are two types of access control policies:

- Standard policies
- Template policies

Standard policies

Standard policies have a fixed owner. For example, if a standard policy is owned by Seller Organization, it will only apply to resources that are owned by Seller Organization and to resources that are owned by its descendant organizational entities, if they exist. Since the Root Organization is the ancestor organization of all other organizations in WebSphere Commerce, any policy that is owned by Root Organization (member ID = -2001), by definition applies to all resources in the site. Thus, standard policies that are owned by the Root Organization are sometimes referred to as Site-level policies.

Standard policies that are not owned by Root Organization are referred to as organizational level policies, since they do not apply site-wide; only to the resources that are owned by the policy owner or by any of its descendant organizational entities. A store administrator can manage the policies for his own organizational entity and its descendant organizational entities. Site administrators can modify all policies.

Template policies

Template policies have a dynamic owner. Template policies apply dynamically to the organizational entity that owns the resource and its ancestor organizational entities. For example, consider that there are 10 organizations under Root Organization, and each one wants to ensure that Store Administrators can modify only resources that are owned by the Organization for which they play their role. There are two ways to set this up:

1. Have one template policy that will dynamically apply to any of the 10 organizations, depending on the resource that is being accessed. The criteria for the access group in the template policy can also be dynamic. For example, if a user is trying to access a resource owned by Organization 3, the owner of the template policy will dynamically change to Organization 3, and the access group will also dynamically scope itself to Organization 3, that is, the user must play the role of Store Administrator for Organization 3.
2. Have 10 policies, each one owned by one of the 10 organizations. The access group for Organization 1 would specify that the user must play the Store Administrator role for Organization 1. The access group for Organization 2 would specify that the user must play the Store Administrator role for Organization 2, and so on.

The advantage of the first solution is that there is only one physical copy of the policy, but 10 logical copies. Template policies can be managed by a site administrator.

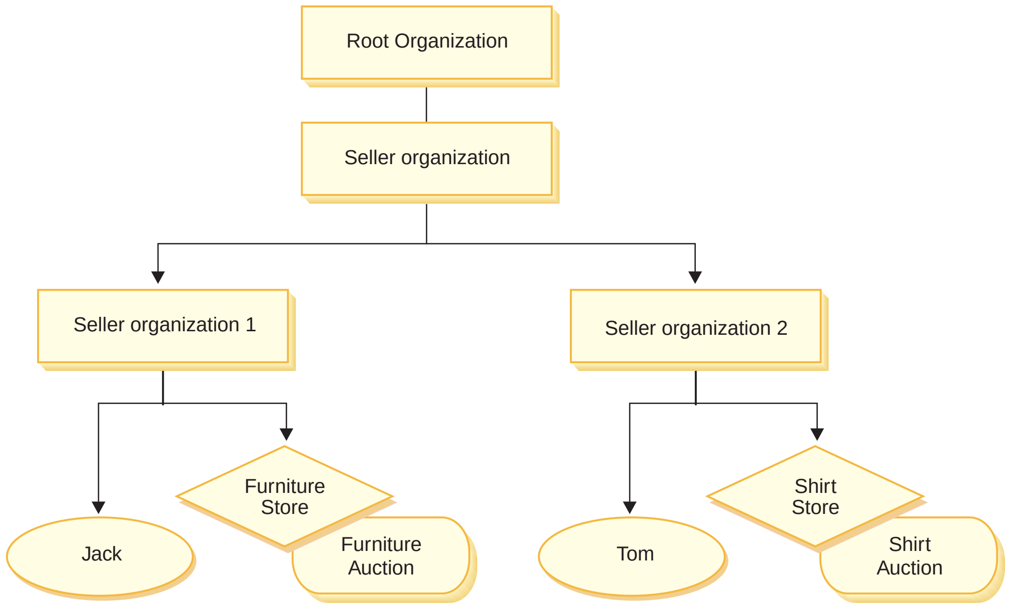
Overriding Template Policies: Another feature of template policies is that they can be overridden for specified organizational entities. Going back to the example above, if an 11th organization entity is added to the WebSphere Commerce site, but this newest organizational entity does not want the above template policy to apply to it, there is a way of specifying this. An entry must be added to the `ACORGPOL` table, specifying the policy id of the template policy, and the organizational entity ID of the 11th organization. This can also be done through the WebSphere Commerce Administration Console, when a Store Administrator deletes or updates a template policy, in the context of particular organization.

When overriding a template policy for a descendant organization of Root Organization, the template policy will still apply at the Root Organization level. If the template policy is being overrided with a more restrictive policy at the descendant organization level, you should override the template policy at the Root Organization level as well. The only way to override a template policy for the Root Organization is through the database, by running the following SQL:

```
insert into ACORGPOL (acpolicy_id, member_id) values ( (select acpolicy_id from
ACPOLICY where policyname = 'policyToOverride'), -2001)
```

Levels of access control

There are two broad levels of access control in WebSphere Commerce: command level (also know as role-based) and resource level (also known as instance-level).



Command-level or role-based access control

Command-level or role-based access control is coarse access control. It determines "who can do what". With role-based access control, you can specify that all users of a particular role can execute certain commands. Consider the access control policy, Sellers can execute sellers commands. In this policy, one of the sellers commands is the `ModifyAuction` command. In the figure above, Jack and Tom both are sellers, so both of them can modify auctions.

Role-based access control is used for controller commands and views. This type of access control does not consider the data resource that the command acts upon. It only determines if the user is allowed to execute a particular controller command or view.

This level of access control is mandatory and is enforced by the Runtime. All controller commands must be protected by command-level access control. In addition, any view that can be called directly, or that can be launched by a redirect from another command (in contrast to being launched by forwarding to the view) must be protected by command-level access control.

Command-level access control for controller commands: Whenever you run a controller command, an access control policy must exist that grants users to perform the `Execute` action on the command resource. The resource is the interface name of the controller command. The access group is usually geared to a single role. For example, you can specify that users with the Account Representative role can execute any command in the `AccountRepresentativesCmdResourceGroup` resource group.

Command Level Access Control for Views: When a view is called directly from the URL, or is the result of a redirect from a command, it must have an access control policy. Such a policy must have the `viewname` specified as an action, in the `ACACTION` table. This action must then be associated with an action group, using the `ACACTACTGP` table. This action group must then be referenced in the appropriate command level policy, in the `ACPOLICY` table.

Instance-level or resource-level access control

Instance-level or resource-level access control policies provide granular access control, determining `who can do what command on which resources`. The previous example of a role-based access control policy that allows Sellers to modify auctions, can be fine-tuned for resource-level access control to be, Sellers can modify auctions owned by the organization for which they play their role. In 23, Jack has the seller role for Seller Organization 1. Tom has the seller role for Seller Organization 2. Jack creates a furniture auction at the furniture store. Tom creates a shirt auction at the shirt store. Jack can modify the furniture auction, but *not* the shirt auction. Tom can modify the shirt auction, but *not* the furniture auction.

To summarize, first the system does a command-level access check. If the user is allowed to execute a command, a subsequent resource-level access control policy is done to determine if the user can access the resource in question.

Resource level access control applies to commands and databeans.

Resource-level access control for commands: After the command level access control checking has been completed, if access has been granted, then resource level checking is done in one of the following two cases:

- The command implements `getResources()` — this method specifies the instances of resources that need to be checked against the current action; where the command is now the action. The WebSphere Commerce Runtime will enforce that the current user has access to all of the resources specified by `getResources()`. By default, `getResources()` returns null, that is, it does not perform any resource level checking.
- The command calls `checkIsAllowed(Object Resource, String Action)` — in cases where the command writer does not know which resources need to be checked at the time that getResources() is called by the Runtime, the command can call this `checkIsAllowed()` method, as needed, to determine if the current action and resource pair is authorized. The action is usually the interface name of the current command. When this method is called, if access is denied, an exception will be thrown: `ECApplicationException( ECMessage._ERR_USER_AUTHORITY, ..)`

Resource level access control for databeans: As explained above, views are protected by command level policies, which are usually based on roles. For example, the command level policy may specify that a Seller Administrator has access to a specific view. It is often necessary to further ensure that the databeans on the JSP are all related to the organization for which the user plays the Seller Administrator role. This is done by having all databeans that need protection (directly or indirectly), implement the Delegator interface. These databeans delegate to a primary (independent) databean which in turn implements the Protectable interface. A primary databean would delegate to itself, and therefore implement both interfaces. Then, whenever a databean is invoked using the Databean Manager's `activate()` method, the WebSphere Commerce Runtime will ensure that there is a policy which grants the current user the authority to perform the `Display` action on the primary databean resource.

How access control prevents unauthorized actions

This section explains how policy-based access control works to ensure that users can perform only actions for which they are authorized.

Checking for authorization before performing a user-initiated action

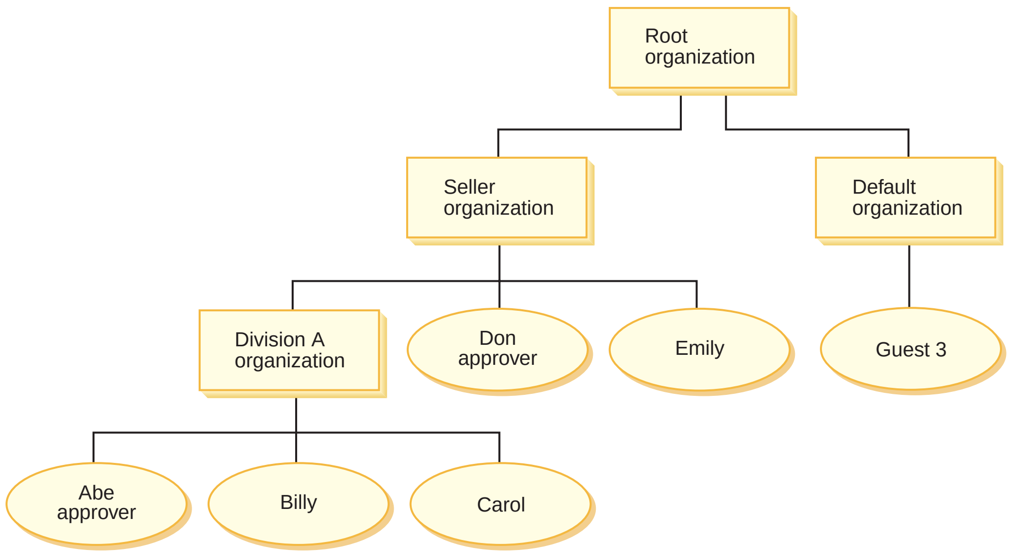
Policy Manager is the access control component that determines whether or not the current user is allowed to execute the specified action on the specified resource. Access control policies are specified in XML format. During instance creation, the default policies are loaded into the appropriate database tables. When WebSphere Commerce Application Server is started up, the access control information is cached in memory so the Policy Manager can quickly check a user's authorization when called to do so. If access control information is changed in the database through the WebSphere Commerce Administration Console, or by loading XML policy data, the access control cache needs to be updated. This can be done by updating the Access Control registry in the WebSphere Commerce Administration Console. Restarting WebSphere Commerce will also result in updating the cache.

When a user attempts to perform an access control protected action, an access control check will be done to make sure that the user is authorized. The Policy Manager looks for all the access control policies that apply to the organization that owns the resource. Then it checks those policies to evaluate if the user is authorized to perform the action on the target resource. If there is at least one such policy, the Policy Manager grants access, otherwise, access is denied.

Evaluating access control policies

This section can be used as a guide to evaluating access control policies. In this section, you are presented with a scenario and guided through an example of how to evaluate a standard and a template access control policy. Each section begins with a description of related policies, and scenarios using each policy. For more information on standard and template policies, see "Types of access control policies" on page 21.

The following diagram graphically displays the scenario:



Organizational hierarchy

From the diagram, you can see the following four organizations are in the site:

- Root organization
- Seller organization

- Default organization
- Division A organization

As you can see, Root organization is the parent of Seller organization and Default organization. Seller organization is the parent of Division A organization

Users

In the diagram, Don and Emily are registered to the Seller Organization. Abe, Billy and Carol are registered to Division A organization. Guest 3 has not registered, but for access control purposes, implicitly belongs to the Default Organization.

Roles

Don has the approver role for the Seller Organization. Abe has the approver role for the Division A organization.

Access Groups

The following access groups are used in this scenario:

- Registered users: This group implicitly includes all users that are registered.
- Approvers for Seller: This group implicitly includes all users that have the role of approver for the Seller organization.
- Approvers for Division A: This group implicitly includes all users that have the role of approver for the Division A organization.

Documents

The document object is a protected resource. The owner of a document is defined to be the organization where it was created.

Access control requirements for updating documents

The following are the access control requirements for updating documents:

1. Registered users can update a document of which they are the creator.
2. Approvers for Division A can update documents owned by Division A, but not documents owned by Seller. Approvers for Seller organization can update documents owned by both Division A, and Seller organization.

Evaluating standard policies

This section guides you through the standard policies and the scenarios to evaluate them.

Access control polices related to updating documents

The following is the policy format and the access control policies that relate to updating documents:

Policy Format: `[Access Group, Action Group, Resource Group, Relationship]`

Policy 1:

```
[Registered Users, Execute Command Action Group,  Update Document
Resource Group, - ]
```

This is a standard role-based policy owned by the Root organization. In this policy, registered users can execute `Update Document` commands.

Policy 2:

```
[Registered Users, Update Document Action Group, document, creator ]
```

This is a standard resource-level policy owned by the Root organization. In this policy, registered users can update a document if they are the creator of that document.

Policy 3:

```
[Approvers for Seller, Update Document Action Group, document, - ]
```

This is a standard resource-level policy owned by Seller organization. In this policy, approvers for Seller can update documents that are owned by Seller.

Policy 4:

```
[Approvers for Division A, Update Document Action Group, document, - ]
```

This is a standard resource-level policy owned by Division A organization. In this policy, Approvers for Division A can update documents that are owned by Division A.

Scenarios

Scenario 1 : Billy attempts to update his own document: The following is the access control evaluation for this scenario:

Command - level check:

1. There is no store ID specified, so the owner of the command is set to Root organization. So, only policies owned by Root Organization will be used to evaluate whether the user has command-level access: policies 1 and 2 are owned by Root organization.
2. Policy 1 grants access, since Billy is a member of the Registered Users access group and he is performing the `Execute` action on the `Update Document` command resource.

Resource - level check:

1. The `Update Document` command specifies that the document resource is to be protected. Billy's document is owned by Division A. So, only policies owned by Division A and its ancestor organizations will apply: policies 1, 2, 3 and 4.
2. Policy 2 grants access since Billy is a member of the Registered Users access group, he is performing the `Update Document` command action on the document resource, and he fulfills the creator relationship with the document.

Since Billy passed both the command-level and resource-level access control checks, he can update his own document.

Scenario 2: Don attempts to update Carol's document: The following is the access control evaluation for this scenario:

Command - level check:

1. There is no store ID specified, so the owner of the command is set to Root organization. So, only policies owned by Root Organization will be used to evaluate whether the user has command-level access: policies 1 and 2 are owned by Root organization.
2. Policy 1 grants access, since Don is a member of the Registered Users access group and he is performing the `Execute` action on the `Update Document` command resource.

Resource - level check:

1. The `Update Document` command specifies that the document resource is to be protected. Carol's document is owned by Division A. So, only policies owned by Division A and its ancestor organizations will apply: policies 1, 2, 3 and 4.
2. Policy 4 grants access since Don is a member of the Approvers for Seller access group, and he is performing the `Update Document` command action on the document resource

Since Don passed both the command-level and resource-level access control checks, he can update Carol's document.

Scenario 3: Abe attempts to update Emily's document: The following is the access control evaluation for this scenario:

Command - level check:

1. There is no store ID specified, so the owner of the command is set to Root organization. So, only policies owned by Root Organization will be used to evaluate whether the user has command-level access: policies 1 and 2 are owned by Root organization.
2. Policy 1 grants access, since Abe is a member of the Registered Users access group and he is performing the `Execute` action on the `Update Document` command resource.

Resource - level check:

1. The `Update Document` command specifies that the document resource is to be protected. Emily's document is owned by Seller organization. So, only policies owned by Seller organization and its ancestor organizations will apply: policies 1, 2 and 3.
2. Policy 3 does NOT grant access since Abe is NOT a member of the Approvers for the Seller access group.

Although Abe passed the command-level check, since he failed the resource-level access control check, he cannot update Emily's document.

Scenario 4:Guest 3 attempts to update his own document: The following is the access control evaluation for this scenario:

Command - level check:

1. There is no store ID specified, so the owner of the command is set to Root organization. So, only policies owned by Root Organization will be used to evaluate whether the user has command-level access: policies 1 and 2 are owned by Root organization.
2. Policy 1 does NOT grant access, since guest 3 is NOT a member of the `Registered Users` access group.

Resource - level check:

1. Resource-level checking is NOT even done since the Command-level check failed

Since Guest 3 failed the command-level check, he cannot update his own document.

Evaluating template policies

This example is based on the previous scenario.

Access control policies related to updating documents

When evaluating template policies, access control policies 1 and 2 used for evaluating standard policies still apply, however, standard policies 3 and 4 are now replaced by template policy 5. For more information on policies 1 and 2 see, "Evaluating standard policies" on page 26.

Policy 5:

```
[Approvers for Organization, Update Document Action Group, document, - ]
```

This policy is a template resource-level policy. Approvers for the organization that owns the document, can update documents.

We also need a new parameterized access group to be used by this template policy. The following access group is added to this scenario:

- Approvers for Organization: This group implicitly includes all users that have the role of approver for ? organization. (the ? parameter will be dynamically changed to the policy owner, as the template policy is applied at runtime).

Scenarios

The following scenarios use policies 1, 2, and 5 only.

Scenario 1: Don attempts to update Carol's document: The following is the access control evaluation for this scenario:

Command - level check:

1. There is no store ID specified, so the owner of the command is set to Root organization. So, only policies owned by Root Organization will be used to evaluate whether the user has command-level access: policies 1 and 2 are owned by Root organization. During policy evaluation, template policies dynamically change ownership to the organization that owns the resource, and subsequently that organization's ancestors, so policy 5 will also apply.
2. Policy 1 grants access, since Don is a member of the Registered Users access group and he is performing the `Execute` action on the `Update Document` command resource.

Resource - level check:

1. The `Update Document` command specifies that the document resource is to be protected. Carol's document is owned by Division A. So, only policies owned by Division A and its ancestor organizations will apply: policies 1, 2. During policy evaluation, template policies dynamically change ownership to the organization that owns the resource, and subsequently that organization's ancestors, so policy 5 will also apply.
2. Template policy 5 is first applied to the organization that owns the resource: Division A. At this moment policy 5 essentially behaves like policy 5a:
   ```
   [Approvers for Division A, Update Document Action Group, document, - ] standard
   resource-level policy owned by Division A.
   ```
3. Policy 5a does NOT grant access, since Don is NOT a member of the Approvers for Division A access group.
4. Template policy 5 will next be applied to the parent organization of Division A: Seller organization. At this moment policy 5 essentially behaves like policy 5b:
   ```
   [Approvers for Seller, Update Document Action Group, document, - ] standard
   resource-level policy owned by Seller
   ```

5. Policy 5b does grant access since Don is a member of Approvers for Seller access group, and he is performing the `Update Document` command action on the document resource.

Since Don passed both the command-level and resource-level access control checks, he can update Carol's document.

Scenario 2: Abe attempts to update Emily's document: The following is the access control evaluation for this scenario:

Command - level check:

1. There is no store ID specified, so the owner of the command is set to Root organization. So, only policies owned by Root Organization will be used to evaluate whether the user has command-level access: policies 1 and 2 are owned by Root organization. During policy evaluation, template policies dynamically change ownership to the organization that owns the resource, and subsequently that organization's ancestors, so policy 5 will also apply.
2. Policy 1 grants access, since Abe is a member of the Registered Users access group and he is performing the `Execute` action on the `Update Document` command resource.

Resource - level check:

1. The`Update Document` command specifies that the document resource is to be protected. Emily's document is owned by Seller organization. So, only policies owned by Seller and its ancestor organizations will apply: policies 1, 2. During policy evaluation, template policies dynamically change ownership to the organization that owns the resource, and subsequently that organization's ancestors, so policy 5 will also apply.
2. Template policy 5 is first applied to the organization that owns the resource: Seller organization. At this moment policy 5 essentially behaves like policy 5a:
   ```
   [Approvers for Seller, Update Document Action Group, document, - ] standard
   resource-level policy owned by Seller
   ```
3. Policy 5a does NOT grant access, since Abe is NOT a member of the Approvers for Seller access group.
4. Template policy 5 will next be applied to the parent organization of Seller organization: Root organization. At this moment policy 5 essentially behaves like policy 5b:
   ```
   [Approvers for Root, Update Document Action Group, document, - ] standard
    resource-level policy owned by Root
   ```
5. Policy 5b does NOT grant access since Abe is NOT a member of Approvers for Root access group.
6. Root organization does not have a parent organization, so template policy 5 has been completely evaluated.

Although Abe passed the command-level check, since he failed the resource-level access control check, he cannot update Emily's document.

Looking at a policy in detail

Now that we understand the basic structure of an access control policy and the types of policies there are, let us look at one of the default policies in detail, using a series of different examples. The policy we will study is the following:

```
AuctionAdministratorsForOrgExecuteAuctionManageCommandsOnAuctionResource
```

Note: This policy is a resource level policy. Its policy type is template.

In the first example, we will learn how to read the policy using the WebSphere Commerce Administration Console, identify its parts, and understand what the policy means. The second example will look at the policy in XML, to help you understand what the same information looks like in the code.

The third example goes a step further in understanding how one policy is related to other policies. Understanding dependencies between policies is an important prerequisite for making changes to access control policies, or creating new ones.

Example 1: Reading a policy

In this example, we will use the WebSphere Commerce Administration Console to look up a policy and identify the parts that define it. We will also use these parts to form a general description of the policy.

Looking up the policy in the Administration Console

1. Log in to the WebSphere Commerce Administration Console. From the Access Management menu, select **Policies**.
2. Verify that the View drop-down menu is set to your organization.
3. On the Policies page, scroll through the list of policies and locate the following policy: `AuctionAdministratorsForOrgExecuteAuctionManageCommandsOnAuctionResource`

   Notice that you can scroll through the list of policies by using the scroll bar as well as using the **First**, **Previous**, **Next** and **Last** links.

Viewing the parts of the policy

1. Select the policy by clicking the box next to it and click **Show Action Group**.
2. On the Action Group page, you will see the action group, AuctionManage. This is the action group associated with the policy. Select `AuctionManage` and click **Show Actions**.
3. On the next page, you will see the following list of actions, or commands, included in the `AuctionManage` action group:
   - `com.ibm.commerce.negotiation.commands.CloseBiddingCmd`
   - `com.ibm.commerce.negotiation.commands.DeleteAuctionCmd`
   - `com.ibm.commerce.negotiation.commands.ModifyAuctionCmd`

   Here, `AuctionManage` includes closing an auction (`CloseBiddingCmd`), deleting an auction, (`DeleteAuctionCmd`), and modifying an auction (`ModifyAuctionCmd`). For more information on commands, refer to the reference section in the online help documentation.

   Notice that you can also access the same list of actions from the Policies page by clicking **Show Actions**.
4. To return to the policies page, select any of the actions, and click **Show Policies**.
5. Select the policy again, but now click **Show Member Group** to see the member (access group) for which this policy applies.
6. Make a note of the member (access) group name. In this case, the member (access) group is `AuctionAdministratorsForOrg`.
7. From the Access Management menu, select **Access Groups**.
8. Find `AuctionAdministratorsForOrg`. Select it and click **Change.**

9. Click **Criteria**. On the Criteria page, look under Selected roles and organizations. You should see the following roles:
   - `Seller-For organization`
   - `Product Manager-For organization`
   - `Buyer (sell-side)-For organization`
   - `Category Manager-For organization`

   Any user assigned one of these roles for the organization that owns the auction resource, is part of the `AuctionAdministratorsForOrg` access group.
10. Leave the Criteria page without making any changes. From the Access Management menu, select **Policies** again. Locate the following policy:

    `AuctionAdministratorsForOrgExecuteAuctionManageCommandsOnAuctionResource`
11. Select the policy and Click **Show Resources**. On the Resources page, you will see the `com.ibm.commerce.negotiation.objects.Auction` resource. This is the resource on which the actions listed in the action group act. In this case, the resource is an auction. Notice that you can access this same list from the Policies page by clicking **Show Resource Group** and drilling down to the individual resources.
12. Now select **Policies** from the Access Management menu, and locate the following policy:

    `AuctionAdministratorsForOrgExecuteAuctionManageCommandsOnAuctionResource`
13. Select the policy and click **Change**. On the Change Policy page, look at the drop-down menu under **Relationship**. Note that the relationship is set to none. This means that the policy does not have a relationship.
14. Click **Cancel** and **OK** to the dialog box.

Understanding what the policy means

Now that we have identified the individual parts of this policy, we can start to piece them together to understand what the policy does. First, we know that the policy applies to all users belonging to the `AuctionAdministratorsForOrg` group. We learned this by clicking **Show Member Group**. From there we used the Access Management menu to go to the Access Group page, and saw that the access group included the following roles: `seller`, `product manager`, `buyer (for the sell-side)`, and the `category manager`. Collectively, users with one of these four roles can be referred to as an Auction Administrator.

We also know that the action group contains the commands for modifying, retracting, and closing an auction, and that the resource group includes only the auction resource that is being managed. Again, we know this by clicking **Show Actions** and **Show Resources** from the Policies page and drilling to the detail level. Lastly, we can tell that the policy does not include a relationship between the access group and the resources.

Putting everything together, we can conclude that this policy permits Auction Administrators to perform all the activities associated with managing auctions on an auction resource, such as modifying, retracting, and closing an auction, as long as the administrator plays the role for the organization that owns the auction.

We can get a sense of what a policy means by looking at its name. In this example, the policy starts with the name of the designated group of users, `AuctionAdministrator`. `ForOrg` indicates that the policy is applied to organizations. `AuctionManageCommands` describes the action group, and `AuctionResource` describes the resource group.

Example 2: Reading a policy in XML

The default access control policies are stored in an XML file that is loaded into your database during instance creation. When you look at a policy in the WebSphere Commerce Administration Console, you are using the interface to view and make changes to the information stored in the database. The information in the database is used by the Policy Manager to evaluate access control. If the database information is more recent than the XML file, you can use the Extractor tool to extract the access control policy information from the database into an XML file.

Most of the time you will use the WebSphere Commerce Administration Console user interface to manage policies. However, if you want to see a policy in XML, or if you want to make an advanced modification, this is what a policy looks like in the XML file:

```
<!-- AuctionAdministrators
manage Auctions (Retract/delete auction,
Modify auction, Close Auction)
-->
<Policy
Name="AuctionAdministratorsForOrgExecuteAuctionManageCommandsOnAuctionResource"
OwnerID="RootOrganization"
UserGroup="AuctionAdministratorsForOrg"
ActionGroupName="AuctionManage"
ResourceGroupName="AuctionDataResourceGroup"
PolicyType="template">
</Policy>
```

Here, the policy is defined by the following:

`Name`: The name of the policy.

`OwnerID`: The organization for which the policy applies.

`UserGroup`: The access group.

`ActionGroupName`: The action group.

`ResourceGroupName`: The resource group.

`PolicyType`: The type of policy, such as site-level, template, or organization.

The file that contains all of the default access control policies is called `defaultAccessControlPolicies.xml` and is located in the following directory:

`X:\`*`installation_directory`*`\xml\policies\xml`.

Note: The descriptions for each default access control file are contained in the `defaultAccessControlPolicies_`*`locale`*`.xml` file, which can be found in the same directory. A change made to a default access control policy in the default access control file, needs to have its corresponding description updated in `defaultAccessControlPolicies_en_US.xml`. It is strongly recommended that any changes made to the XML files be reserved for advanced users.

Example 3: Identifying other policies associated with your policy

In this last example, we will take a look at how an access control policy can have dependencies on other policies.

Policies that define the commands (actions) that a group of users (an access group) can perform on a resource are called resource-level policies. For example, the policy we have been looking at in detail:

`AuctionAdministratorsForOrgExecuteAuctionManageCommandsOnAuctionResource` is an example of a resource-level policy.

However, the actions permitted by a resource-level policy are also dependent on the actions permitted for each role belonging to the policy's access group. Policies that describe what actions are permitted for a particular role are called role-based policies.

To identify the role-based policies associated with a resource-level policy, do the following:

Looking up the roles associated with the policy

1. Log in to the WebSphere Commerce Administration Console and locate the resource-level policy on the Policies page. Using the same example, we know the policy we want is the following:
   `AuctionAdministratorsForOrgExecuteAuctionManageCommandsOnAuctionResource`

   .
2. Identify the access group associated with the policy. In this case, we already know that the access group is `AuctionAdministratorsForOrg`.
3. Look up the roles associated with the access group. For `AuctionAdministratorsForOrg`, we know from the previous examples that the roles are: `Buyers (sell-side)`, `Category Managers`, `Product Manager`, and `Sellers`.

Looking up the role-based policies for each role

1. Turn to the Appendix at the end of this book and find the section heading, Role-Based Policies. You will use the Appendix to locate each role-based policy associated with a role.
2. Find the `Buyers(sell-side)ExecuteBuyers(sell-side)CommandsResourceGroup` policy. This policy is associated with the `Buyers (sell-side)` role. We know this because of the `Buyers(sell-side)` prefix to the policy.
3. Find the rest of the role-based policies associated with `Buyers (sell-side)`, `Category Manager`, `Product Manager`, and `Sellers` roles, using their prefixes to identify the right policies. You should come up with the following list:
   - `Buyers(sell-side)ExecuteBuyers(sell-side)CommandsResourceGroup`
   - `Buyers(sell-side)ExecuteBuyers(sell-side)Views`
   - `CategoryManagersExecuteCategoryManagersCmdResourceGroup`
   - `CategoryManagersExecuteCategoryManagersViews`
   - `ProductManagersExecuteProductManagersCmdResourceGroup`
   - `ProductManagersExecuteProductManagersViews`
   - `SellersExecuteSellersCmdResourceGroup`
   - `SellersExecuteSellersViews`

4. Each role-based policy permits users with that role to carry out particular controller commands or views. To see what action is associated with a role-based policy, look up the policy on the Policies page from the WebSphere Commerce Administration Console, using the same procedure from Example 1.

Why identifying dependencies between policies is important

Understanding which role-based policies are associated with a resource-level policy is often a prerequisite for customizing your policies, and for creating new ones.

In Chapter 5, “Customization scenarios” on page 45, you will learn more about resource-level and role-based policies, including how to recognize them, understand their differences, and see how they are related to each other.

Chapter 4. Customizing default access control policies

The default access control policies provided by WebSphere Commerce address the basic requirements that organizations have for regulating the actions and information available to their users. Often, the default policies may be sufficient for your site's needs. At the same time, the default policies are highly customizable, which enables you to tailor them to your own requirements.

The `SiteAdministratorsCanDoEverything` policy, is a special default policy that grants super-user access to administrators with the Site Administrator role. In this policy, a Site Administrator can perform any action on any resource, even if those actions or resources have not been defined. It is important to be aware of this when assigning this role to users.

This chapter provides information on how to make basic changes to the default access control policies included with WebSphere Commerce. We begin by introducing certain concepts and relationships you'll need to understand.

Note: If you encounter terms or concepts that are unfamiliar to you, see Chapter 3, "Access control concepts" on page 9 for more information.

Identifying the policies affected by a change

In the previous chapter, you learned that policies are often related to other policies. You also learned how to start with a resource-level policy and identify the role-based policies associated with it. In this section we will explain in more detail how policies are related to each other and why you need to understand their relationships before you can modify an existing policy, or create a new one. In many cases, you need to change several policies to properly implement a change.

Understanding the relationship between role-based and resource-level policies

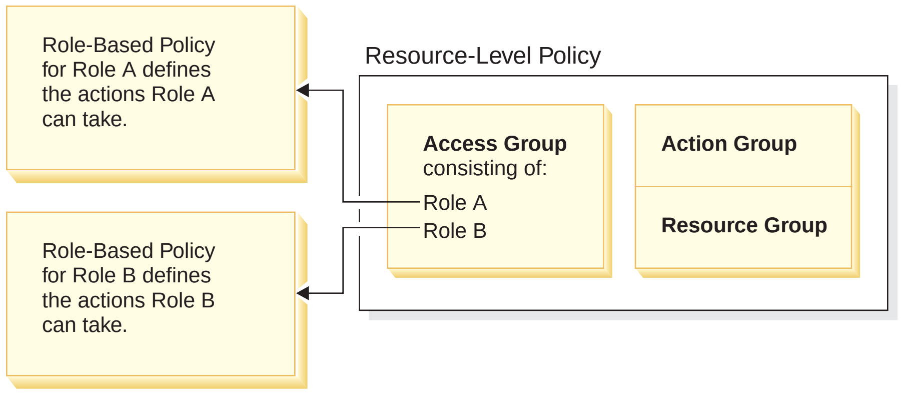
In WebSphere Commerce, each action that can be taken by a user is assigned to one or more roles using role-based policies as follows:

- Each default role has a corresponding access group. For instance, the access group for the role Store Administrator is `StoreAdministrators`.
- Each ″role-based″ access group generally has two associated role-based policies:
  - A policy that defines the controller commands the role is authorized to execute.
  - A policy that defines the view actions the role is authorized to execute. View actions map to views in the VIEWREG table. For instance, `StoreListView` displays a web page with the list of stores in the system.

Some controller commands have only a role-based policy, but no resource-level policy. This occurs if the command is not operating on any protectable resources. For example, the command `SetCurrencyPreferenceCmd` does not need a resource-level policy since it can only change the currency preference for the user running the command. If it was able to change the currency preference of another user, then the user object would have to be protected, and a resource-level policy would be needed. .

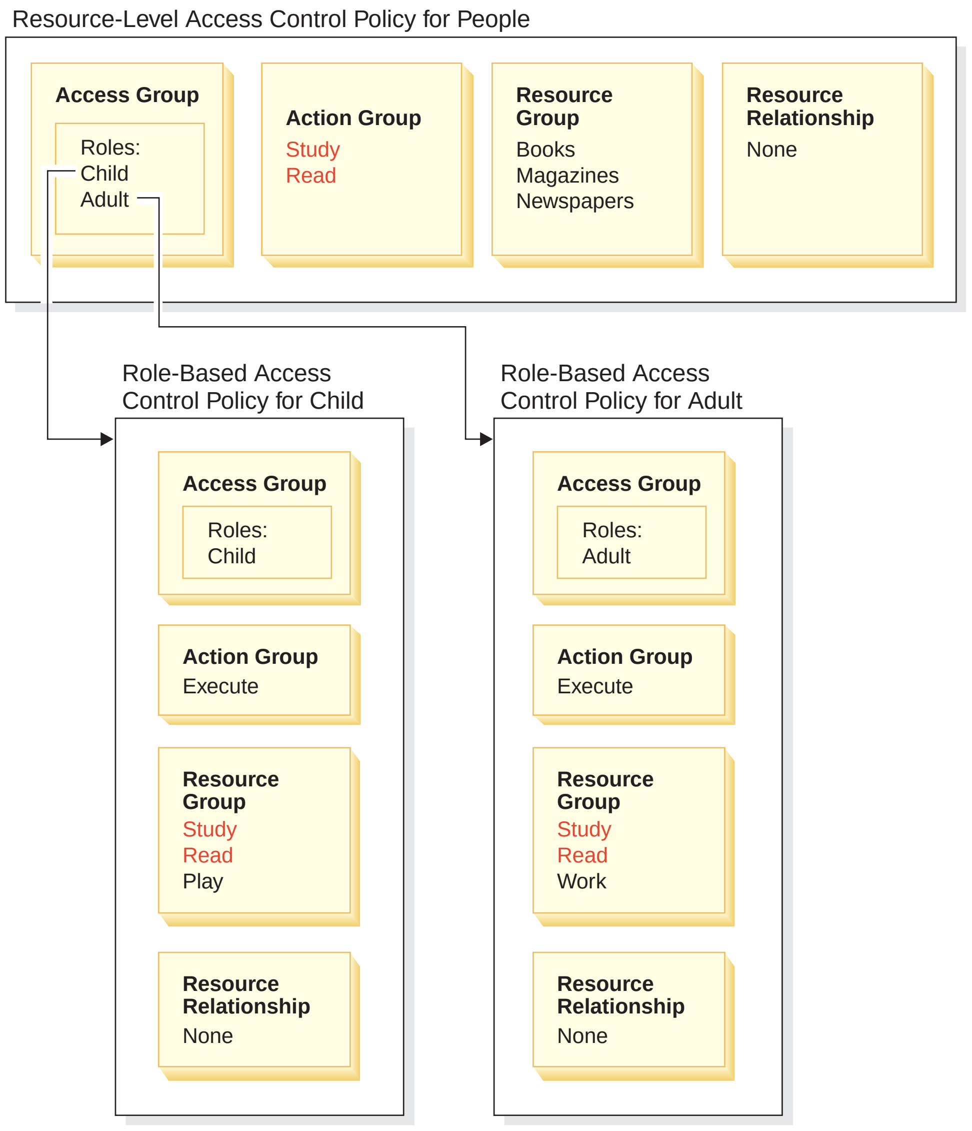
Resource-level policies for controller commands are directly related to certain role-based policies for controller commands. In the resource-level policy, the controller command is part of the action group, but in the role-based policy, the controller command is part of the resource group. The figure below illustrates this relationship. The resource-level policy includes Roles A and B in its access group, which brings the role-based policies for Roles A and B into play. While the resource-level policy grants authorization for users with roles A or B to take certain actions on a specific set of resources, the associated role-based policies provide authorization for users with roles A and B to take those actions in general.

Figure 3. Relationship between a resource-level policy and its associated role-based policies


Role-Based Policy for Role A defines the actions Role A can take.
Role-Based Policy for Role B defines the actions Role B can take.
Resource-Level Policy
Access Group consisting of:
Role A
Role B
Action Group
Resource Group

The following figure shows a sample resource-level policy that authorizes users in the `People` access group to read or study certain resources - namely books, magazines, and newspapers. This policy is correctly formulated because the role-based policies for the roles, `child` and `adult` also authorize them to read or study books, magazines, and newspapers.

Figure 4. A resource-level policy and the role-based policies that affect it.



Notice that in role-based policies for controller commands:

- The action group contains only a single action: `Execute`.
- The resource group contains the controller command that can be executed.

Similarly, in role-based policies for views:

- The action group contains the views that can be executed.
- The resource group contains a single resource: `com.ibm.commerce.command.ViewCommand`.

On the other hand, in resource-level policies:

- The action group contains the set of actions that can be performed on the resources in the resource group.
- The resource group contains a list of the actual business resources that can be acted upon.

A resource-level policy can only authorize users in a particular role to take actions already authorized by the corresponding role-based policy. For instance, in the above example, the role `child` is authorized to take the following actions:

- Study
- Read
- Play

Suppose that the resource-level policy is now changed to include a new action called `work`. Users with the role `adult` will be able to perform the action `work`. However, users with the role `child` will not. The reason for this is apparent when you check the role-based policies for the two roles. The policy for `adult` lists the action `work` in its resource group. The policy for `child` does not. Even though both `child` and `adult` are properly authorized by the resource-level policy, the role-based policy for `child` does not authorize the action `work`.

Because of the way resource-level policies are tied to role-based policies, the best way to track down all the policies affected by a particular change is to work backwards from the resource-level policy. The first step is to examine the access group of the resource-level policy and determine if it is contains any roles. You can view the complete list of default roles by selecting Access Management > Roles from the Administration Console.

If the resource-level policy's access group includes roles, review their role-based policies to see whether they need to be changed. If you are adding an action to the action group of a resource-level policy, you need to make sure that the relevant role-based policies also authorize the new action. If you are deleting an action from a resource-level policy, and no other resource-level policies reference this action, it's best to remove the corresponding resource from the associated role-based policies.

Understanding the policy model

An authorizing policy must be present for a user to perform an action. However, WebSphere Commerce permits users to take an action if **any** policy provides the needed authorization. Therefore, if you define a new, more restrictive policy than the default, you must delete or modify the broader default policy to prevent it from overriding your new policy.

For instance, suppose default policy A authorizes all registered users to submit auction bids. You want to change this policy so that auction bidding is limited to users with the buyer role. If you merely define a new policy that authorizes buyers to create auction bids, then your new policy will have no effect. Default policy A will still permit all registered users to bid. To make your new policy take effect, you must delete the broader, default policy.

Table 1 summarizes the additional changes you need to make when you create, delete, or change a resource-level policy.

Table 1. Additional changes needed when you change a resource-level policy that uses roles.

When you make this change to a resource-level policy:	You must also make the following change if the resource-level's access group uses roles:
Add an action to the policy's action group.	Ensure that the applicable role-based policies include the action in their resource groups.

Table 1. Additional changes needed when you change a resource-level policy that uses roles. (continued)

Remove an action from the policy's action group.	No additional change required. For consistency, it is better to remove this action from the corresponding resource groups in the related role-based policies. This should only be done if no other action groups are referring to this action. If other action groups are referring to this action, likely there are role-based policies that still need to have this action in their resource group.
Use a different action group.	Ensure that the applicable role-based policies include in their resource groups the new action group's actions.
Add a role to the policy's access group.	Ensure that the role-based policy corresponding to the new role, refers to a resource group that includes the actions specified in the resource-level policy.
Remove a role from the policy's access group.	No additional change required. For consistency, it is better to modify the corresponding role-based policy so that it no longer references these actions in its resource group.
Use a different access group.	Ensure that the applicable role-based policies include in their resource groups the actions in the resource-level policy's action group.
Create a new policy.	Check whether there is an existing policy that authorizes the same actions. Delete if necessary.
Delete the policy.	To prevent any users from taking that policy's actions, delete any other policies that authorize the same actions.

Determining whether a policy is role-based or resource-level

Role-based policies are also known as command-level policies because they authorize users with a particular role to execute a set of commands. Resource-level policies authorize a group of users to execute a set of commands on a particular set of resources. For instance, a role-based policy might authorize children to eat. While a resource-level policy might authorize children eat rice.

You can usually determine whether a policy is a role-based policy or a resource-level policy by looking at its name.

Role-based policies

Policies that define the controller commands that a role can execute follow the naming convention:

<AccessGroupforRoleXYZ> Execute <XYZCmdResourceGroup>

For instance: `ProductManagersExecuteProductManagersCmdResourceGroup`.

In role-based policies for controller commands, the action group contains a single entry called `Execute` and the resource group contains a list of WebSphere Commerce commands that users with that role can execute.

Policies that define the views that a role can execute follow the naming convention:

<AccessGroupforRoleXYZ> Execute <XYZViews>

For instance: `SalesManagersExecuteSalesManagerViews`.

In role-based policies for views, the action group contains a list of the views that users with that role can execute.

Resource-level policies

Policies that define who can take actions on data resources (business objects that can be created or manipulated) follow the naming convention:

<AccessGroupXYZ> Execute <XYZCommands> On <XYZResource>

For instance: `AllUsersExecuteOrderProcessOnOrderResource`.

In resource-level policies, the action group contains WebSphere Commerce commands and the resource group identifies the specific business resources that can be acted upon.

One exception is policies that authorize the creation of an entity such as an order, a bid, or an RFQ. These policies do not act on the entity itself because it has not yet been created. Instead, they act on the containing entity. For instance, an auction is created in the context of a store, a user is created in the context of an organization. Most resources are created in the context of a store. Consequently, these policies have names such as:

<AccessGroupXYZs> Execute <XYZCommands> On <StoreEntityResource>

For instance:
`AuctionAdministorsForOrgExecuteAuctionCreateCommandsOnStoreEntityResource`.

Policies that define who can view DataBean resources (Data beans contain information about data resources such as a bid or an order; usually used in JSPs) follow the naming convention:

<AccessGroupXYZs> Display <XYZDatabeanResourceGroup>

For instance: `MembershipViewersForOrgDisplayMembershipDatabeanResourceGroup`.

Tips for changing default policies

Keep the following in mind when you change your default policies:

- Most access groups are defined by user roles such as buyer or product manager. To better understand these roles and what actions they are permitted to take, see “Roles” on page 11.
- Before you change a policy to use a different access group, review the definition of that access group to ensure it meets your requirements. To do so, select **Access Management > Access Groups** from the Administration Console.
- Depending on the value you select for View, the Policies page displays either the site-level policies or the policies specific to a particular organization:
  - If you set the View field to `Root Organization`, you are shown the standard policies owned by the root organization, and the master versions of the template policies.
  - If you set the View field to the name of an organization, you are shown the standard policies owned by that organization and the template policies that can be modified by that organization
- Rename any default policies you change so that the policy name reflects what the policy does and so that you can identify the default policies you have

changed. Consider implementing a naming convention for your customized policies. If appropriate, you should also modify the description of the policy and its display name.

Note: The WebSphere Administration Console can only perform simple modifications to the access control policy definitions and access group definitions. The more robust solution is to update the data using XML files. The following operations can only be done through XML:

1. Defining new actions, resources, attributes, relationships, relationship groups.
2. Defining complex implicit resource groups, and complex implicit access groups.

After you make your policy changes

Each time you create or modify an access control policy, you must perform certain tests to verify that the policy is working correctly.

Once you have finished testing all your new and changed policies that are currently in the database, it is a good idea to extract that information into XML files. These files have the same format as the initial access control policy related files: `defaultAccessControlPolices.xml`, `defaultAccessControlPolicies_locale.xml`, and `ACUserGroup_locale.xml`. This step is necessary because changes made using the Administration Console affect only the policy information stored in the database. The XML files that were used to load the default access control policies and their components during instance creation, are not updated automatically.

You should maintain consistency between the XML files and the access control information in the database for several reasons:

- When you create an instance of WebSphere Commerce, the policy and access group definitions are loaded from the XML files.
- The XML files offer a convenient way to directly view and edit your policies and their component parts so keeping the files up-to-date is essential.

Testing your policy changes

For each policy, ensure the following:

- A user that belongs to the policy's access group is able to take the specified actions on the specified resources. If you have removed authorization to perform an action, you should also test to make sure that the user can no longer perform the action.
- A user that does not belong to the policy's access group is unable to take the specified actions on the specified resources.

For example, suppose you implement Auction customization scenario 1 in Chapter 5, in which you remove the ability of auction administrators to close auction bidding. To test whether this change is working properly, log in as a user who belongs to the `auction administrator` access group and perform the following actions:

- Modify an auction
- Delete an auction.

You should also verify that an Auction Administrator cannot close bidding.

Then, log in as a user who does not belong to the `auction administrator` access group and attempt to perform the same actions. If the policy is working correctly, your attempts should fail.

Extracting your policy changes into the XML files

When you have finalized and tested your policy changes, you should update the XML files to keep them in sync with policy information in the databases. The Appendix describes the different XML files related to access control policies, and access groups. It also explains how to extract policy changes from the databases into the XML files and how to load the policy information from the XML files into the databases.

Chapter 5. Customization scenarios

The customization scenarios presented below let you apply what you have learned about access control policies to make a variety of basic changes to your default policies. For all of these scenarios, it is assumed that a Site Administrator is modifying the policies for Root Organization. Once you step through some of the scenarios, you will be able to follow the same methodology to make changes not specifically covered here.

The scenarios are organized by business area. Within each business area, the scenarios are presented in order of increased complexity.

Table 2. Table of contents for scenarios

Business area	**Starting on page**
Auctions	"Auctions scenario 1: Removing the ability of auction administrators to close auction bidding" on page 46
Contracts	"Contracts scenario 1: Remove the ability of contract administrators to add or delete attachments to contracts" on page 50
Orders	"Orders scenario 1: Permitting only buyers to create orders" on page 52
Membership	"Membership scenario 1: Remove the ability of users to self-register" on page 58
Coupons	"Coupons scenario 1: Allowing only buyers to redeem coupons" on page 62
Procurement	"Procurement scenario 1: Allowing procurement shopping cart managers to manage the procurement shopping cart for orders created by their organization" on page 66
Inventory	"Inventory scenario 1: Permit fulfillment center managers to update fulfillment centers but not to delete them" on page 69
Business intelligence	"Business intelligence scenario 1: Allowing auditors to view business intelligence reports" on page 70

If you are looking for a scenario that illustrates a particular kind of change, refer to Table below, which cross-references the scenarios by the type of customization illustrated.

Table 3. Customization scenarios organized by type of customization

Customization	**See page**
Adding a role to a policy's access group	64
Changing a policy's action group	67,69
Changing a policy's resource relationship	54,66
Changing a policy to use a different access group	49,52,54,59,62,64

Table 3. Customization scenarios organized by type of customization (continued)

Creating a new access group and using it in a policy	56,60
Creating a new action group and using it in a policy	60,67
Creating a new resource-level policy	51,67
Creating a new role-based policy	60,70
Creating a new role and using it in a resource-level policy	60,70
Deleting a policy	47,48,58
Removing an action from a policy's action group	3,50

Table 3: Customization scenarios organized by type of customization

Auctions scenario 1: Removing the ability of auction administrators to close auction bidding

By default, auction administrators for a store can modify or delete auctions for the store, as well as close bidding. In certain cases, you may not want to grant auction administrators the authority to close bidding, either because you want this action handled by others or because you do not require this action for your store.

In this scenario, you will remove the authority of auction administrators to close bidding. To accomplish this change, you will do the following:

1. Use the Appendix to find the resource-level policy that defines the actions that auction administrators can take.
2. Determine the name of the action group for the policy.
3. Delete the action for closing auction bidding from the policy's action group.

Steps to take

Identify the policy whose action group must be changed

1. Look under Auctions, in the Appendix , to identify the resource-level policy to be changed. The policy is:
   `AuctionAdministratorsForOrgExecuteAuctionManageCommandsOnAuctionResource`
2. From the Administration Console, click **Access Management > Policies**.
3. For View, select **Root Organization** to display the site-level policies.
4. Locate the policy in the list.
5. Note the name of the policy's action group— `AuctionManage`. This is the action group you need to change to remove the action for closing bidding.

Remove the action for closing bidding from the policy's action group

1. Click **Access Management > Action Group**.
2. From the list of action groups, select **AuctionManage**.
3. Click **Change** to display the Change Resource Group page.
4. From the Selected Actions list, select **com.ibm.commerce.negotiation.commands.CloseBiddingCmd**.

5. Click **Remove**.
6. Click **OK**.

Update the policy registry with your changes

1. Click **Configuration > Registry**.
2. From the list of registries, select **Access Control Policies** .
3. Click **Update**.

Auctions scenario 2: Removing the ability of auction administrators to retract bids

By default, auction administrators for a store can retract bids submitted at their auctions. In some cases, you might not want to grant this authority to anyone. To make this change, you must find the resource-level policy that defines who can retract bids and delete it.

In Auctions Scenario 1, the action, close bidding, was one of several included in the policy. Consequently, you had only to remove the action from the policy's action group. In this scenario, however, an entire policy controls bid retraction. Therefore, you must delete a policy not just an action.

To delete the policy, you need to do the following:

- Use the Appendix to find the resource-level policy that covers the retraction of auction bids by auction administrators.
- Delete the policy.

Note: Before you delete the policy, make note of its name, access group name, resource group name, and action group name so you can recreate it for the next scenario.

Steps to take

1. Look under Auctions in the Appendix, to identify the resource-level policy to be changed. The policy is:
   `AuctionAdministratorsForOrgExecuteAdminRetractBidCommandsOnAuctionResource`
2. From the Administration Console, click **Access Management > Policies**.
3. For View, select **Root Organization** to display the site-level policies.
4. From the list of policies, select the following:
   `AuctionAdministratorsForOrgExecuteAdminRetractBidCommandsOnAuctionResource`
5. Click **Delete**.

Update the policy registry with your changes

1. Click **Configuration > Registry**.
2. From the list of registries, select **Access Control Policies**.
3. Click **Update**.

Auctions scenario 3 : Removing the ability of auction administrators to retract bids in one organization.

By default, auction administrators for a store can retract bids submitted at their auctions. In some cases, as a site administrator, you might want to change this policy for a particular organization. To make this change, you must delete the template policy that authorizes this action for this organization.

Note: In WebSphere Commerce Professional Edition, there are only three organizations, Root Organization, Default Organization and Seller Organization.

After you delete the policy, that organization's auction administrator will no longer be able to retract bids. The auction administrators for the other organizations will be unaffected by the change.

To delete the policy, you need to do the following:

- Use the Appendix to find the resource-level policy that authorizes the retraction of auction bids.
- Locate the policy in the list of policies for the organization.
- Delete the policy.

Steps to take

Delete the policy

1. Look under Auctions, in the Appendix, to identify the resource-level policy to be changed. The policy is:

   `AuctionAdministratorsForOrgExecuteAdminRetractBidCommandsOnAuctionResource`
2. From the Administration Console, click **Access Management > Policies**.
3. For View, select the organization whose policy you want to delete. When you select a particular organization, rather than `Root Organization`, your policy changes apply only to that organization rather than to all organizations in the site.
4. From the list of policies, select the following:

   `AuctionAdministratorsForOrgExecuteAdminRetractBidCommandsOnAuctionResource`
5. Click **Delete**.

Update the access control policy registry with your changes

1. Click **Configuration > Registry**.
2. From the list of registries, select **Access Control Policies**.
3. Click **Update**.

Auctions scenario 4: Limiting auction bidding to buyers

By default, all registered users are permitted to bid for products being auctioned at a store, regardless of their position in their organization. In some cases, you may want to limit bidding to a restricted group of users such as those assigned the `buyer` role in WebSphere Commerce.

In this scenario, you will change a resource-level policy, as well as its associated role-based policy. To limit bidding to members of a buying organization with the `buyer` role, you need to do the following:

- Use the Appendix to find the resource-level policy that specifies who can create an auction bid.
- Change the policy's access group from all registered users, to those with the `buyer` role.
- Rename the policy, description, and display name.
- Identify the command for creating bids.
- Use the Appendix to find the role-based policy for buyers (buy-side). This policy defines the commands that users with the Buyer (buy-side) role can execute. You must update this policy's resource group to permit buyers to execute the command for creating bids.
- Update this role-based policy's resource group to include the command for creating bids.

Steps to take

Identify the resource-level policy

1. Look under Auctions, in the Appendix, to identify the resource-level policy to be changed. The policy is: `RegisteredApprovedUsersExecuteBidCreateCommandsOnAuctionResource`.
2. From the Administration Console, click **Access Management > Policies**.
3. For View, select **Root Organization** to display the site-level policies.
4. From the list of policies, select **RegisteredApprovedUsersExecuteBidCreateCommandsOnAuctionResource**.
5. Note the name of the policy's action group— `BidCreate`. This is the action group you need to view to find the name of the command for creating a bid.

Change the access group for the policy

1. Click **Change** to display the Change Policy page.
2. For User Group, click **Find** and select **Buyers (buy-side)**.
3. Click **OK**.
4. Rename the policy, display name, and description of the policy, by editing their text.
5. Click **OK**.

Identify the command for creating bids

1. Click **Access Management > Action Groups**.
2. From the list of action groups, select **BidCreate**.
3. Click **Change** to display the Change Action Group page. Note the name of the command for creating bids: `com.ibm.commerce.negotiation.commands.BidSubmitCmd`. You must add this command to the resource group that contains the list of commands a buyer can execute.

Identify the role-based policy and resource group for the buyers (buy-side) role

1. Look under Role-Based Policies in the Appendix to find the role-based policy for buyers (buy-side). The policy is:

   `Buyers(buy-side)ExecuteBuyers(buyside)CommandsResourceGroup`.
2. Click **Access Management** > **Policies**.
3. For View, select **Root Organization** to display the site-level policies.

4. Note the name of the resource group: `Buyers(buy-side)CommandsResourceGroup`. Now you have the name of the resource group you need to update.

Update the resource group in the role-based policy to include the command for creating bids

1. Click **Access Management > Resource Groups**.
2. Select **Buyers(buy-side)CommandsResourceGroup**.
3. Click **Change** to display the Change Resource Group page.
4. Click **Next** to display the Details page.
5. From the Available Resources list, select **com.ibm.commerce.negotiation.commands.BidSubmitCmd**. This is the command for creating bids.
6. Click **Add** to add the command to the resource group.
7. Click **Finish**.

Update the access control policy registry with your changes

1. Click **Configuration >Registry**.
2. From the list of registries, select **Access Control Policies**.
3. Click **Update**.

Contracts scenario 1: Remove the ability of contract administrators to add or delete attachments to contracts

By default, contract administrators for a store can add or delete attachments to contracts they manage. In some cases, you might not want to grant this authority to contract administrators.

In this scenario, you will change a resource-level policy that defines the actions that a contract administrator can take. To remove the authority of contract administrators to add or delete attachments to contracts, you need to do the following:

- Use the Appendix to find the resource-level policy that defines the actions that contract administrators can take.
- Determine the name of the action group for the policy.
- Delete the actions for adding attachments and deleting attachments from the list of actions in the policy's action group.

Steps to take

Identify the resource-level policy and action group

1. Look under Contracts, in the Appendix, to identify the resource-level policy to be changed. The policy is:

   `ContractAdministratorsForOrgExecuteContractManageCommandsOnContractResource`
2. From the Administration Console, click **Access Management > Policies**.
3. For View, select **Root Organization** to display the site-level policies.
4. Locate the policy in the list.
5. Note the name of the policy's action group—`ContractManage`. This is the action group you need to change to remove the actions for adding and deleting attachments.

Remove the actions for adding and deleting attachments from the policy's action group

1. Click **Access Management** > **Action Group**.
2. From the list of action groups, select **ContractManage**.
3. Click **Change** to display the Change Resource Group page.
4. From the Selected Actions list, select the following actions: **com.ibm.commerce.contract.commands.ContractAttachmentAddCmd** **com.ibm.commerce.contract.commands.ContractAttachmentDeleteCmd**.
5. Click **Remove**.
6. Click **OK**.

Update the access control policy registry with your changes

1. Click **Configuration > Registry**.
2. From the list of registries, select **Access Control Policies**.
3. Click **Update**.

Contracts scenario 2: Permit both contract operators and contract administrators to deploy contracts

By default, contract operators for a store can deploy contracts. In some cases, you might want to grant this authority to contract administrators as well.

The flexible design of access control policies offers several methods for implementing this change:

- You can create a new access group containing both contract operators and contract administrators and assign the new access group to the policy that defines who can deploy contracts.
- You can add the `deploy contract` actions to the policy that specifies the actions a contract administrator can perform.
- You can create a new policy that permits contract administrators to deploy contracts.

This scenario illustrates the third approach. It shows you how to create a new resource-level policy that authorizes contract administrators to deploy contracts.

To create this policy, you need to do the following:

- Use the Appendix to find the resource-level policy that authorizes contract operators to deploy contracts.
- Note the name of the action group for this policy.
- Note the name of the resource group for this policy.
- Define a new policy for the `contract administrator` access group, specifying the action group and resource group from the policy that authorizes contract operators to deploy contracts.

Steps to take

Identify the action group and resource group to use in the new policy

1. Look under Contracts, in the Appendix, to find the resource-level policy that authorizes contract operators to deploy contracts The policy is: `ContractOperatorsForOrgExecuteContractDeployCommandsOnContractResource`.

2. From the Administration Console, click **Access Management > Policies**.
3. For View, select **Root Organization** to display the site-level policies.
4. Locate the policy in the list.
5. Note the name of the policy's action group—`ContractDeploy`. This is the action group you need to use in defining your new policy.
6. Note the name of the resource group—`ContractDataResourceGroup`, This is the resource group you need to use in defining your new policy.

Define the new policy

1. Click **New** to display the New Policy page.
2. For Name, specify:
   `ContractAdministratorsForOrgExecuteContractDeployCommandsOnContractResource`
3. For Display Name, specify a short description of the policy in your local language.
4. For Description, specify a longer description of what the policy does, in your local language.
5. For User Group, click **Find** and select **ContractAdministratorForOrg**.
6. Click **OK**.
7. For Resource Group, select **ContractDataResourceGroup**.
8. For Action Group, select **ContractDeploy**.
9. For Policy Type, select **Template Policy** to designate the policy as a template policy.
10. Click **OK**.

Update the access control policy registry with your changes

1. Click **Configuration > Registry**.
2. From the list of registries, select **Access Control Policies**.
3. Click **Update**.

Orders scenario 1: Permitting only buyers to create orders

By default, all users are permitted to create orders for products, regardless of their position in their organization. In some cases, you may want to limit the ability to create orders to a restricted group of users, such as the employees of the buying organization. Typically, these employees are assigned the Buyer (buy-side) role for the buying organization.

To limit order creation to members of a buying organization with the `buyer` role, you need to do the following:

- Use the Appendix to find the resource-level policy that specifies who can create an order.
- Change the policy's access group from all users to those with the `buyer` role.
- Update the policy's name, display name, and description.
- Identify the command for creating orders.
- Use the Appendix to find the role-based policy for buyers (buy-side). This policy defines the commands that users with the Buyer(buy-side) role can execute. You must update this policy's resource group to permit buyers to execute the command for creating orders.
- Update this role-based policy's resource group to include the commands for creating orders.

Note: This resource-level policy is a template policy. In this scenario, we have changed the master copy of this template at the Root Organization level. If you want to change it only for a particular organization, other than Root Organization, you have to change the View to the other organization before changing the policy. This results in the template policy being overridden for this organization alone. Then a new standard policy is created for this organization, which has the more restricted access group of Buyer (buy-side) users. Since the less restrictive template policy still applies at the Root Organization level, it must be overridden at that level as well. Currently, the only way to do this is by updating the ACORGPOL table in the database manually, and by running the following SQL:

```
insert into ACORGPOL (acpolicy_id, member_id) values ( (select acpolicy_id
 from ACPOLICY where policyname = ' AllUsersExecuteOrderCreateCommands
OnStoreResource'), -2001)
```

Steps to take

Identify the resource-level policy

1. Look under Orders, in the Appendix, to identify the resource-level policy to be changed. The policy is: `AllUsersExecuteOrderCreateCommandsOnStoreResource`.
2. From the Administration Console, click **Access Management > Policies**.
3. For View, select **Root Organization** to display the site-level policies.
4. From the list of policies, select **AllUsersExecuteOrderCreateCommandsOnStoreResource.** Note the name of the policy's action group—`OrderCreateCommands`. This is the action group you need to view to find the names of the commands for creating an order.

Change the access group

1. Click **Change** to display the Change Policy page.
2. For User Group, click **Find** and select **Buyers (buy-side)**.
3. Click **OK**.
4. Update the policy's name, display name, and description to reflect the change of access group.
5. Click **OK**.

Identify the command for creating orders

1. Click **Access Management > Action Groups**.
2. From the list of action groups, select **OrderCreateCommands** .
3. Click **Change** to display the Change Action Group page. Note the names of the commands for creating orders:

   ```
   com.ibm.commerce.order.commands.OrderCopyCmd
    com.ibm.commerce.order.commands.OrderScheduleCmd
    com.ibm.commerce.orderitems.commands.OrderItemMoveCmd
    com.ibm.commerce.orderitems.commands.OrderItemUpdateCmd
    com.ibm.commerce.requisitionlist.commands.RequisitionListSubmitCmd
   ```

   You must add these commands to the resource group that contains the list of commands a buyer can execute.

   Note: The command, `com.ibm.commerce.orderitems.commands.AdminOrderItemUpdateCmd`, is not needed.

Identify the role-based policy for buyers (buy-side)

1. Look under Role-Based Policies, in the Appendix, to find the role-based policy for buyers (buy-side). The policy is: `Buyers(buyside)ExecuteBuyers(buyside)CommandsResourceGroup`.
2. Click **Access Management** > **Policies**.
3. For View, select **Root Organization** to display the site-level policies.
4. Locate the policy in the list.
5. Note the name of the resource group—`Buyers(buyside)CommandsResourceGroup`. This is the resource group you need to update.

Update the resource group in the role-based policy to include the commands for creating orders

1. Click **Access Management > Resource Groups**.
2. From the list of resource groups, select **Buyers(buyside)CommandsResourceGroup**.
3. Click **Change** to display the Change Resource Group page.
4. Click **Next** to display the Details page.
5. From the Available Resources list, select the following commands for creating orders:

```
com.ibm.commerce.order.commands.OrderCopyCmd

com.ibm.commerce.order.commands.OrderScheduleCmd
 com.ibm.commerce.orderitems.commands.OrderItemMoveCmd
 com.ibm.commerce.orderitems.commands.OrderItemUpdateCmd
 com.ibm.commerce.requisitionlist.commands.RequisitionListSubmitCmd
```

6. Click **Add.**
7. Click **Finish**.

Update the access control policy registry with your changes

1. Click **Configuration > Registry**.
2. From the list of registries, select **Access Control Policies**.
3. Click **Update**.

Orders scenario 2: Allowing only Buyer Administrators to modify orders

Note: This scenario does not apply to WebSphere Commerce Professional Edition.

By default, all users are permitted to modify orders they have created, regardless of their position in their organization. In some cases, you may want only the organization's buyer administrator to have the authority to modify orders.

In this scenario, you will change a resource-level policy, as well as a role-based policy. To allow only buyer administrators to modify orders belonging to members of a buyer organization, you need to do the following:

- Use the Appendix to find the resource-level policy that specifies who can modify an order.
- Change the policy's access group from all users, to those with the `buyer administrator` role.
- Remove the specification of the resource relationship to permit buyer administrators to modify orders belonging to other users.

- Update the policy’s name, display name, and description.
- Identify the commands for modifying orders.
- Use the Appendix to find the role-based policy for buyer administrator. This policy defines the commands that users with the buyer administrator role can execute. You must update this policy’s resource group to permit buyer administrators to execute the commands for modifying orders.
- Update the role-based policy’s resource group to include the commands for modifying orders.

Steps to take

Identify the resource-level policy

1. Look under Orders, in the Appendix, to identify the resource-level policy to be changed. The policy is: `AllUsersExecuteOrderWriteCommandsOnOrderResource`.
2. From the Administration Console, click **Access Management > Policies**.
3. For View, select **Root Organization** to display the site-level policies.
4. From the list of policies, select **AllUsersExecuteOrderWriteCommandsOnOrderResource**.
5. Note the name of the policy’s action group—`OrderWriteCommands`. You need to view this action group to find the name of the command for creating an order.

Change the access group

1. Click **Change** to display the Change Policy page.
2. For User Group, click **Find** and select **Buyer Administrators**.
3. Click **OK**.
4. Update the policy’s name, display name, and description to reflect the change of access group.
5. Click **OK**.

Identify the commands for modifying orders

1. Click **Access Management > Action Groups**.
2. From the list of action groups, select **OrderWriteCommands** .
3. Click **Change** to display the Change Action Group page. Make note of the names of the commands for modifying orders:

```
com.ibm.commerce.order.commands.OrderCancelCmd
 com.ibm.commerce.order.commands.OrderCopyCmd-Write
 com.ibm.commerce.order.commands.OrderUnlockCmd
 com.ibm.commerce.orderitems.commands.OrderItemAddCmd
 com.ibm.commerce.orderitems.commands.OrderItemDeleteCmd
 com.ibm.commerce.orderitems.commands.OrderItemMoveCmd
 com.ibm.commerce.orderitems.commands.OrderItemUpdate.Cmd
```

   You must add these commands to the resource group that contains the list of commands a buyer can execute.

   Notes:

   a. The command, `com.ibm.commerce.orderitems.commands.AdminOrderItemUpdateCmd`, is not needed.

   b. When you add the command, `com.ibm.commerce.order.commands.OrderCopyCmd-Write`, to the resource group, it appears under Available Resources as `com.ibm.commerce.order.commands.OrderCopyCmd.`

Identify the role-based policy for the buyer administrator role

1. Look under Role-Based Policies in the Appendix to find the role-based policy for buyer administrators. The policy is: `BuyerAdministratorsExecuteBuyersAdministratorsCommands`.
2. Click **Access Management** > **Policies**.
3. For View, select **Root Organization** to display the site-level policies.
4. Locate the policy in the list.
5. Make note of the name of the resource group— `BuyersAdministratorsCommmandsResourceGroup`.

   This is the name of the resource group you need to update.

Update the resource group in the role-based policy to include the commands for modifying orders

1. Click **Access Management > Resource Groups**.
2. Select **BuyersAdministratorsCommandsResourceGroup**.
3. Click **Change** to display the Change Resource Group page.
4. Click **Next** to display the Details page.
5. From the Available Resources list, select the commands for modifying orders:
   ```
   com.ibm.commerce.order.commands.OrderCancelCmd
    com.ibm.commerce.order.commands.OrderCopyCmd
    com.ibm.commerce.order.commands.OrderUnlockCmd
    com.ibm.commerce.orderitems.commands.OrderItemAddCmd
    com.ibm.commerce.orderitems.commands.OrderItemDeleteCmd
    com.ibm.commerce.orderitems.commands.OrderItemMoveCmd
    com.ibm.commerce.orderitems.commands.OrderItemUpdate.Cmd
   ```
6. Click **Add** to add the command to the resource group.
7. Click **Finish**.

Update the access control policy registry with your changes

1. Click **Configuration > Registry**.
2. From the list of registries, select **Access Control Policies**.
3. Click **Update**.

Orders scenario 3: Allowing RMA approvers to approve all RMAs

By default, return merchandise authorization (RMA) approvers for a store are only permitted to approve RMAs for their own stores. In some cases, you may want to allow RMA approvers to approve RMAs for any store. This might be desirable if several stores are owned by the same organization or if the same person handles the RMA approvals for multiple stores.

In this scenario, you will create a new access group and use it in a new resource-level policy. To allow RMA approvers to approve RMAs against any store, you need to do the following:

- Use the Appendix to find the resource-level policy that permits RMA approvers for an organization to approve RMAs for their organization.
- Note the name of the resource group and action group used in the policy.
- View the policy’s access group, `RMAApproversForOrg`, and note the roles it includes. The access group is defined using both organizations and roles as selection criteria. To give users authority to perform an action across multiple organizations, the access group must be defined without organizational criteria.

- Create a new access group, RMAApprovers, that uses the same roles but does not include the organizational criteria.
- Create a new policy using:
  - The new access group, RMAApprovers
  - The action group from the existing policy
  - The resource group from the existing policy

Steps to take

Identify the action group and resource group to use in defining the new policy

1. Look under Orders, in the Appendix, to find the resource-level policy that authorizes RMAApproversForOrg to approve RMAs for their stores. The policy is: RMAApproversForOrgExecuteRMAApproveCommandsOnRMAResource
2. From the Administration Console, click **Access Management > Policies**.
3. For View, select **Root Organization** to display the site-level policies.
4. Locate the policy in the list.
5. Note the name of the policy's action group—RMAApproveCommands. This is the action group you will use in defining your new policy.
6. Note the name of the resource group—RMADataResourceGroup, This is the resource group you will use in defining your new policy.
7. Note the name of the access group—RMAApproversForOrg. View this access group to see the roles to include in your new access group.

Identify the roles to be used in the new access group

1. Click **Access Management > Access Groups**.
2. From the list of access groups, select **RMAApproversForOrg**.
3. Click **Change**.
4. Select **Criteria** to display the Criteria page.
5. Under Selected Roles and Organizations, note the roles used in the access group:
   - Customer Service Supervisor
   - Seller
   - Sales Manager
   - Operations Manager
6. Click **Cancel** to return to the list of access groups.

Define the new access group

1. Click **New** to display the Details page for the new access group.
2. For Name, specify RMAApprovers.
3. For Description, specify a description of the access group.
4. For Parent Organization, select Root Organization.
5. Click **Next** to display the Criteria page for the new access group.
6. Click **Criteria based on organizations and roles**.
7. From the list of roles, select the following roles:
   - **Customer Service Supervisor**
   - **Seller**
   - **Sales Manager**

- **Operations Manager**

8. Click **Finish**.

Define the new policy

1. Click **Access Management > Policies**.
2. Click **New** to display the New Policy page.
3. For Name, specify: `RMAApproversExecuteRMAApproveCommandsOnRMAResource`
4. For Display Name, specify a short description of the policy in your local language.
5. For Description, specify a longer description of what the policy does, in your local language.
6. For User Group, click **Find** and select **RMAApprovers**.
7. Click **OK**.
8. For Resource Group, select **RMADataResourceGroup**.
9. For Action Group, select **RMAApproveCommands**.
10. Click **OK**.

Update the access control policy registry with your changes

1. Click **Configuration > Registry**.
2. From the list of registries, select **Access Control Policies**.
3. Click **Update**.

Membership scenario 1: Remove the ability of users to self-register

By default users are permitted to self-register if they belong to a registered organization. Membership administrators are also authorized to register users that belong to their organization. For sites that require strictly controlled access, it might be necessary to remove the ability to self-register and require that users be registered by membership administrators.

Note: In WebSphere Commerce Professional Edition, there are only three organizations, Root Organization, Default Organization and Seller Organization.

In this scenario, you will remove the resource-level policy that permits users to self-register but leave in place a policy that permits membership administrators to register users in their organization.

To delete the resource-level policy that allows users to self-register, do the following:

- Use the Appendix to find the resource-level policy that allows users to self-register.
- Delete the policy.

Steps to take

Delete the policy

1. Look under Membership, in the Appendix, to find the resource-level policy that allows users to self-register. The policy is: `GuestsExecuteUserSelfRegistrationCommandsOnOrganizationResource`.
2. From the Administration Console, click **Access Management > Policies**.
3. For View, select **Root Organization** to display the site-level policies.

4. From the list of policies, select **GuestsExecuteUserSelfRegistrationCommandsOnOrganizationResource**
5. Click **Delete**.

Update the access control policy registry with your changes

1. Click **Configuration > Registry**.
2. From the list of registries, select **Access Control Policies**.
3. Click **Update**.

Membership scenario 2: Allowing only registered and approved users to change their address information

By default, users can modify their address information if their registration has been approved or is pending approval. In some cases, you might want only registered and approved users to manage their addresses.

In this scenario, you will change the access group for the resource-level policy that authorizes users to manage their address information, as follows:

- Use the Appendix to find the resource-level policy that allows users to manage their address information.
- Change the access group for the policy.

  Because the access group `RegisteredApprovedUsers` does not contain any roles, you do not need to update a role-based policy for this change.

Steps to take

Change the resource-level policy's access group

1. Look under Membership, in the Appendix, to find the resource-level policy that allows users to manage their address information. The policy is—`NonRejectedUsersExecuteAddressManageCommandsOnUserResource`.

   Note: Non-rejected users are users whose registration has not been rejected. Their registration has either been approved or is pending approval.
2. From the Administration Console, click **Access Management > Policies**.
3. For View, select **Root Organization** to display the site-level policies.
4. From the list of policies, select **NonRejectedUsersExecuteAddressManageCommandsOnUserResource.**
5. Click **Change** to display the Change Policy page.
6. For User Group, click **Find** and select **RegisteredApprovedUsers**.
7. Click **OK**.
8. Update the policy's name, display name, and description to reflect the change of access group.
9. Click **OK**.

Update the access control policy registry with your changes

1. Click **Configuration > Registry**.
2. From the list of registries, select **Access Control Policies**.
3. Click **Update**.

Membership scenario 3: Allowing member registrars to register users

By default, membership administrators for an organization are authorized to register member of their organization. The access group, `MemberAdministratorsForOrg`, includes several roles such as buyer administrator and seller administrator, which are authorized to perform a variety of administrative tasks. In some cases, you might want to create a separate role that is authorized only to register organization members:

Here is an overview of the steps involved:

- Create a new role, and for it, a new access group, a new resource group, and a new role-based policy.
- Modify an existing resource-level policy to use the new role.

In this scenario, you will do the following:

- Define a new role called `Member Registrar`.
- Define a new access group, called `MemberRegistrars`, which includes the member registrar role.
- Use the Appendix to find the resource-level policy that permits membership administrators to register members.
- Note the name of the action in its action group. You must create a new resource group with this action and use it in the role-based policy for the new role. Keep in mind that, in role-based policies for actions, the action group contains only a single action execute. The resource group contains the actions (commands) that can be executed.
- Define a new resource group, called `MemberRegistrationCommands`, which includes the command for registering members. You will use this resource group in the role-based policy for the member registrar role.
- Define a new role-based policy for member registrars, which uses the MemberRegistrars access group and the MemberRegistrationCommands resource group.
- Modify the resource-level policy that defines who can register members and change its access group from MembershipAdministrators to MemberRegistrars.

Steps to take

Define the new role

1. From the administration console, click **Access Management > Roles**.
2. On the Roles page, click **New**.
3. For Name, specify Member Registrar.
4. For Description, specify a description of the member registrar role in your local language.
5. Click **OK**.

Define a new access group containing the member registrar role

1. Click **Access Management > Access Groups**.
2. On the Access Groups page, click **New** to display the Details page for the new access group.
3. For Name, specify: MemberRegistrars.
4. For Parent Organization, select Root Organization.

5. For Description, specify a description of the access group in your local language.
6. Click **Next** to display the Criteria page for the new access group.
7. Click **Based on organizations and roles**.
8. From the Role list, select **Member Registrar**.
9. Click **For Organization** to specify that the role must be within the users' own organization.
10. Click **Finish**.

Identify the actions to use in the resource group for the member registrar role-based policy

1. Look under Membership, in the Appendix, to find the policy that permits membership administrators to register users. The policy is:

   ```
   MembershipAdministratorsForOrgExecuteUserAdminRegistration
   CommandsOnOrganizationResource
   ```
2. Click **Access Management > Policies**.
3. For View, select **Root Organization** to display the site-level policies.
4. Locate the policy in the list.
5. Note the name of the policy's action group—`UserAdminRegistration`. This is the action group you need to view to identify the actions for registering members.
6. Click **Access Management > Action Groups**.
7. From the list of action groups, select **UserAdminRegistration**.
8. Click **Change** to display the Change Action Group page.
9. Note the name of the command for registering members: `com.ibm.commerce.usermanagement.commands.UserRegistrationAdminAddCmd`.

Define the new resource group to be used in the role-based policy for member registrars

1. Click **Access Management > Resource Groups** to display the Resource Groups page.
2. Click **New** to display the General page for the new resource group.
3. For Name, specify `UserAdminRegistrationCommands`.
4. For Display Name, specify a description of the resource group in your local language.
5. For Description, specify a longer description of the resource group, in your local language.
6. For Type, select **Explicit Resource Group**.
7. Click **Next**.
8. Click **Next** to display the Details page for the new resource group.
9. From the Available Resources list, select the following:

   `com.ibm.commerce.usermanagement.commands.`
   `UserRegistrationAdminAddCmd`
10. Click **Add**.
11. Click **Finish**.

Define a role-based policy for the member registrar role

1. Click **Access Management** > **Policies**.
2. On the Policies page, click **New**.
3. For Name, specify **MemberRegistrarsExecuteUserAdminRegistrationCommands**.

4. For Display Name, specify a description of the policy in your local language.
5. For Description, specify a longer description of what the policy does, in your local language.
6. For User Group, click **Find** and select **MemberRegistrars**.
7. Click **OK**.
8. For Resource Group, select **UserAdminRegistrationCommands**.
9. For Action Group, select **ExecuteCommandActionGroup**.
10. Click **OK**.

Modify the resource-level policy to use the new access group

1. From the list of policies, select the following:

   ```
   MembershipAdministratorsForOrgExecuteUserAdminRegistration
   CommandsOnOrganizationResource
   ```

   .
2. Click **Change** to display the Change Policy page.
3. Update the policy's name, display name and description to reflect the change of access group.
4. For User Group, click **Find** and select **MemberRegistrars**.
5. Click **OK**.

Update the access control policy registry with your changes

1. Click **Configuration > Registry**.
2. From the list of registries, select **Access Control Policies**.
3. Click **Update**.

Coupons scenario 1: Allowing only buyers to redeem coupons

By default, all registered users are permitted to redeem coupons. In some cases, you may want to limit coupon redemption to users with the `buyer` role in WebSphere Commerce.

In this scenario, you will change a resource-level policy, as well as its associated role-based policy. To limit coupon redemption to users with the `buyer` role, you need to do the following:

- Use the Appendix to find the resource-level policy that specifies who can redeem a coupon.
- Change the policy's access group from all registered users, to those with the `buyer` role.
- Identify the command for redeeming coupons.
- Use the Appendix to find the role-based policy for buyers (buy-side). This policy defines the commands that users with the buyer(buy-side) role can execute. You must update this policy's resource group to permit buyers to execute the command for redeeming coupons.
- Update this role-based policy's resource group to include the commands for redeeming coupons.

Steps to take

Identify the resource-level policy and its action group

1. Look under Coupons, in the Appendix, to identify the resource-level policy to be changed. The policy is:

   `RegisteredApprovedUsersExecuteCouponRedemptionCommandsOnCouponWalletResource`
2. From the Administration Console, click **Access Management > Policies**.
3. For View, select **Root Organization** to display the site-level policies.
4. From the list of policies, select the following:

   `RegisteredApprovedUsersExecuteCouponRedemption`
   `CommandsOnCouponWalletResource`
5. Note the name of the policy's action group— `CouponRedemption`. This is the action group you must view to find the name of the commands for redeeming coupons.

Change the access group

1. Click **Change** to display the Change Policy page.
2. For User Group, click **Find** and select **Buyers (buy-side)**.
3. Click **OK**.
4. Update the policy's name, display name, and description to reflect the change of access group.
5. Click **OK**.

Identify the commands for redeeming coupons

1. Click **Access Management > Action Groups**.
2. From the list of action groups, select **CouponRedemption**.
3. Click **Change** to display the Change Action Group page. Note the name of the commands for creating bids:

   ```
    com.ibm.commerce.couponredemption.commands.CouponDSSCmd
     com.ibm.commerce.couponredemption.commands.UseCouponIdCmd
   ```

   You must add these commands to the resource group that contains the list of commands a buyer can execute.

Identify the role-based policy for buyers (buy-side)

1. Look under Role-Based Policies in the Appendix to find the role-based policy for buyers (buy-side). The policy is:

   `Buyers(buy-side)ExecuteBuyers(buyside)CommandsResourceGroup`
2. Click **Access Management** > **Policies**.
3. For View, select **Root Organization** to display the site-level policies.
4. Locate the policy in the list.
5. Note the name of the resource group: `Buyers(buyside)CommandsResourceGroup`. This is the name of the resource group you need to update.

Update the resource group in the role-based policy to include the command for creating bids

1. Click **Access Management > Resource Groups**.
2. Select **Buyers(buy-side)CommandsResourceGroup**.
3. Click **Change** to display the Change Resource Group page.
4. Click **Next** to display the Details page.

5. From the Available Resources list, select **com.ibm.commerce.couponredemption.commands.CouponDSSCmd** **com.ibm.commerce.couponredemption.commands.UseCouponIdCmd**. These are the commands for redeeming coupons.
6. Click **Add** to add the commands to the resource group.
7. Click **Finish**.

Update the access control policy registry with your changes

1. Click **Configuration > Registry**.
2. From the list of registries, select **Access Control Policies**.
3. Click **Update**.

Coupons scenario 2: Permitting both coupon administrators and Store Administrators to create e-coupon promotions

By default, coupon administrators for a store can create e-coupon promotions for their store. In some cases, you might want to grant this authority to store administrators as well.

The flexible design of access control policies offers several methods for implementing this change:

- You can add the store administrator role to the access group for the policy that specifies who can create e-coupon promotions.
- You can create a new policy that permits store administrators to create e-coupon promotions.

This scenario illustrates the first approach. It shows you how to add the store administrator role to the resource-level policy that authorizes coupon administrators to create coupons.

To make this change, you need to do the following:

- Use the Appendix to find the resource-level policy that specifies who can create e-coupon promotions.
- Change the policy’s access group to include users with the `store administrator` role.
- View the resource-level policy’s action group to identify the command for creating e-coupon promotions.
- Use the Appendix to find the role-based policy for `store administrators`. This policy defines the commands that users with the `store administrator` role can execute. You must update this policy’s resource group to permit store administrators to execute the commands for creating e-coupon promotions.
- Update this role-based policy’s resource group to include the command for creating e-coupon promotions.

Steps to take

Identify the action group and access group for the resource-level policy

1. Look under Auctions, in the Appendix, to identify the resource-level policy to be changed. The policy is:

   `CouponAdministratorsForOrgExecuteCouponPromotionCreateCommands`
   `OnStoreEntityResource`

2. From the Administration Console, click **Access Management > Policies**.
3. For View, select **Root Organization** to display the site-level policies.
4. Locate the policy in the list.
5. Note the name of the policy's action group—`CouponPromotionCreate`. This is the action group you must view to find the name of the command for creating e-coupon promotions.
6. Note the name of the policy's access group—`CouponAdministratorsForOrg`. This is the access group you must update to include the store administrator role.

Change the access group

1. Click **Access Management > Access Groups**.
2. From the list of access groups, select **CouponAdministratorsForOrg**
3. Click **Change** to display the Details page.
4. Click **Criteria** to display the Criteria page.
5. From the Role list, select **Store Administrator**.
6. Click **For Organization** to specify that the role must be within the users' own organization.
7. Click **Add**.
8. Click **OK**.

Identify the commands for creating e-coupon promotions

1. Click **Access Management > Action Groups**.
2. From the list of action groups, select **CouponPromotionCreate**.
3. Click **Change** to display the Change Action Group page. Note the name of the command for creating e-coupon promotions—`com.ibm.commerce.tools.ecoupon.ECouponPromotionSaveCmd`. You must add this command to the resource group that contains the list of commands a store administrator can execute.

Identify the role-based policy for store administrators

1. Look under Role-Based Policies in the Appendix to find the role-based policy for store administrators. The policy is: `StoreAdministratorsExecuteStoreAdministratorsCmdResourceGroup.`
2. Click **Access Management** > **Policies**.
3. For View, select **Root Organization** to display the site-level policies.
4. Locate the policy in the list.
5. Note the name of its resource group—`StoreAdministratorsCmdResourceGroup`. This is the name of the resource group you need to update.

Update the resource group in the role-based policy to include the command for creating e-coupon promotions

1. Click **Access Management > Resource Groups**.
2. Select **StoreAdministratorsCmdResourceGroup**.
3. Click **Change** to display the Change Resource Group page.
4. Click **Next** to display the Details page.
5. From the Available Resources list, select `com.ibm.commerce.tools.ecoupon.ECouponPromotionSaveCmd`. This is the command for creating e-coupon promotions.
6. Click **Add**.
7. Click **Finish**.

Update the access control policy registry with your changes

1. Click **Configuration > Registry**.
2. From the list of registries, select **Access Control Policies**.
3. Click **Update**.

Procurement scenario 1: Allowing procurement shopping cart managers to manage the procurement shopping cart for orders created by their organization

Note: This scenario does not apply to WebSphere Commerce Professional Edition.

By default, procurement shopping cart managers are authorized to manage the procurement shopping cart when they have created the order. In some cases, you might want to extend the authority of procurement shopping cart managers to let them manage the procurement cart for orders created by any member of their organization.

To make this change, you need to do the following:

- Use the Appendix to find the resource-level policy that authorizes procurement shopping cart administrators to manage procurement shopping carts.
- Change the resource relationship for this policy from `creator` to `same organizational entity as creator`.

Steps to take

Change the resource relationship for the resource-level policy

1. Look under Procurement, in the Appendix, to find the resource-level policy that authorizes procurement shopping cart managers to manage procurement shopping carts for orders. The policy is:

   ```
   ProcurementShoppingCartManagersExecuteProcurementShopping
   CartManageOnOrderResource
   ```
2. From the Administration Console, click **Access Management > Policies**.
3. For View, select **Root Organization** to display the site-level policies.
4. From the list of policies, select the following:

   **ProcurementShoppingCartManagersExecuteProcurementShopping
   CartManageOnOrderResource**
5. Click **Change** to display the Change Policy page.
6. For Relationship, select **sameOrganizationalEntityAsCreator**.
7. Click **OK**.

Update the access control policy registry with your changes

1. Click **Configuration > Registry**.
2. From the list of registries, select **Access Control Policies**.
3. Click **Update**.

Procurement scenario 2: Allow procurement buyer administrators to submit the procurement shopping cart for orders created by their organization

Note: This scenario does not apply to WebSphere Commerce Professional Edition.

By default, procurement shopping cart managers can save or submit procurement shopping carts if they have created the order. In some cases, you might want to divide the responsibility for these tasks. You could allow procurement shopping cart managers to save procurement shopping carts containing orders they have created but give procurement buyer administrators in the same organization as the order creator the authority to submit the procurement shopping cart. This might be beneficial if you want the procurement buyer administrator to review planned purchases before they are submitted.

To make this change, you need to do the following:

- Use the Appendix to find the resource-level policy that authorizes procurement shopping cart managers to center managers to manage fulfillment centers.
- Remove the action for submitting a procurement shopping cart from the policy's action group.
- Define a new action group containing the command for submitting a procurement shopping cart. You will use this action group to define the new resource-level policy that authorizes procurement buyer administrators to submit procurement shopping carts if they are in the same organization as the creator of the order.
- Create a new resource-level policy that authorizes procurement buyer administrators to submit procurement shopping carts if they are in the same organization as the creator of the order.

Steps to take

Identify the resource-level policy's action group and resource group

1. Look under Procurement, in the Appendix, to find the resource-level policy that authorizes procurement shopping cart managers to manage procurement shopping carts for orders. The policy is:
   ```
   ProcurementShoppingCartManagersExecuteProcurement
   ShoppingCartManageOnOrderResource
   ```
2. From the Administration Console, click **Access Management > Policies**.
3. Locate the policy in the list of policies.
4. Note the name of its action group — `ProcurementShoppingCartManage`. You will update this action group to remove the action for submitting procurement shopping carts.
5. Note the name of its resource group — `OrderDataResourceGroup`. You will use this resource group to define the new resource-level policy .

Update the resource-level policy's action group

1. Click **Access Management > Action Groups**.
2. From the list of action groups, select **ProcurementShoppingCartManage**.
3. Click **Change** to display the Change Action Group page.

4. From the Selected Actions list, select **com.ibm.commerce.me.commands.SubmitShoppingCartCmd**. You will create a new action group with this action and use the action group in your new resource-level policy.
5. Click **Remove**.
6. Click **OK**.

Define a new action group

1. Click **Access Management > Action Groups**.
2. Click **New** to display the New Action Group page.
3. For Name, specify `ProcurementShoppingCartSubmit`.
4. For Display Name, specify a short description of the action group in your local language.
5. For Description, specify a longer description of what the action group does, in your local language.
6. From the Available Actions list, select **com.ibm.commerce.me.commands.SubmitShoppingCartCmd**.
7. Click **Add**.
8. Click **OK**.

Define the new policy

1. Click **Access Management > Policies**.
2. For View, click **Root Organization** to display the site-level policies.
3. Click **New** to display the New Policy page.
4. For Name, specify:
   ```
   ProcurementBuyerAdministratorsExecuteProcurementShoppingCartSubmitCommands
   OnOrderResource
   ```
5. For Display Name, specify a short description of the policy in your local language.
6. For Description, specify a longer description of what the policy does, in your local language.
7. For User Group, click **Find** and select **ProcurementBuyerAdministrators**.
8. Click **OK**.
9. For Resource Group, select **OrderDataResourceGroup**.
10. For Action Group, select **ProcurementShoppingCartSubmit**.
11. For Relationship, select **sameOrganizationalEntityAsCreator**.
12. For Policy Type, select **Template Policy** to designate the policy as a template policy.
13. Click **OK**.

Update the access control policy registry with your change

1. Click **Configuration > Registry**.
2. From the list of registries, select **Access Control Policies**.
3. Click **Update**.

Inventory scenario 1: Permit fulfillment center managers to update fulfillment centers but not to delete them

By default, fulfillment center managers are authorized to update or delete the fulfillment centers associated with their store. In some cases, you might want to allow fulfillment center managers to update fulfillment centers but not to delete them.

To make this change, you need to do the following:
- Use the Appendix to find the resource-level policy that authorizes fulfillment center managers to manage fulfillment centers.
- Remove the action for deleting a fulfillment center from the policy's action group.

Steps to take

Remove the action for deleting a fulfillment center

1. Look under Procurement, in the Appendix, to find the resource-level policy that authorizes procurement shopping cart managers to manage procurement shopping carts for orders. The policy is:
   ```
   FulfillmentCenterManagersForOrgExecuteFulfillmentCenter
   ManageCommandsOnFulfillmentResource
   ```
2. From the Administration Console, click **Access Management > Policies**.
3. Locate the policy in the list of policies.
4. Note the name of its action group—`FulfillmentCenterManage`. You need to update this action group to remove the action for deleting fulfillment centers.
5. Click **Access Management > Action Groups.**
6. From the list of action groups, select **FulfillmentCenterManage**.
7. Click **Change** to display the Change Action Group page.
8. From the Selected Actions list, select **com.ibm.commerce.inventory.commands.FulfillmentCenterDeleteCmd**.
9. Click **Remove**.
10. Click **OK**.

Update the access control policy registry with your changes

1. Click **Configuration > Registry**.
2. From the list of registries, select **Access Control Policies**.
3. Click **Update**.

Inventory scenario 2: Permit only logistics managers and operations managers to create, update, or delete fulfillment centers

By default, fulfillment center managers are authorized to create, update or delete the fulfillment centers associated with their store. The fulfillment center access group includes the roles: seller, logistics manager, and operations manager. In some cases, you might not want sellers to be authorized as fulfillment center managers.

To make this change, you need to do the following:
- Use the Appendix to find the resource-level policy that authorizes fulfillment center managers to manage fulfillment centers.

- Remove the seller role from the definition of the `fulfillment center managers` access group.

Steps to take

Remove the seller role from the access group

1. Look under Procurement, in the Appendix, to find the resource-level policy that authorizes procurement shopping cart managers to manage procurement shopping carts for orders. The policy is:

   ```
   FulfillmentCenterManagersForOrgExecuteFulfillmentCenterManage
   CommandsOnFulfillmentResource
   ```
2. From the Administration Console, click **Access Management > Access Groups**.
3. From the list of access groups, select **FulfillmentCenterManagersForOrg**.
4. Click **Change** to display the Change Access Group page.
5. Click **Access Management > Access Groups**.
6. Click **Change** to display the Details page.
7. Click **Criteria** to display the Criteria page.
8. From the Role list, select **Seller**.
9. Click **Remove**.
10. Click **OK**.

Update the access control policy registry with your changes

1. Click **Configuration > Registry**.
2. From the list of registries, select **Access Control Policies**.
3. Click **Update**.

Business intelligence scenario 1: Allowing auditors to view business intelligence reports

By default, intelligence report viewers are permitted to view business intelligence reports for their store. In some cases, you might also want to create a new role called `auditor` and authorize users with this role to view a store's business intelligence reports.

Here is an overview of the steps involved:

- Create a new role, and for it, a new access group, a new resource group, and a new role-based policy.
- Add the new role to the resource-level policy's access group.
- Define a new role called `Auditor`.
- Define a new access group, called `Auditors`, which includes the auditor role.
- Add the auditor role to the access group of the resource-level policy that defines who can view business intelligence reports for their stores.

In this scenario, you will do the following:

- Use the Appendix to find the resource-level policy that permits business intelligence report viewers to view business intelligence reports.
- Note the name of the action in its action group. You must create a new resource group with this action and use it in the role-based policy for the new role. Keep

in mind that, in role-based policies for actions, the action group contains only a single action execute. The resource group contains the actions (commands) that can be executed.

- Define a new resource group, called `AuditorCommands`, which includes the command for viewing business intelligence reports. You will use this resource group in the role-based policy for the auditor role.
- Define a new role-based policy for auditors, which uses the Auditors access group and the AuditorCommands resource group.
- Add the auditor role to the access group for the resource-level policy that defines who can view business intelligence reports for their store.

Steps to take

Define the new auditor role

1. From the administration console, click **Access Management > Roles**.
2. On the Roles page, click **New**.
3. For Name, specify Auditor.
4. For Description, specify a description of the auditor role in your local language.
5. Click **OK**.

Define a new access group for the auditor role

1. Click **Access Management > Access Groups**.
2. On the Access Groups page, click **New** to display the Details page for the new access group.
3. For Name, specify—Auditors.
4. For Description, specify a description of the access group in your local language.
5. For Parent Organization, select Root Organization.
6. Click **Next** to display the Criteria page for the new access group.
7. Click **Based on organizations and roles**.
8. From the Role list, select **Auditor**.
9. Click **Add**.
10. Click **Finish**.

Identify the actions to use in the resource group for the auditor role’s role-based policy

1. Look under Business Intelligence, in the Appendix, to find the policy that authorizes intelligence report viewers to view business intelligence reports. The policy is:
   ```
   IntelligenceReportViewersForOrgExecuteViewBusinessIntelligenceReport
   CommandsOnStoreEntityResource
   ```
2. From the Administration Console, click **Access Management > Policies**.
3. For View, select **Root Organization** to display the site-level policies.
4. Locate the policy in the list.
5. Note the name of the policy’s action group—`ViewBusinessIntelligenceReport`. This is the action group you must view to identify the actions for registering members.
6. Click **Access Management > Action Groups**.
7. From the list of action groups, select **ViewBusinessIntelligenceReport**.
8. Click **Change** to display the Change Action Group page.

9. Note the name of the command for viewing business intelligence reports—`com.ibm.commerce.bi.commands.BIShowReportCmd`.

Define the new resource group to be used in the role-based policy for the auditor role

1. Click **Access Management > Resource Groups** to display the Resource Groups page.
2. Click **New** to display the General page for the new resource group.
3. For Name, specify AuditorCommands.
4. For Display Name, specify a description of the resource group in your local language.
5. For Description, specify a longer description of the resource group, in your local language.
6. Click **Next**.
7. For Type, select **Explicit Resource Group**.
8. Click **Next** to display the Details page for the new resource group.
9. From the Available Resources list, select **com.ibm.commerce.bi.commands.BIShowReportCmd**.
10. Click **Add**.
11. Click **Finish**.

Define the role-based policy for the auditor role

1. Click **Access Management > Policies**.
2. On the Policies page, click **New**.
3. For Name, specify **AuditorsExecuteAuditorCommands**.
4. For Display Name, specify a description of the policy in your local language.
5. For Description, specify a longer description of what the policy does, in your local language.
6. For User Group, click **Find** and select **Auditors**.
7. Click **OK**.
8. For Resource Group, select **AuditorCommands**.
9. For Action Group, select **ExecuteCommandActionGroup**.
10. Click **OK**.

Add the auditor role to the resource-level policy's access group

1. Click **Access Management > Access Groups.**
2. From the list of access groups, select **IntelligenceReportViewersForOrg**.
3. Click **Change** to display the Change Access Group page.
4. Click **Criteria** to display the Criteria page for the access group.
5. From the Role list, select **Auditor**.
6. Click **For Organization** to specify that the role must be within the users' own organization.
7. Click **Add**.
8. Click **OK**.

Update the policy registry with your changes

1. Click **Configuration > Registry**.
2. From the list of registries, select **Access Control Policies**.
3. **Click Update.**

Chapter 6. Using XML files to customize access control polices

The WebSphere Commerce Administration Console allows you to make simple changes to access control policies and their parts. To make more sophisticated changes, you need to edit the XML files directly.

Before you begin making changes to the XML files for access control, you should read the chapter on access control in *IBM WebSphere Commerce Programmer's Guide*. This chapter provides a technical overview of access control and explains how to create customized commands, entity beans, and JSP templates that can be protected by access control policies.

Once you have finished the code customizations following the guidance provided in the *IBM WebSphere Commerce Programmer's Guide*, you can edit the XML files for access control to establish the protections you require.

Changes that can only be made by editing and loading the XML files

The following changes can only be made by editing and then loading the appropriate XML files:
- Protecting a new command or view
- Creating or modifying a relationship
- Creating or modifying a relationship group
- Protecting a new resource
- Creating or modifying attributes
- Creating or modifying access groups using complex criteria
- Creating or modifying resource groups using complex criteria

About the XML files for access control

The names and descriptions of WebSphere Commerce's XML files, DTD files, and XSL files for the XML Transformer, are shown in the following table.

Table 4. WebSphere Commerce XML files for access control

File Name	Description
ACUserGroups_de_DE.xmll ACUserGroups_en_US.xml ACUserGroups_es_ES.xml ACUserGroups_fr_FR.xml ACUserGroups_it_IT.xml ACUserGroups_ja_JP.xml ACUserGroups_ko_KR.xml ACUserGroups_pt_BR.xml ACUserGroups_zh_CN.xml ACUserGroups_zh_TW.xml	Access group definitions and descriptions in each supported language.
defaultAccessControlPolicies.xml	Main file containing the definitions of default access control policies, action groups, resource groups, relationships, relationship groups, actions, resource categories, and attributes.
defaultAccessControlPolicies_de_DE.xml defaultAccessControlPolicies_en_US.xml defaultAccessControlPolicies_es_ES.xml defaultAccessControlPolicies_fr_FR.xml defaultAccessControlPolicies_it_IT.xml defaultAccessControlPolicies_ja_JP.xml defaultAccessControlPolicies_ko_KR.xml defaultAccessControlPolicies_pt_BR.xml defaultAccessControlPolicies_zh_CN.xml defaultAccessControlPolicies_zh_TW.xml	Files containing the display names and descriptions for default access control policies, action groups, actions, resource groups, resource categories, relationships, and attributes, in each supported language.
ACPoliciesfilter.xml	Filter file used in the extraction of changed access control information from the databases.
accesscontrolpolicies.dtd	The access control policies XML file must conform to this DTD.
accesscontrolpoliciesnls.dtd	The access control policies NLS (national language specific) XML file (display names and descriptions only) must conform to this DTD.
ACUserGroups_en_US.dtd	The access control user groups XML file must conform to this DTD.
accesscontrol.xsl	The XSL transform rule file for the access control policies XML file.

Table 4. WebSphere Commerce XML files for access control (continued)

accesscontrolnls.xsl	The XSL transform rule file for the access control policies NLS XML file (display names and descriptions only).
ACUserGroup.xsl	The XSL transform rule file for the access group XML files.
wcstoacpolicies.xsl	The XSL transform rule file for the `ExtractedACPolicies.xml` file after extract, to create the access control policies XML file.
wcstoacpoliciesnls.xsl	The XSL transform rule file for the `ExtractedACPolicies.xml` after extract, to create the access control policies NLS XML file.
wcstoacusergroup.xsl	XSL transform rule file for the `ExtractedACPolicies.xml` file after extract, to create the access group XML file

Customizing the XML files

Protecting views

Any view that is called directly from an URL, or that is launched as a redirect from another command, needs a role-based access control policy in order to be displayed. The following example displays a role-based policy for views:

```
<Policy Name="ProductManagersExecuteProductManagersViews"
OwnerID="RootOrganization"
UserGroup="ProductMangers"
ActionGroupName="ProductMangersViews"
ResourceGroupName="ViewCommandResourceGroup">
</Policy>
```

The ResourceGroup name, `ViewCommandResourceGroup`, indicates that this is a role-based policy for views. The policy states that users in the `ProductManagers` user group, can display the views in the `ProductMangersViews` action group.

The following is an example of the `ProductMangersViews` action group:

```
<ActionGroup Name="ProductManagersViews"
 OwnerID="RootOrganization">

<ActionGroupAction Name="ProductImageView"/>
<ActionGroupAction Name="ProductManufacturerView"/>
<ActionGroupAction Name="ProductSalesTaxView"/>

 </ActionGoup>
```

The example above lists the three actions, `ProductImageView`, `ProductManufacturerView`, and, `ProductSalesTaxView` that can be performed in the `ProductManagerViews` action group.

The following is an example of the `ProductImageView` action definition:

```
<Action Name="ProductImageView"
CommandName="ProductImageView">
</Action>
```

The `Name` attribute, `ProductImageView,`is used as a tag for referencing the action elsewhere in the XML such as when associating the action with an action group.

Note: The name of the view, stored in the VIEWNAME column of the VIEWREG table, must match the `CommandName` in the action definition. The value of `CommandName` is stored in the ACTION column of the ACACTION table. The `Name` and `CommandName` attributes do not have to be the same.

Adding a new view using existing policies

To add a new view that is accessible by roles with existing role-based View policies, do the following:

1. Create a new action definition in the XML file that has the view name MyNewView.
```
<Action Name="MyNewView"
   CommandName="MyNewView">
</Action>
```
2. Determine which roles should have access to this view, and associate the new action with the corresponding action groups in the XML file:
```
<ActionGroup Name="ProductManagersViews"
    OwnerID="RootOrganization">

  <ActionGroupAction Name="ProductImageView"/>
  <ActionGroupAction Name="ProductManufacturerView"/>
  <ActionGroupAction Name="ProductSalesTaxView"/>
  <ActionGroupAction Name="MyNewView"/>

 </ActionGroup>
```
3. Load your XML changes into the database. For more information on loading the XML changes, see “Loading your changes into the database” on page 97.
4. Update the Access Control Policies Registry in the Administration Console.

Since there is already a role-base policy that includes this action group, the view can now be used.

Adding a new view using a new policy

To add a new view that is accessible by a new role that does not have an existing role-based policy, do the following:

1. Create a new action definition in the XML file that has the view name MyNewView.
```
<Action Name="MyNewView
CommandName="MyNewView">
</Action>
```
2. Create a new action group to be associated with the new role:
```
<ActionGroupName="XYZViews"
    OwnerID="RootOrganization">
</ActionGroup>
```
3. Associate the new action with the new action group:
```
< ActionGroupName="XYZViews"
     OwnerID="RootOrganization">

 <ActionGroupAction Name="MyNewView"/>

    </ActionGroup>
```
4. Create a policy that references the new action group:

```
<Policy Name="XYZExecuteXYZViews"
OwnerID="RootOrganization"
UserGroup="XYZ"
ActionGroupName="XYZViews"
ResourceGroupName="ViewCommandResourceGroup">
</Policy>
```

5. Load your XML changes into the database. For more information on loading the XML changes, see “Loading your changes into the database” on page 97.
6. Update the Access Control Policies Registry in the Administration Console.

You can now use your view.

Protecting controller commands

All controller commands require a role-based access control policy in order to be executed. A controller or task command also requires a resource-level policy if the command is doing resource-level checking. For more information see “Implementing resource-level access control” on page 79. The following example displays a role-based policy for controller commands:

```
< Policy Name="SellersExecuteSellersCmdResourceGroup"
OwnerID="RootOrganization"
UserGroup="Sellers"
ActionGroupName="ExecuteCommandActionGroup"
ResourceGroupName="SellersCmdResourceGroup">
</Policy>
```

The ActionGroupName, `ExecuteCommandActionGroup`, indicates that this is a role-based policy for controller commands. The policy states that users in the `Sellers` access group can execute the commands in the `SellersCmdResourceGroup`, resource group.

The following is an example of the `SellersCmdResourceGroupresource` group definition:

-
  ```
  <ResourceGroup Name="SellersCmdResourceGroup" OwnerID="RootOrganization">
  <ResourceGroupResource Name="com.ibm.contract.commands.Contract
  CancelCmdResourceCategory"/>
  <ResourceGroupResource Name="com.ibm.contract.commands.Contract
  CloseCmdResourceCategory"/>
  <ResourceGroupResource Name="com.ibm.contract.commands.Contract
  CreateCmdResourceCategory"/>
  </ResourceGroup>
  ```

The example above shows the following three resources in the resource group, that each correspond to a controller command:

- `com.ibm.contract.commands.ContractCancelCmdResourceCategory`
- `com.ibm.contract.commands.ContractCloseCmdResourceCategory`
- `com.ibm.contract.commands.ContractCreateCmdResourceCategory`

The following is a sample definition of a resource:

```
<ResourceCategory Name="com.ibm.commerce.contract.commands.Contract
CloseCmdResourceCategory"
ResourceBeanClass="com.ibm.commerce.contract.commands.ContractCloseCmd">

<ResourceAction Name="ExecuteCommand"/>

</ResourceCategory>
```

The `Name` attribute,
`com.ibm.commerce.contract.commands.ContractCloseCmdResourceCategory`, is used

as a tag to refer to the resource in the XML file. The ResourceAction Name, `ExecuteCommand`, is used to specify the actions that can operate on the resource. This information is used in the Administration console when using access control policies to populate the Action selection box that corresponds to a particular resource. In this case, the action `Execute` is specified. The `Execute` action is defined in the following:

```
<Action Name="ExcecuteCommand
CommandName="Execute">
</Action>
```

Note: The interface name of the controller command must match the `ResourceBeanClass` in the resource definition. The value of the `ResourceBeanClass` is stored in the RESCLASSNAME column of the ACRESCGRY table. These commands can be used as resources because they extend the ControllerCommand interface, which extends the AccCommand interface, which in turn extends the Protectable interface. For more information on these interfaces refer to the *IBM WebSphere Commerce Programmer's Guide*.

Adding a new controller command using existing policies

To add a new controller command to be accessible by roles that have existing role-based controller command policies, do the following:

1. Create a new resource definition in the XML file that corresponds to the interface name of the controller command.

```
<ResourceCategory  Name="com.xyz.commands.MyNewControllerCmdResourceCategory"
   ResourceBeanClass="com.xyz.commands.MyNewControllerCmd">

  <ResourceAction Name="ExecuteCommand"/>
 </ResourceCategory>
```

2. Determine which roles should have access to the command and associate the new resource with the corresponding resource groups in the XML file:

```
 <ResourceGroup Name="SellersCmdResourceGroup"  OwnerID="RootOrganization">
  <ResourceGroupResource Name="com.ibm.commerce.contract.
commands.ContractCancelCmdResourceCategory"/>
  <ResourceGroupResource Name="com.ibm.commerce.contract.
commands.ContractCloseCmdResourceCategory"/>
  <ResourceGroupResource Name="com.ibm.commerce.contract.
commands.ContractCreateCmdResourceCategory"/>

  <ResourceGroupResource Name="com.xyz.commands.
MyNewControllerCmdResourceCategory"/>

 </ResourceGroup>
```

3. Load your XML changes into the database. For more information on loading the XML changes, see "Loading your changes into the database" on page 97.
4. Update the Access Control Policies Registry in the Administration Console.

Since there is already a role-based policy that includes this resource group, you can now use your new controller command, if it is not doing any resource-level checking.

Adding a new controller command using a new policy

To add a new controller command to be accessed by a new role, that does not have an existing role-based policy, do the following:

1. Create a new resource definition in the XML file that corresponds with the interface name of the controller command. See "Adding a new controller command using existing policies" step one, for an example.

2. Create a new resource group to be associated with the new role:

```
<ResourceGroup Name="XYZCmdResourceGroup" OwnerID="RootOrganization">
</ResourceGroup>
```

3. Associate the new resource with the new resource group:

```
<ResourceGroup Name="XYZCmdResourceGroup" OwnerID="RootOrganization">
<ResourceGroupResource Name="com.xyz.commands.MyNewControllerResourceCategory"/>
</ResourceGroup>
```

4. Create a policy that references your new resource group:

```
<Policy Name="XYZExecuteXYZsCmdResourceGroup"
   OwnerID="RootOrganization"
   UserGroup="XYZ"
   ActionGroupName="ExecuteCommandActionGroup"
   ResourceGroupName="XYZCmdResourceGroup">
</Policy>
```

5. Load your XML changes into the database. For more information on loading the XML changes, see "Loading your changes into the database" on page 97.
6. Update the Access Control Policies Registry in the Administration Console.

You can now use your controller command if it is not doing any resource-level checking.

Implementing resource-level access control

You can add resource level access control to controller or task commands. Resource-level checking is done at WebSphere Commerce runtime, based on data returned by the `getResources()` method of a command. Resource-level checking can also be done during the `performExcecute()` portion of the command by making direct calls to the access control policy manager using the method `void checkIsAllowed(Object resource, String action) throws ECException`. This method will throw the `ECApplicationException` if the current user is not allowed to perform the specified action on the specified resource.

Note: By default, the `getResources()` method returns null, and no resource—level checking is done.

You need to create a resource-level policy for new commands in the following instances:

- The new command extends from another command that is doing a resource-level check.
- The new command itself does resource-level access control checking.

The following is an example of a resource-level policy:

```
<Policy Name="ContractMangersForOrgExecuteContractManageCommandsOnContractResource"
  OwnerID="RootOrganization"
  UserGroup="ContractManagersForOrg"
  ActionGroupName="ContractManage"
  ResourceGroupName="ContractDataResourceGroup"
  PolicyType="template">
</Policy>
```

Where:

`Name`: The name of the policy.

`PolicyType`: The policy type. This is a template policy and will dynamically apply to the organizational entity that owns the resource and it's ancestors.

`OwnerID`: The member that owns the policy. This is a template policy and will dynamically change to be the resource-owning organizational entity and it's ancestors, as the policy is applied by the Access Control Policy Manager.

`UserGroup`: The policy applies to users of this group. The naming convention for access groups where roles are dynamically scoped to the resource organizational entity and it's ancestor's is to append `ForOrg` to the group name.

`ActionGroupName`: The name of the action group that contains the actions to be performed on the resource.

`ResourceGroupName`: The name of the resource group that contains the resources to be acted upon.

In the example above, the action group `ContractManage` is an action group that contains the set of commands that will act on the `ContractDataResourceGroup`. The following is an example of the action group that is used in the above resource-level policy:

```
<ActionGroupName="ContractManage" OwnerID="RootOrganization">
<ActionGroupActionName="com.ibm.commerce.contract.commands.ContractCancelCmd"/>
<ActionGroupActionName="com.ibm.commerce.contract.commands.ContractCloseCmd"/>
<ActionGroupActionName="com.ibm.commerce.contract.commands.ContractDeleteCmd"/>
</ActionGroup>
```

The commands that were previously defined as resources for role-based policies are now defined as actions. The following is a sample definition of an action that is a part of the above `ContractManage` group:

```
<Action Name="com.ibm.commerce.contract.commands.ContractCloseCmd"
CommandName="com.ibm.commerce.contract.commands.ContractCloseCmd">
</Action>
```

Note: The value of `CommandName` should correspond to the interface name of the command that is doing the resource-level check.

Most commands work with enterprise beans. These beans are usually the resources that the resource-level policies are protecting. The following is a sample definition of the resource group that is used in the above resource policy:

```
<ResourceGroup Name="ContractDataResourceGroup" OwnerId="RootOrganization">
<ResourceGroupResource Name="com.ibm.commerce.contract.
objects.ContractResourceCategory"/>
</ResourceGroup>
```

In this example, `ContractDataResourceGroup` is defined and is composed of one resource. The resource is defined as follows:

```
<ResourceCategory Name="com.ibm.commerce.contract.objects.ContractResourceCategory"
ResourceBeanClass="com.ibm.commerce.contract.objects.Contract"
 <ResourceAction Name="com.ibm.commerce.contract.commands.ContractCancelCmd"/>
 <ResourceAction Name="com.ibm.commerce.contract.commands.ContractCloseCmd"/>
 <ResourceAction Name="com.ibm.commerce.contract.commands.ContractDeleteCmd"/>
 </ResourceCategory>
```

Where:

`Name`: A tag used to reference this resource elsewhere in the XML file.

`ResourceBeanClass`: The class representing the resource to protect. This class must implement the Protectable interface. If the resource is an enterprise bean, it's remote interface should extend the Protectable interface.

`ResourceAction`: Specifies the actions that will be operating on this resource. This information is used by the Administration Console when determining which actions are valid with a particular resource.

Note: For more information on the Protectable interface, refer to the *WebSphere Commerce Programmer's Guide*.

Protecting data beans

Data beans contain information about business objects and are used to display object information on a web page. Dynamic web pages are usually mapped to views within WebSphere Commerce, and these views are protected by role-based policies. It is sometimes necessary to further protect the content of the web page by protecting it's data beans, if they exist.

When data beans are populated using the `DataBeanManager.activate(..)` method, the data bean managers enforce access control on them. Data beans can be protected directly or indirectly, using the Delegator interface. Directly protected data beans also implement the Protectable interface. If an indirectly protected data bean does not implement the Delegator interface, or returns a null value for the `getDelegate() method`, it is not protected and can be displayed by anyone.

Note: For more information on the Protectable interface, refer to the *WebSphere Commerce Programmer's Guide*

The following is an example of a resource-level policy for a data bean:

```
<Policy Name="AllUsersDisplayOrderDataBeanResourceGroup"
   OwnerID="RootOrganization"
   UserGroup="AllUsers"
   ActionGroupName="DisplayDataBeanActionGroup"
   ResourceGroupName="OrderDataBeanResourceGroup"
   RelationName="creator">
```

The ActionGroupName, `DisplayDataBeanActionGroup`, indicates that this policy is a policy for data beans. This action group includes one `Display` action.

Where:

`Name`: The name of this policy.

`UserGroup`: The access group that contains the users to whom the policy applies. In this case, it includes all users.

`ActionGroupName`: The value `DisplayDataBeanActionGroup` indicates that is is a resource-level policy for data beans.

`ResourceGroupName`: The name of the resource group that contains the data beans to be protected.

`RelationName`: The relationship that must be fulfilled between a user and the resource. In this case, the user must be the creator of the business `Order` resource.

The `OrderDataBeanResourceGroup` is defined as follows:

```
<ResourceGroup Name="OrderDataBeanResourceGroup" OwnerID="RootOrganization">
<ResourceGroupResource Name="com.ibm.commerce.order.beans.
OrderListDataBeanResourceCategory"/>
<ResourceGroupResource Name="com.ibm.commerce.order.beans
.OrderDataBeanResourceCategory"/>
</ResourceGroup>
```

The `OrderDataBeanResourceGroup` consists of two resources. The following is a sample resource definition for a DataBean:

```
<ResourceCategory Name="com.ibm.commerce.order.beans.OrderDataBeanResourceCategory"
ResourceBeanClass="com.ibm.commerce.order.beans.OrderDataBean">
<ResourceAction Name="DisplayDataBean"/>
</ResourceCategory>
```

Where:

`Name`: A tag used to refer to this resource in the XML file.

`ResourceBeanClass`: The class name of the databean that is being directly protected. This class must implement the Protectable interface.

`ResourceAction`: An element needed for policy editing in the Administration Console. In this case, this element indicates that `Display` is the valid action to be performed on this resource.

Grouping resources by attributes

Resource groups can be defined entirely by using the CONDITIONS column in the ACRESGRP table. The CONDITIONS column stores the XML document containing the constraints and attribute value pairs used for grouping resources. This type of resource group is called an implicit resource group, and is usually used when the class name of the resource is not sufficient. For example, if an access control policy applies to `Order` resources that have a status equal to `P` (pending) or `E` (editing by a customer service representative), a resource group can be defined for this.

Note: In order to group resources by attributes other than class name, the resource must implement the Groupable interface. For more information on the Groupable interface, refer to the *IBM WebSphere Commerce Programmer's Guide*.

The following is an example of the `Order` resource group:

```
<ResourceGroup  Name="OrderResourceGroupwithPEStatus"
      OwnerID="RootOrganization">
  <ResourceCondition>
   <![CDATA[
    <profile>
     <andListCondition>
   <orListCondition>
        <simpleCondition>
         <variable name="Status"/>
         <operator name="="/>
         <value data="P"/>
        </simpleCondition>
        <simpleCondition>
         <variable name="Status"/>
         <operator name="="/>
         <value data="E"/>
        </simpleCondition>
    </orListCondition>
        <simpleCondition>
         <variable name="classname"/>
         <operator name="="/>
        <value data="com.ibm.commerce.order.objects.Order"/>
       </simpleCondition>
     </andListCondition>
    </profile>
  ]]>
 </ResourceCondition>

</ResourceGroup>
```

Where:

`Name`: The name of the resource group stored in the GRPNAME column of the ACRESGRP table.

`OwnerID`: The owner of the resource group. This must be the root organization.

`<ResourceCondition>`: Specifies the data that will be loaded to the CONDITIONS column of the ACRESGRP table, to define the resource group.

`<![CDATA[...]]>`: Signifies a section of character data that are used exactly as they are typed .

`<profile>`: A required parameter for all resource conditions.

An essential component of the resource group definition is the `<simpleCondition>` element that has `name="classname"`. This element indentifies the java class of the resource that the group applies to. The java class, `com.ibm.commerce.order.objects.Order`, can be seen in the following example:

```
<simpleCondition>
   <variable name="classname"/>
   <operator name="="/>
   <value data="com.ibm.commerce.order.objects.Order"/>
   </simpleCondition>
```

The following example specifies the condition on the `com.ibm.commerce.objects.order.objects.Order` resource, that the status should equal `P`.

```
<simpleCondition>
 <variable name="Status"/>
       <operator name="="/>
       <value data="P"/>
   </simpleCondition>
```

In the above example, the `<variable name="`*`value`*`"/>` represents the attribute names recognized by the `getGroupingAttributeValue (String attributeName, GroupContext context)()` method on the resource. This method is part of the Groupable interface. For the purposes of Implicit Resource Group management within the WebSphere Commerce Administration Console, the attribute should also be defined in the ACATTR table and associated with the resource in the ACRESATREL table. When it is time to find the applicable policies for a given resource and action, this condition will be checked by calling the `getGroupingAttributeValue(..)` method, which in this case passes in `Status` as the `attributeName` parameter.

The `<orListCondition>`, specifies that the conditions within this block should be applied using a boolean `OR`. In this case, the status is either `P` or `E`. The`<andListConditon>`, specifies that the conditions within this block should be applied using a boolean AND. In this case, (Classname = com.ibm.commerce.order.objects.Order) AND (Status = P OR Status=E).

A sample attribute definition for populating the ACATTR table is shown in the following:

```
<Attribute Name="Status" Type="String">
</Attribute>
```

The Name element is a term to identify the attribute, and the Type element identifies the datatype of the attribute. Possible values of the attribute are:

- String
- Integer
- Double
- Currency
- Decimal
- URL
- Image
- Date

The association of an attribute to a resource is specified within the Resource definition. For example, theStatus attribute is associated with the OrderResourceCategory in the following example:

```
<ResourceCategory Name="com.ibm.commerce.order.objects.OrderResourceCategory"
     ResourceBeanClass="com.ibm.commerce.order.objects.Order" >

  <ResourceAttributes Name="Status"
   AttributeTableName="ORDERS"
   AttributeColumnName="STATUS"
   ResourceKeyColumnName="ORDERS_ID"/>
</ResourceCategory>
```

Where:

<ResourceAttributes>: A block of code that associates an attribute with a resource.

AttributeTableName: The name of the database table of the resource.

AttributeColumnName: The name of the column in the resource table that stores the attribute.

ResourceKeyColumnName: The name of the column in the resource table that stores the primary key.

Defining relationships

Access control policies have an optional relationship element. This relationship can only be created by loading an XML policy file with the relationship definition seen below:

```
  <Relation Name="value">
   </Relation>
```

The Name entry is the name of the relationship used in any policy, and is added to the ACRELATION table. Name corresponds to the relationship parameter of the fulfills() method on the protectable resource.

The following example displays the definition of a relationship called `creator`.

```
<Relation Name="creator">
</Relation>
```

Defining relationship groups

Relationship groups contain open conditions which are the conditions for belonging to the relationship group. If you need to define relationship groups, you must do so by defining the relationship group information in your XML file, or by modifying the `defaultAccessControlPolicies.xml` file as seen below:

```
<RelationGroup
  Name="aValue"
  OwnerID="aValue">
  <RelationCondition><![CDATA[
   <profile>
    Relationship Chain Open Condition XML
   </profile>
  ]]></RelationCondition>
 </RelationGroup>
```

Relationship chains

Each relationship group consists of one or more RELATIONSHIP_CHAIN open conditions, grouped by `andListCondition` or `orListCondition` elements. A relationship chain is a series of one or more relationships. The length of a relationship chain is determined by the number of relationships it consists of. This can be determined by examining the number of `<parmeter name= "X" value="Y">` entries in the XML representation of the relationship chain. The following is an example of a relationship chain with a length of one.

```
<openCondition name="RELATIONSHIP_CHAIN">
<parameter name="RELATIONSHIP"
value="aValue"/>
</openCondition>
```

Where:

`<parameter name="Relationshiop" value="something">`: A string representing the relationship between the user and the resource.

`name` : The relationship parameter of the `fulfills()` method on the protectable resource.

When a relationship chain has a length of two or more it is a series of two relationships. The first ,`<parmeter name= "X" value="Y">`, entry is between a user and an organizational entity. The last `<parmeter name= "X" value="Y">` entry is between an organizational entity and the resource. Intermediate `<parmeter name= "X" value="Y">` entries in the chain are between organizations. The following is an example of a relationship chain with a length of two.

```
<openCondition name=RELATIONSHIP_CHAIN">
<parameter name="aValue1" value="aValue2"/>
<parameter name="RELATIONSHIP" value="aValue3"/>
</openCondition>
```

Where:

`aValue1` : Possible values include `HIERARCHY` and `ROLE`. `HIERARCHY` specifies that there is a hierarchical relationship between the user and the organizational entity in the membership hierarchy. `ROLE` specifies that the user plays a role in the organizational entity. If the value of `aValue1` is `HIERARCHY`, the possible values include `child`, which returns the organizational entity for which the user is a direct child in the member hierarchy. If the value of aValue1 is ROLE, possible values include any valid entries in the NAME column of the ROLE table which return all of the organizational entities for which the current user plays this role.

aValue3: A string representing the relationship between one or more organizational entities retrieved from evaluating the first parameter and the resource. This value corresponds to the relationship parameter of the fulfills() method on the protectable resource. If more than one organizational entity was returned by evaluating parameter aValue1 , this part of the RELATIONSHIP_CHAIN is satisfied if at least one of these organizational entities satisfies the relationship specified by parameter aValue2.

Note: For more information on defining relationship groups, see "Defining relationship groups" on page 85

Defining single-chain relationship groups

If as part of your access control policy, you are required to enforce that a user must belong to the organizational entity that is for example, the BuyingOrganizationalEntity of the resource, you will have to create a relationship group that is composed of one relationship chain that has a length of two. This is shown in the following example:

```
<RelationGroup Name="MemberOf->BuyerOrganizationEntity"
OwnerID="RootOrganization
<RelationCondition><![CDATA[
<profile>
<openCondition name="RELATIONSHIP_CHAIN">
<parameter name="HIERARCHY" value="child"/>
<parameter name="RELATIONSHIP" value="BuyingOrganizationalEntity"/>
</openCondition>
</profile>
]]><RelationCondition>
<RelationGroup>
```

The relationship chain has a length of two because it consists of two separate relationships. The first relationship is between the user and its parent organizational entity. The user is the child in that relationship. For the second relationship, the access control policy manager checks if the parent organizational entity fulfills the BuyingOrganizationalEntity relationship with the resource. In other words, it returns true if it is the buying organizational entity of the resource.

Note: For information on the openCondition tag, refer to the *WebSphere Commerce Accelerator Customization Guide*.

Another example would be if you had to enforce that the user have the role of Account Representative for the organizational entity that is the buying organizational entity of the resource. Again, this uses a relationship group that is composed of one relationship chain of a length of two. The first part of the chain finds all of the organizational entities for which the user has the Account Representative role. Then for the set of organizational entities, the access control policy manager checks if at least one of them fulfills the BuyingOrganizationalEntity relationship with the resource. If it does, a value of true is returned.

The following example shows how to define this type of relationship group:

```
<RelationGroup Name="AccountRep->BuyerOrganizationalEntity"
OwnerID="RootOrganization">
<RelationCondition><![CDATA[
<profile>
```

```
 <openCondition name="RELATIONSHIP_CHAIN">
 <parameter name="ROLE" value="Account Representative"/>
 <parameter name="RELATIONSHIP" value="BuyingOrganizationalEntity"/>
</openCondition>
 </profile>
]]><RelationCondition>
<RelationGroup>
```

Defining multiple-chain relationship groups

If you need to compose a relationship group that contains a multiple-chain relationship, you must specify whether the user must satisfy all of the relationship chains, meaning it is an AND scenario, or the user must satisfy at least one of the relationship chains, meaning it is an OR scenario.

In the following example the user must be the creator of the resource and must belong to the `BuyingOrganizationalEntity` specified in the resource. The first chain, that specifies the user must be the creator of the resource is has a length of one. The second chain, that specifies that the user must belong to the `BuyingOrganizationalEntity` specified in the resource, has a length of two.

```
<RelationshipGroup Name="Creator_And_MemberOf->BuyerOrganizationalEntity"
 OwnerID="RootOrganization">
<RelationCondition><![CDATA[
<profile>
<andListCondition>
<openCondition name="RELATIONSHIP_CHAIN">
<parameter name="RELATIONSHIP" value="creator" />
</openCondition>
<openCondition name="RELATIONSHIP_CHAIN">
<parameter name="HIERARCHY" value="child"/>
<parameter name="RELATIONSHIP" value="BuyingOrganizationalEntity"/>
</openCondition>
</andListCondition>
</profile>
]]></RelationCondition>
</RelationGroup>
```

Note: If you require the user to satisfy either of the two relationship chains, the `<andListConditon>` tag should be changed to the `<orListConditon>` tag.

Access groups

The default access groups that are part of WebSphere Commerce are found in language specific XML files, such as *wc_install_directory*/xml/policies/xml/ACUserGroups_locale.xml. This file follows the DTD specified by *wc_install_directory*/xml/policies/dtd/ACUserGroups_en_US.dtd.

The following is the format of an access group element:

```
<UserGroup Name="value"
   OwnerID="value"
   Description="value"

   <UserCondition>
     <![CDATA[
       <profile>
        Condition XML
       </profile>
     ]]>
   </UserCondition>
</UserGroup>
```

Where:

Name: The name of the access group, stored in the MBRGRPNAME column of the MBRGRP table.

OwnerID: The Member ID that owns this access group. The combination of Name and OwnerID must be unique. Special values that can be used include: RootOrganization (-2001) or DefaultOrganization (-2000).

Description (optional): An optional attribute used to describe the access group.

UserCondition (optional): An optional element specifying implicit conditions of membership in this access group. This criteria is stored in the CONDITIONS column of the MBRGRPCOND table.

Condition XML: Using the condition framework, any valid combination of the orListCondition, andListCondition, simpleCondition, and trueConditionCondition elements.

The following SimpleCondition names are supported for the UserCondition element:

Table 5. Supported simple condition names

Variable Name	Description	Supported Operators	Supported Values	Qualifiers	Qualifier Values
role	Specifies that the user must have this role in the MBRROLE table·.	= !=	Any value of the NAME column in the ROLE table.	org (if not specified, the user must have the role for any organization in the MBRROLE table.	• OrgEntityID : Where the user must have the role. • ?: When it is used in a template policy.
registration status	Specifies that the user must have this registration status.	= !=	Any value of the REGISTER-TYPE column in the USERS table such as G for guest, and R for registered.	none	n/a
status	Specifies that the user must have this member state. This is usually used for the status of registration approval.	= !=	Any value of the STATE column in the MEMBER table such as 0 for pending registration approval, 1 for registration approved, and 2 for registration rejected.	none	n/a

Table 5. Supported simple condition names (continued)

org	Specifies that the user must be registered to this parent organization. This is stored in the MBRREL table.	= !=	• Any value of the ORGENTITY_ID in the ORGENTITY table. • ?- if it is a template policy.	none	n/a

Note: The ? will be dynamically changed to the resource-owning organizational entity and subsequently it's ancestors when the template policy is applied at runtime. Access groups defined with ? only work with template policies.

Examples of simpleConditions for access groups

role:

Role without a qualifier: The following example displays a `role` simpleCondition without a qualifier; most commonly used in role-based policies. In this example the user must have a Seller Administration role for any organizational entity.

```
<UserConditon>
  <![CDATA[
  <profile>
   <simpleCondition>
    <variable name="role"/>
    <operator name="="/>
    <value data="Seller Administrator"/>
   </simpleCondition>
  </profile>
]]>
</UserCondition>
```

Role with a qualifier: The following example displays a `role` simpleCondition with a qualifier; most commonly used for organization-level policies. In this example the user must have a Seller role for organizational entity `100`.

```
<UserCondition>
   <!CDATA[
  <profile>
    <simpleCondition>
    <variable name="role"/>
    <operator name="="/>
    <value data="Seller"/>
     <qualifier name="org" data="100"/>
    <simpleCondition>
  </profile>
]]>
</UserCondition>
```

Role with a qualifier and parameter: The following example displays a `role` simpleCondition with a qualifier and parameter. This works only in template policies. In this example, the user must have a Sales Manager, Account Manager, or Seller role in the organizational entity that owns the resource specified in the template policy.

```
<UserCondition><!CDATA[
  <profile>
   <orListCondition>
```

```
      <simpleCondition>
       <variable name="role"/>
       <operator name="="/>
       <value data="Sales Manager"/>
        <qualifier name="org" data="?"/>
      </simpleCondition>
      <simpleCondition>
       <variable name="role"/>
       <operator name="="/>
       <value data="Account Representative"/>
        <qualifier name="org" data="?"/>
      </simpleCondition>
      <simpleCondition>
       <variable name="role"/>
       <operator name="="/>
       <value data="Seller"/>
        <qualifier name="org" data="?"/>
      </simpleCondition>
    </orListCondition>
  </profile/>
]]></UserCondition>
```

registrationStatus: The following example displays a `registrationStatus` simpleCondition. In this example, the user must be registered (`USERS.REGISTERTYPE = R`).

```
<UserCondition><!CDATA[
  <profile>
  <simpleCondition>
   <variable name="registrationStatus"/>
   <operator name="="/>
   <value data="R"/>
  </simpleCondition>
  </profile>
]]></UserCondition>
```

status: The following example displays a `status` simpleCondition. In this example, the user must have had registration approved. (`MEMBER.STATUS = 1`)

```
<UserCondition><![CDATA[
  <profile>
    <simpleCondition>
     <variable name="status"/>
     <operator name="="/>
     <value data="1"/>
     </simpleCondition>
  </profile>
]]></UserCondition>
```

org: The following example displays an `org` simpleCondition. In this example, the user must be registered in organizational entity `100`. In the MBRREL table, the user must have the values of ANCESTOR_ID = `100`, and SEQUENCE = `1`.

```
<UserCondition><![CDATA[
  <profile>
   <simpleCondition>
    <variable name="org"/>
    <operator name="="/>
    <value data="100"/>
   </simpleCondition>
  </profile>
]]>
</UserCondition>
```

Policies

The *wc_install_directory*/xml/policies/xml/defaultAccessControlPolicies.xml file defines the default access control policies that are shipped out of the box. It follows the DTD specified by: *wc_install_directory*/xml/policies/dtd/accesscontrolpolicies.dtd.

The following is the template of a policy element:

```
<Policy Name="value"
  OwnerId="value"
  UserGroup="value"
  UserGroupOwner="value"
  ActionGroupName="value"
  ResourceGroupName="value"
  PolicyType="value"
  RelationName="value"
  RelationGroupName="value"
  RelationGroupOwner="value"
</Policy>
```

Where:

Name: The name of the policy. This is loaded into the POLICYNAME column of the ACPOLICY table. The Name and OwnerID together must be unique.

OwnerID: The member ID of the organizational entity that owns the policy. This will be loaded into the member_id column of the ACPOLICY table. The OwnerID and Name together must be unique. There are two special values that are recognized by the transformer tool, these are the RootOrganization: -2001, and DefaultOrganization: -2000

UserGroup: The name of the access group specified in the MBRGRPNAME column of the MBRGRP table. This is loaded into the mbrgrp_id column of the ACPOLICY table. The default access groups are defined in the wc_install_directory/xml/policies/xml/ACUserGroups_language.xml file.

UserGroupOwner: The member ID of the member that owns the Access Group. This is needed when the access group is owned by a member other than the policy owner. If this is not specified, it is assumed that the access group is owned by the member that is specified by the OwnerID attribute.

ActionGroupName: The name of the action group specified in GROUPNAME column of ACACTGRP table. It is used to get the corresponding action group ID (ACACTGRP_ID) that will be stored in the ACPOLICY table. Role-based policies for controller commands have ActionGroupName set to ExecuteCommandActionGroup. Policies for databeans have ActionGroupName set to DisplayDatabeanActionGroup.

ResourceGroupName: The name of the Resource Group, specified in the GRPNAME column of the ACRESGRP table. It is used to get the corresponding resource group ID (ACRESGRP_ID) that is stored in the ACPOLICY table. Role-based policies for views have ResoureGroupName set to ViewCommandResourceGroup.

PolicyType: The type of policy. Valid values are template (POLICYTYPE will be set to 1 in the ACPOLICY table). If this attribute is not specified, the policy type value will remain unchanged. (By default the value of this column is null. Any value other than 1 implies a non-template policy type). For more information on the types of policies see .

`RelationName (optional)`: The name of the Relationship, as specified in the RELATIONNAME column of the ACRELATION table. If it is specified, it is used to get the corresponding relationship ID (ACRELATION_ID) that is stored in the ACPOLICY table.

`RelationGroupName (optional)`: The name of the Relationship Group, as specified in the GRPNAME column of the ACRELGRP table. If this attribute is specified, `RelationName` should not be specified, since Relationship Group takes precedence.

`RelationGroupOwner`: The member ID that owns the Relation Group. This attribute is necessary only if the `RelationGroupName` attribute is specified and if the value of the `OwnerID` attribute is not RootOrganization; in this case, `RelationGroupOwner` must be specified as RootOrganization (-2001).

Policy examples

Role-based policies:

For controller commands: In this example, users belonging to the `AllUsers` access group can execute controller commands that are part of the `AllUserCmdResourceGroup` resource group.

```
<Policy Name="AllUsersExceuteAllUserCmdResourceGroup"
   OwnerID="RootOrganization"
   UserGroup="AllUsers"
   ActionGroupName="ExecuteCommandActionGroup"
   ResourceGroupName="AllUserCmdResourceGroup">
</Policy>
```

For views: In this example, users belonging to the `MarketingManagers` access group can execute the views belonging to `MarketingManagersViews` action group.

```
<Policy Name="MarketingManagersExecuteMarketingManagersViews"
   OwnerID="RootOrganization"
   UserGroup="MarketingManagers"
   ActionGroupName="MarketingManagersViews"
   ResourceGroupName="ViewCommandResourceGroup">
</Policy>
```

Resource-level policies:

For commands: In this example, users belonging to `RegisteredApprovedUsers` access group can perform the actions specified by the `CouponRedemption` action group on resources specified by the `CouponWalletResourceGroup`, as long as the users fulfill the `creator` relationship with respect to the resource.

```
<Policy Name="RegisteredApprovedUsersExecuteCouponRedemptionCommandsOn
WalletResource"
   OwnerID="RootOrganization"
   UserGroup="RegisteredApprovedUsers"
   ActionGroupName="CouponRedemption"
   ResourceGroupName="CouponWalletResourceGroup"
   RelationName="creator">
</Policy>
```

For DataBeans: In this example, users belonging to the `AllUsers` access group can Display databeans specified by the `UserDatabeanResourceGroup` resource group, as long as the users fulfill the `owner` relationship with respect to the resource.

```
<Policy Name="AllUsersDisplayUserDatabeanResourceGroup"
   OwnerID="RootOrganization"
   UserGroup="AllUsers"
   ActionGroupName="DisplayDatabeanActionGroup"
   ResourceGroupName="UserDatabeanResourceGroup"
   RelationName="owner">
</Policy>
```

Template policies: In this example, users belonging to the `MembershipAdministratorsForOrg` access group, can perform the actions specified by the `ApproveGroupUpdate` action group on resources specified by the `OrganizationDataResourceGroup`.

```
<Policy Name=MembershipAdministratorsForOrgExecuteApproveGroupUpdateCommands
OnOrganizationResource"
   OwnerID="RootOrganization"
   UserGroup="MembershipAdministratorsForOrg"
   ActionGroupName="ApproveGroupUpdate"
   ResourceGroupName="OrganizationDataResourceGroup"
   PolicyType="template">
</Policy>
```

When this template policy is applied, the policy owner will dynamically change from RootOrganization to the organizational entity that owns the resource and subsequently, its ancestor organizational entities, up to and including Root Organization. Examining the definition of the `MembershipAdministratorsForOrg` access group would reveal the following condition for membership:

```
<UserCondition><![CDATA[
<profile>
 <orListCondition>
  <simple condition>
   <variable name="role"/>
   <operator name="="/>
   <value data="Buyer Administrator"/>
    <qualifier name="org" data="?"/>
  </simpleCondition>
  <simpleConditon>
   <variable name="role"/>
   <operator name="="/>
   <value data="Seller Administrator"/>
    <qualifier name="org" data="?"/>
  </simpleConditon>
</orListCondtion>
</profile>
]]></UserCondition>
```

Note: The `simpleCondition` of `role` is qualified by `org = `**`?`**. This **`?`** is dynamically substituted along with the policy owner, as explained above. This dynamic behavior is only available to template policies. So in this example, users that have the `Buyer Administrator` or `Seller Administrator` role for the organizational entity that owns the resource, satisfy the condition for membership in this access group.

Translatable policy data

The following is a template of the translatable access control elements that at a minimum, must be defined in the `defaultAccessControlPolicies_`*`locale`*`.xml` file.

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="no" ?>
 <!--The following TRANSLATABLE access control elements should
 be defined in this file:
 <Attribute_nls>
 <Action_nls>
 <Relation_nls>
 <ResourceCategory_nls>
 <ActionGroup_nls>
 <ResourceGroup_nls>
 <Policy_nls>-->
<!DOCTYPE PoliciesNLS SYSTEM "../dtd/accesscontrolpoliciesnls.dtd">

<PoliciesNLS LanguageID="value">


 <!--Insert access control element definitions here -->
 </PoliciesNLS>
```

The `LanguageID` attribute is a string corresponding to the language of the locale-specific data. Valid values of the `LanguageID` are:

- en_US
- fr_FR
- de_DE
- it_IT
- es_ES
- pt_BR
- zh_CN
- zh_TW
- ko_KR
- ja_JP

Non-translatable policy data

The following is a template of a customized policy file containing non-translatable data:

```
  <?xml version="1.0" encoding="ISO-8859-1" standalone="no" ?>

<!DOCTYPE Policies SYSTEM "../dtd/accesscontrolpolicies.dtd">

 <!--The following NON-TRANSLATABLE access control elements
 should be defined in this file:

 <Attribute>
 <Action>
 <ResourceCategory>
 <Relation>
 <RelationGroup>
 <ActionGroup>
 <ResourceGroup>
 <Policy-->
<Policies>

 <!--Insert access control element definitions here-->
 </Policies>
```

Locale-specific data

The following optional locale-specific data can be loaded to give additional description to the access control elements already defined in the non-translatable XML file. The default locale-specific data can be found at the following address:

```
wc_install_directory\xml\policies\xml\
defaultAccessControlPolicies_locale.xml
```

For example, `defaultAccessControlPolicies_en_US.xml`.

Attribute: The following example defines additional attribute element information:

```
<Atrribute_nls AttributeName="Status"
DisplayName_nls="Status attribute"
Description_nls="Resource status attribute"
/>
```

Where:

`AttributeName`: The name of the attribute. This value is stored in the ATTRNAME column of the ACATTR table.

`DisplayName_nls`: The display name of the attribute. This value is stored in the DISPLAYNAME column of the ACATTRDESC table.

`Description_nls`: An optional description of the attribute. This value is stored in the DESCRIPTION column of the ACATTRDESC table.

Action: The following example defines additional action element information:

```
<Action_nls ActionName="OrderAdjustmentButton"
DisplayName_nls="Order Adjustment Button View"
Description_nls="The view for loading buttons in the order adjustment page
 when placing an order from Commerce Acclerator"
/>
```

Where:

`ActionName`: The name of the action. This value is stored in the ACTION column of the ACACTION table.

`DisplayName_nls`: The display name of the action. This value is stored in the DISPLAYNAME column of the ACACTDESC table.

`Description_nls`: An optional description of the action. This value is stored in the DESCRIPTION column of the ACACTDESC table.

Relation: The following example defines additional relation element information:

```
<Relation_nls RelationName="creator"
DisplayName_nls="creator"
Description_nls="creator"
/>
```

Where:

`RelationName`: The name of the relationship. This value is stored in the RELATIONNAME column of the ACRELATION table.

`DisplayName_nls`: The display name of the relationship. This value is stored in the DISPLAYNAME column of the ACRELDESC table.

`Description_nls`: An optional description of the relationship. This value is stored in the DESCRIPTION column of the ACRELDESC table.

Resource Category: The following example defines additional resource category information:

```
<ResourceCategory_nls ResourceCategoryName="com.ibm.commerce.
catalog.objects."InterestItemList"
DisplayName_nls="Interest Item List"
Description_nls="Interest Item List command"
/>
```

Where:

`ResourceCategoryName`: The name of the resource category. This value is stored in the RESCLASSNAME column of the ACRESCGRY table.

`DisplayName_nls`: The display name of the resource category. This value is stored in the DISPLAYNAME column of the ACRSCGDES table.

`Description_nls`An optional description of the resource category. This value is stored in the DESCRIPTION column of the ACRSCGDES table.

Action Group: The following example defines additional action group information:

```
<ActionGroup_nls ActionGroupName="DoEverything"
DisplayName_nls="Do Everything"
Description_nls="Permits access to all Actions"
/>
```

Where:

`ActionGroupName`: The name of the action group. This value is stored in the GROUPNAME column of the ACACTGRP table.

`DisplayName_nls`: The display name of the action group. This value is stored in the DISPLAYNAME column of the ACACGPDESC table.

`Description_nls`: An optional description of the action group. This value is stored in the DESCRIPTION column of the ACACGPDESC table.

Resource Group: The following example defines additional resource group information:

```
<ResourceGroup_nls ResourceGroupName="AllResourceGroup"
DisplayName_nls="All Resources Group"
Description_nls="All Resources"
/>
```

Where:

`ResourceGroupName`: The name of the resource group. This value is stored in the GRPNAME column of the ACRESGRP table.

`DisplayName_nls`: The display name of the resource group. This value is stored in the DISPLAYNAME column of the ACRESGPDES table.

`Description_nls`: An optional description of the resource group. This value is stored in the DESCRIPTION column of the ACRESGPDES table.

Policy: The following example defines additional policy information:

```
<Policy_nls PolicyName="SiteAdministratorsCanDoEverything"
OnwerID="RootOrganization"
DisplayName_nls="Site Administrators Can Do Everything"
Description_nls="Policy that allows Site Administrators to do everything"
/>
```

Where:

`PolicyName`: The name of the access control policy. This value is stored in the POLICYNAME column of the ACPOLICY table.

`OwnerID`: The member ID of the organizational entity that owns this policy.

`DisplayName_nls`: The display name of the policy. This value is stored in the DISPLAYNAME column of the ACPOLDESC table.

`Description_nls`: An optional description of the policy. This value is stored in the DESCRIPTION column of the ACPOLDESC table.

After you have changed the XML files

Testing your changes

For information on testing your changes see "After you make your policy changes" on page 43.

Loading your changes into the database

If you make policy changes by working directly with the XML files, you must load the changed XML files back into the databases. It is important to maintain consistency between the XML files and the access control information in the databases for several reasons:

- When you create an instance of WebSphere Commerce, the policy and access group definitions are loaded from the XML files.
- If you want to implement the same access control policies in a second instance of WebSphere Commerce, you can do so by copying the XML files to the appropriate directory before creating the second instance.
- The XML files offer a convenient way to directly view and edit your policies and their component parts so keeping the files up-to-date is essential.

Loading your XML changes into the database

The loading process reads the XML files containing the access control policy information and access group definitions and loads them into the appropriate databases. The policy and access group information contained in the XML files is loaded at installation, however, you must reload the files if you make changes to them.

Note: If you create customized XML files, you need to copy them into the `<`*`wc_install_directory`*`>/xml/policies/xml` directory to have them loaded into the databases.

For 400 : if you create customized XML files, you must use the full path to the DTD in your file. The access control policies DTDs are located in `/QIBM/ProdData/WebCommerce/xml/policies/dtd`.

To load the access groups and access control policies, run the following commands.

For NT 2000

1. From the directory `<`*`wc_install_directory`*`>\bin`, run the following command files as needed in the order listed here:

- To load the user (access) group definitions, run the **acugload** command file. **Syntax**: acugload.cmd <*database name*> <*database user*> <*database user password*> <*UserGroups xml file*> **Example**: acugload mall dbuser dbusrpwd ACUserGroups_en_US.xml
- To load the main access control policies file, run the **acpload** command file. **Syntax:** acpload.cmd <*database name*> <*database user*> <*database user password*> <*Policies xml file*> **Example**: acpload mall dbuser dbusrpwd defaultAccessControlPolicies.xml
- To load the display names and descriptions file, run the **acpnlsload** command file. **Syntax:** acpnlsload.cmd <*database name*> <*database user*> <*database user password*> <*NLS Policies xml file*> **Example**: acpnlsload mall dbuser dbusrpwd defaultaccesscontrolpolicies_en_US.xml

2. Check the log files **acugload.log**, **acpload.log,** and **acpnlsload.log** in `<wc_install_directory>\logs` for any errors.

For AIX Solaris Linux

+ The database user ID must have read/write/execute authority to the directories
+ `<wc_install_directory>/xml/policies, and <wc_install_directory>/bin` , as
+ well as their sub-directories and files. Also ensure the database user ID has
+ read/write authority on `<wc_install_directory>/logs/messages.txt` and
+ `<wc_install_directory>/logs/trace.txt`.

1. Login as the database user ID.
2. From the directory `<wc_install_directory>/bin`, run the following shell scripts as needed in the order listed here:

1. To load the user (access) group definitions, run the **acugload** shell script. **Syntax:** acugload.sh <*database name*> <*database user*> <*database* user *password*> <*UserGroups xml filename*> **Example**: acugload mall dbuser dbusrpwd ACUserGroups_en_US.xml
2. To load the main access control policies file, run the **acpload** shell script. **Syntax:** acpload.sh <*database name*> <*database user*> <*database user password*> <*Policies xml filename*> **Example**: acpload mall dbuser dbusrpwd defaultAccessControlPolicies.xml
3. To load the display names and descriptions file, run the `acpnlsload` shell script. **Syntax:** acpnlsload.sh <*database name*> <*database user*> <*database user password*> <*NLS Policies xml filename*> **Example**: acpnlsload mall dbuser dbusrpwd defaultaccesscontrolpolicies_en_US.xml

Check the log files `acugload.log`, `acpload.log`, and `acpnlsload.log` in `<wc_install_directory>/logs` for any errors.

Note: After performing these scripts you must check the log files, as any errors that may occur while running these scripts will not appear on the command line.

For 400

From the command line, run the following commands as needed in the order specified.

- To load the user (access) group definitions, run the `LODWCSUG` command. **Syntax:** LODWCSUG DATABASE(<*database name*>) SCHEMA(<*schema_name*>) PASSWD(<*instance_password*>) INSTROOT(<*instance_root*>) INFILE(<*full path for XML file*>)

- To load the main access control policies file, run the `LODWCSAC` command. **Syntax:** LODWCSAC DATABASE (<database name>) SCHEMA (<schema_name>) PASSWD (<instance_password>) INSTROOT (<instance_root>) INFILE (<*full path for XML file*>)
- To load the display names and descriptions file, run the `LODWCSACD` command. **Syntax:** LODWCSACD DATABASE (<*database name*>) SCHEMA (<*schema_name*>) PASSWD (<*instance_password*>) INSTROOT (<*instance_root*>) INFILE (<*full path to XML file*>)

Extracting policy and access group definitions from the databases into your XML files

The extraction process reads the policy and access group information in the access control databases and generates files that capture the information in XML format.

For NT 2000

1. From the <*wc_install_directory*>`\bin` directory, run the following `acpextract` command:

   ```
   acpextract.cmd <database name> <database user> <database user password>
   ACPoliciesfilter.xml
   ```

   For example,

   ```
   acpextract.cmd mall dbuser dbusrpwd ACPoliciesfilter.xml
   ```

   The following files are created:
   - `ExtractedACPolicies.xml`: This file contains data extracted by the `Extract` command for the given filter criteria.
   - `ExtractedACPolicies.dtd`: The DTD for the `ExtractedACPolicies.xml` file.
   - `AccessControlUserGroups.xml`: The file containing the access group definitions.
   - `AccessControlPolicies.xml`: The file containing the language-independent access control policy information.
   - `AccessControlPolicies_LOCALE.xml`: The language-dependent access control policies file that contains the display names and descriptions.
2. Check the log file <*wc_install_directory*>`\logs\acpextract.log` for any processing errors that might have occurred.

For AIX Solaris Linux

1. Login as the database user ID.
2. From the <*wc_install_directory*>`\bin` directory, run the following `acpextract` shell script:

   ```
   acpextract.sh <database name> <database user>
   <database user password> ACPoliciesfilter.xml
   ```

   For example,

   ```
   acpextract.sh mall dbuser dbusrpwd ACPoliciesfilter.xml
   ```

   The following files are created:
   - `ExtractedACPolicies.xml`: This file contains data extracted by the `Extract` command for the given filter criteria.
   - `ExtractedACPolicies.dtd`: The DTD for the `ExtractedACPolicies.xml` file.

- AccessControlUserGroups.xml: The file containing the access group definitions.
- AccessControlPolicies.xml: The file containing the language-independent access control policy information.
- AccessControlPolicies_LOCALE.xml: The language-dependent access control policies file that contains the display names and descriptions.

3. Check the log file <*wc_install_directory*>\logs\acpextract.log for any processing errors that might have occurred.

For 400

1. From the command line, run the following EXTWCSAC command:

```
EXTWCSAC DATABASE (<database name>)
 SCHEMA (<schema_name>) PASSWD (<database user>)
INSTROOT (<instance_root>) FILTER (<input filter XML file>) OUTDIR
 (<output directory
 for new files>)
```

   The following files are created in the directory specified using the OUTDIR parameter:

   - ExtractedACPolicies.xml: This file contains data extracted by the Extract command for the given filter criteria.
   - ExtractedACPolicies.dtd: The DTD for the ExtractedACPolicies.xml file.
   - AccessControlUserGroups.xml: The file containing the access group definitions.
   - AccessControlPolicies.xml: The file containing the language-independent access control policy information.
   - AccessControlPolicies_LOCALE.xml: The language-dependent access control policies file that contains the display names and descriptions.

Appendix. Default access control policies

The Appendix lists the default policies provided with WebSphere Commerce. They are organized into the following categories:

- **Role-based policies:** The role-based policies for each default role. These policies are also referred to as command-level policies because they define who can execute each command.
- **Resource-level policies:** The resource-level policies, grouped by business area. These policies define the actions a group of users can perform on specific resources. Under each business area, policies are organized by the type of resource they regulate:
  - **Data resources** - business objects that can be manipulated such as an order or a bid.
  - **DataBean resources** - contain information about business objects. DataBeans are used to display object information on a Web page.

Table 6.

Policies	**Starting on page**
Role-based policies	"Role-Based Policies" on page 102
Resource-level policies by business area:	"Resource-Level Policies By Business Area" on page 103
Orders	"Orders" on page 103
Trading (contracts)	"Trading (Contracts)" on page 104
Approvals	"Approvals" on page 104
Auctions	"Auctions" on page 105
Business Intelligence	"Business Intelligence" on page 105
Membership	"Membership" on page 105
Buyer administration console	"Buyer Administration Console" on page 106
Campaigns	"Campaigns" on page 106
Catalog	"Catalog" on page 107
Connectvity and notification	"Connectivity and Notification" on page 107
Procurement	"Procurement" on page 108
Coupons	"Coupons" on page 108
Customer profiling	"Customer Profiling" on page 108
Discounts	"Discounts" on page 108
Inventory	"Inventory Management" on page 109
Scheduled Inventory	"Scheduled Inventory" on page 109
Inventory management	"Inventory Management" on page 109
Order management	"Order Management" on page 110
Payment	"Payment" on page 110
Administration console pages for editing policies, access groups, resource groups, and action groups	"Administration console pages for editing policies, access groups, resource groups, and action groups" on page 111
Product Advisor	"Product Advisor" on page 111

Table 6. (continued)

RFQ	"RFQ" on page 111
Rules	"Rules" on page 112
Scheduler	"Scheduler" on page 112

Role-Based Policies

Table 7.

AccountRepresentativesExecuteAccountRepresentativesCmdResourceGroup
AccountRepresentativesExecuteAccountRepresentativesViews
AllUsersExecuteAllUserCmdResourceGroup
AllUsersExecuteAllUsersViews
BecomeUserCustomerServiceGroupExecutesBecomeUserCmdsResourceGroup
BuyerAdministratorsExecuteBuyerAdminstratorsViews
BuyerAdministratorsExecuteBuyerAdminstratorsCommands
BuyerApproversExecuteBuyerApproversCmdResourceGroup
BuyerApproversExecuteBuyerApproversViews
Buyers(buy-side)ExecuteBuyers(buy-side)CommandsResourceGroup
Buyers(buy-side)ExecuteBuyers(buy-side)Views
Buyers(sell-side)ExecuteBuyers(sell-side)CommandsResourceGroup
Buyers(sell-side)ExecuteBuyers(sell-side)Views
CategoryManagersExecuteCategoryManagersCmdResourceGroup
CategoryManagersExecuteCategoryManagersView
CustomerServiceRepresentativesExecuteCustomerServiceRepCmdResourceGroup
CustomerServiceRepresentativesExecuteCustomerServiceRepresentativeView
CustomerServiceSupervisorsExecuteCustomerServiceSupervisorCmdResourceGroup
CustomerServiceSupervisorsExecuteCustomerServiceSupervisorViews
CustomersExecuteCustomersViews
GuestsExecuteGuestUsersCmdResourceGroup
LogisticsManagersExecuteLogisticsManagersCmdResourceGroup
LogisticsManagersExecuteLogisticsManagersViews
MarketingManagersExecuteMarketingManagerCmdResourceGroup
MarketingManagersExecuteMarketingManagersViews
OperationsManagersExecuteOperationsManagersCmdResourceGroup
OperationsManagersExecuteOperationsManagersView
PickPackersExecutePickPackersCmdResourceGroup
PickPackersExecutePickPackersViews
ProcurementBuyersExecuteProcurementBuyersCmdResourceGroup
ProductManagersExecuteProductManagersCmdResourceGroup
ProductManagersExecuteProductManagersViews
ReceiversExecuteReceiversCmdResourceGroup
ReceiversExecuteReceiversViews

Table 7. (continued)

ReturnsAdministratorsExecuteReturnsAdministratorsCmdResourceGroup
ReturnsAdministratorsExecuteReturnsAdministratorsViews
SalesManagersExecuteSalesManagersCmdResourceGroup
SalesManagersExecuteSalesManagersViews
SellerAdministratorsExecuteSellerAdministratorsCommands
SellerAdministratorsExecuteSellerAdministratorsViews
SellersExecuteSellersCmdResourceGroup
SellersExecuteSellersView
SiteAdministratorsCanDoEverything
StoreAdministratorsExecuteStoreAdministratorsCmdResourceGroup
StoreAdministratorsExecuteStoreAdministratorViews

Resource-Level Policies By Business Area

Orders

Table 8.

Data Resources	
Order	AllUsersExecuteOrderCreateCommandsOnStoreResource
	AllUsersExecuteOrderPrepareCommandsOnOrderResource
	AllUsersExecuteOrderProcessOnOrderResource
	AllUsersExecuteOrderReadCommandsOnOrderResource
	AllUsersExecuteOrderWriteCommandsOnOrderResource
	AllUsersExecuteReturnAgainstOrderOnOrderResource
	AllUsersExecuteScheduledOrderCancelOnOrderResource
	OrderManagersForOrgExecuteOrderManageCommandsOnOrderResource
Requisition List	AllUsersExecuteRequisitionListCreateCommandsOnStoreEntityResource
	AllUsersExecuteRequisitionListExclusiveProcessCommands OnPrivateRequisitionListResource
	AllUsersExecuteRequisitionListExclusiveReadCommandsOn PrivateRequisitionListResource
	AllUsersExecuteRequisitionListSharedProcessCommandsOn SharedRequisitionListResource
	AllUsersExecuteRequisitionListSharedReadCommandsOn SharedRequisitionListResource
	AllUsersExecuteRequisitionListWriteCommandsOn RequisitionListResource
Interest Item	AllUsersExecuteInterestItemReadCommandsOnInterestItemListResource
	AllUsersExecuteInterestItemWriteCommandsOnInterestItemListResource

Table 8. (continued)

RMA (Return Merchandise Authorization)	AllUsersExecuteRMACreateCommandsOnStoreResource
	AllUsersExecuteRMAProcessCommandsOnRMAResource
	AllUsersExecuteRMAReadCommandsOnRMAResource
	AllUsersExecuteRMAWriteCommandsOnRMAResource
	RMAApproversForOrgExecuteRMAApproveCommandsOnRMAResource
	RMADisposersForOrgExecuteRMADisposeCommandsOnRMAResource
	RMAManagersForOrgExecuteRMAManageCommandsOnRMAResource
	RMAReceiversForOrgExecuteRMAReceiveCommandsOnRMAResource
	StoreAdministratorsForOrgExecuteRMACreditCommandsOnStoreEntityResource
DataBeans	
Order	AllUsersDisplayApprovalsOrderDataBeansResourceGroup
	AllUsersDisplayOrderDatabeanResourceGroup
Requisition List	AllUsersDisplaySharedRequisitionListDataBeansIfSame OrganizationalEntityAsCreator
Interest Item	AllUsersDisplayInterestItemDatabeanResourceGroup
RMA	AllUsersDisplayRMADatabeanResourceGroup

Trading (Contracts)

Table 9.

Data Resource	
Contract	ContractAdministratorsForOrgExecuteContractCreateCommandsOn MemberResource
	ContractAdministratorsForOrgExecuteContractManageCommandsOn ContractResource
	ContractOperatorsForOrgExecuteContractDeployCommandsOn ContractResource
	ContractOperatorsForOrgExecuteContractSubmitCommandsOn ContractResource
	ContractViewersExecuteContractDisplayCommandsOnContractResource
Business Policy	BusinessPolicyAdministratorsForOrgExecuteBusinessPolicy CreateCommandsOnStoreResoure
	BusinessPolicyAdministratorsForOrgExecuteBusinessPolicy ManageCommandsOnBusinessPolicyResourcee
DataBeans	AccountHandlersDisplayTradingDatabeanResourceGroup

Approvals

Table 10.

Data Resources	

Table 10. (continued)

	AllUsersExecuteAllUsersActionGroupCommandsOnOrderResource
	AllUsersExecuteApproveCommandsOnApprovalResource
	AllUsersExecuteCancelApproveCommandsOnApprovalResource

Auctions

Table 11.

Data Resources	
Auction	AuctionAdministratorsForOrgExecuteAdminRetractBidCommands OnAuctionResource
	AuctionAdministratorsForOrgExecuteAuctionCreateCommands OnStoreEntityResource
	AuctionAdministratorsForOrgExecuteAuctionManageCommands OnAuctionResource
Auction Style	AuctionAdministratorsForOrgExecuteAuctionStyleCreateCommands OnStoreEntityResource
	AuctionAdministratorsForOrgExecuteAuctionStyleManageCommands OnAuctionStyleResource
Bid Control Rule	AuctionAdministratorsForOrgExecuteBidControlRuleCreateCommands OnStoreEntityResource
	AuctionAdministratorsForOrgExecuteBidControlRuleManageCommands OnBidControlRuleResource
Bid	RegisteredApprovedUsersExecuteBidCreateCommandsOnAuctionResource
	RegisteredApprovedUsersExecuteBidManageCommands OnBidResourcesTheyOwn
AutoBid	RegisteredApprovedUsersExecuteAutoBidCreateCommands OnAuctionResource
	RegisteredApprovedUsersExecuteAutoBidManageCommands OnAutoBidResourcesTheyOwn
DataBeans	AuctionDatabeanOwnersDisplayAuctionDatabeans

Business Intelligence

Table 12.

Data Resources	
	BusinessAnalystsForOrgExecuteViewContext ListCommandsOnStoreEntityResource
	IntelligenceReportViewersForOrgExecuteViewBusinessIntelligenceReport CommandsOnStoreEntityResource

Membership

Table 13.

Data Resources	
User	GuestsExecuteUserSelfRegistrationCommandsOnOrganizationResource

Table 13. (continued)

	`MembershipAdministratorsForOrgExecuteUserAdminRegistration CommandsOnOrganizationResource`
	`MembershipAdministratorsForOrg ExecuteUserAdminUpdateCommandsOnUserResource`
	`NonRejectedUsersExecuteUserSelfRegistration ContinuationCommandsOnUserResource`
Organization	`MembershipAdministratorsForOrgExecute OrgEntityRegistrationCommandsOnOrganizationResource`
	`MembershipAdministratorsForOrgExecuteOrg EntityUpdateCommandsOnOrganizationResource`
Address	`MembershipAdministratorsForOrg ExecuteAddressManageCommandsOnMemberResource`
	`NonRejectedUsersExecuteAddressManageCommandsOn UserResource`
Role	`MembershipAdministratorsForOrgExecute RoleManageCommandsOnUserResource`
	`OrganizationRoleAdministratorsExecute RoleManageCommandsOnOrganizationResource`
Member Group	`MemberGroupAdministratorsForOrgExecute MemberGroupCreateCommandsOnMemberResource`
	`MemberGroupManagersForOrgExecute MemberGroupManageCommandsOnMemberGroupResource`
DataBeans	`MembershipAdministratorsForOrgDisplay OrganizationDatabeanResourceGroup`
	`MembershipViewersForOrgDisplayMembershipDatabeanResourceGroup`

Buyer Administration Console

Table 14.

Data Resources	
Approval Group	`MembershipAdministratorsForOrgExecute ApproveGroupUpdateCommandsOnOrganizationResource`
Member Group	`MembershipAdministratorsForOrgExecute MemberGroupMemberUpdateCommandsOnMemberGroupResource`
	`MembershipAdministratorsForOrgExecute MemberGroupMemberUpdateCommandsOnUserResource`

Campaigns

Table 15.

Data Resources	
	`CampaignManagersForOrgExecute CampaignRelatedCreateCommandsOnStoreEntityResource`
	`CampaignManagersForOrgExecute CampaignUpdateCommandsOnCampaignResource`

Table 15. (continued)

	CampaignManagersForOrgExecute CollateralUpdateCommandsOnCollateralResource
	CampaignManagersForOrgExecute EMarketingSpotUpdateCommandsOnEMarketingSpotResource
	CampaignManagersForOrgExecute InitiativeUpdateCommandsOnInitiativeResource
DataBeans	CampaignManagersForOrgDisplayCampaignDatabeanResourceGroup

Catalog

Table 16.

Data Resources	
	CatalogEntryManagersForOrgExecute CatalogEntryManageCommandsOnCatalogEntryResource
	CatalogEntryManagersForOrgExecute CatalogEntryRelationManageCommandsOnCatalogResource
	CatalogEntryManagersForOrgExecute StoreCatalogEntryManageCommandsOnStoreEntityResource
	CatalogGroupManagersForOrgExecute CatalogGroupManageCommandsOnCatalogGroupResource
	CatalogGroupManagersForOrgExecute ProductSetAddCommandsOnCatalogResource
	CatalogGroupManagersForOrgExecute ProductSetManageCommandsOnProductSetResource
	CatalogManagersForOrgExecute CatalogManageCommandsOnCatalogResource
	CatalogManagersForOrgExecute StoreCategoryManageCommandsOnCatalogResource
DataBeans	CatalogGroupManagersForOrgDisplay CatalogGroupDataBeansResourceGroup
	ProductAdministratorsForOrgDisplayProductDataBeansResourceGroup

Connectivity and Notification

Table 17.

Data Resources	
	BackendOrderAdministratorsForOrgExecute BackendOrderStatusCreateCommandsOnOrderDataResource
	BackendPickPackersForOrgExecute BackendPickPackListCommandsOnFulfillmentCenterDataResource
	StoreAdministratorsForOrgExecute MessagingAdminCommandsOnStoreEntityResource
DataBeans	StoreAdministratorsForOrgDisplayMessagingDataBeans

Procurement

Table 18.

Data Resources	
	`ProcurementAdministratorsForOrgExecute ProcurementAuthenticationAndRegistrationOnOrderDataResource`
	`ProcurementShoppingCartManagersExecute ProcurementShoppingCartManageOnOrderResource`

Coupons

Table 19.

Data Resources	
	`CouponAdministratorsForOrgExecute CouponPromotionCreateCommandsOnStoreEntityResource`
	`CouponAdministratorsForOrgExecuteCouponPromotionDeleteCommands OnCouponPromotionResource`
	`RegisteredApprovedUsersExecute CouponDeleteCommandsOnCouponWalletResource`
	`RegisteredApprovedUsersExecute CouponRedemptionCommandsOnCouponWalletResource`
	`StoreAdministratorsForOrgExecute ScheduledCouponCmdsOnStoreResource`
DataBeans	`CouponAdministratorsForOrgDisplayECouponPromotionListBeans`

Customer Profiling

Table 20.

Data Resources	
	`CustomerProfileEditorsForOrgExecute SegmentManageCommandsOnStoreEntityResource`
DataBeans	`CustomerProfileEditorsForOrgDisplay SegmentationDatabeansResourceGroup`

Discounts

Table 21.

Data Resources	
	`DiscountAdministratorsForOrgExecute DiscountAssociateCommandsOnCalculationCodeResource`
	`DiscountAdministratorsForOrgExecute DiscountCreateCommandsOnStoreEntityResource`
	`DiscountAdministratorsForOrgExecute DiscountDeployCommandsOnCalculationCodeResource`
DataBeans	`DiscountViewersForOrgDisplayDiscountDatabeans`

Inventory Management

Table 22.

Data Resources	
	ExpectedInventoryManagersForOrgExecute InventoryManageCommandsOnStoreEntityResource
	FulfillmentCenterManagersForOrgExecute FulfillmentCenterCreateCommandsOnOrganizationResource
	FulfillmentCenterManagersForOrgExecute FulfillmentCenterManageCommandsOnFulfillmentResource
	InventoryAdjustersForOrgExecute InventoryAdjustCommandsOnStoreEntityResource
	PickBatchInventoryManagersForOrgExecuteReleaseReadyShipCommands OnFulfillmentCenterResource
	PickPackGeneratorsForOrgExecute PickPackGenerateCommandsOnFulfillmentCenterResource
	ReturnReasonsManagersForOrgExecute ReturnReasonsCommandsOnStoreEntityResource
	VendorInventoryManagersForOrgExecute VendorCreateCommandsOnStoreEntityResource
	VendorInventoryManagersForOrgExecute VendorManageCommandsOnVendorResource
DataBeans	StoreAdministratorsForOrgDisplay OrderFulfillmentStatusDataBeansResourceGroup

Scheduled Inventory

Table 23.

Data Resources	
	StoreAdministratorsForOrgExecute InventoryScheduledCommandsOnStoreEntityResource

Inventory Management

Table 24.

DataBeans	
	ExpectedInventoryManagersForOrgDisplay ExpectedInventoryDataBeansResourceGroup
	FulfillmentCenterManagersForOrgDisplay FulfillmentCenterDataBeansResourceGroup
	PickBatchInventoryManagersForOrgDisplay PickBatchInventoryDataBeansResourceGroup
	ProductFindInventoryManagersForOrgDisplay ProductFindInventoryDataBeansResourceGroup
	ReceiverOrderManagersForOrgDisplay ReceiverOrderManagementDataBeansResourceGroup
	ReturnReasonsManagersForOrgDisplay ReturnReasonsOrderManagementDataBeansResourceGroup

Table 24. (continued)

	ReturnsAdminOrderManagersForOrgDisplay ReturnsAdminOrderManagementDataBeansResource
	SuperUserOrderManagersForOrgDisplay SuperUserOrderManagementDataBeansResourceGroup
	VendorInventoryManagersForOrgDisplay VendorInventoryDataBeansResourceGroup

Order Management

Table 25.

Data Resources	
	CustomerOrderManagersExecute CustomerServiceCustomerWriteCommandsOnUserResource
	CustomerOrderManagersForDefaultOrgExecute CustomerServiceCustomerWriteCommandsOnUse
	CustomerOrderManagersForOrgExecute CustomerServiceOrderCreateCommandsOnStoreEntityResource
	CustomerOrderManagersForOrgExecute CustomerServiceOrderWriteCommandsOnOrderResource
	CustomerOrderManagersForOrgExecute CustomerServiceReturnCreateCommandsOnStoreEntity
	CustomerOrderManagersForOrgExecute CustomerServiceReturnWriteCommandsOnRMAResource
DataBeans	CustomerOrderManagersDisplay CustomerUserManagementDatabeans
	CustomerOrderManagersForDefaultOrgDisplay CustomerUserManagementDatabeans
	CustomerOrderManagersForOrgDisplay CustomerOrderManagementDatabeans
	LogisticsManagersForOrgDisplay OrdersAndReturnsListsDatabeans
	ReturnsManagersForOrgDisplayReturnsListsDatabean
	UserOrderManagersDisplayUserDatabeans
	UserOrderManagersForDefaultOrgDisplayUserDatabeans

Payment

Table 26.

Data Resources	
	AccountAdministratorsForOrgExecute AccountManageCommandsOnAccountResource
	AccountManagersForOrgExecute AccountCreateCommandsOnOrganizationResource
	AccountViewersForOrgExecute PaymentSummaryGenerateCommandsOnAccountResource
	AccountViewersForOrgExecute StorePaymentAdminCommandsOnStoreEntityResource

Table 26. (continued)

	`AllUsersExecutePaymentOrderWrite CommandsOnOrderResource`

Administration console pages for editing policies, access groups, resource groups, and action groups

Table 27.

Data Resources	
	`DescendantStoreAdministratorsExecute ACViewPoliciesForOrgActionsOnOrganizationResource`
	`StoreAdministratorsForOrgExecute ACPolicyCreateCommandsOnOrganizationResource`
	`StoreAdministratorsForOrgExecute ACPolicyEditCommandsOnACPolicyResource`
	`StoreAdministratorsForOrgExecute ACViewApplicablePoliciesActionsOnOrganizationResource`
	`StoreAdministratorsForOrgExecute ACViewPoliciesForUpdateActionsOnOrganizationResource`
DataBeans	`StoreAdministratorsForOrgExecute UserGroupSearchViews`

Product Advisor

Table 28.

DataBeans	
	`ProductAdvisorStatisticiansForOrgDisplay ProductAdvisorStatisticsDatabeans`
	`SalesAssistantStatisticiansForOrgDisplay SalesAssistantStatisticsDatabeans`

RFQ

Table 29.

Data Resources	
	`RFQAdministratorsAdministerRFQs`
	`RFQAdministratorsManageRFQResponses`
	`RFQBuyersEvaluateRFQResponsesForRFQsTheyOwn`
	`RFQBuyersForOrgExecuteRFQCreate CommandsOnStoreEntityDataResourceGroup`
	`RFQBuyersManageRFQResourcesTheyOwn`
	`RFQBuyersManageRFQResponsesForRFQsTheyOwn`
	`RFQSalesManagersExecuteRFQResponse ManageCommandsOnRFQResponseResource`
	`RFQSalesManagersForOrgCreateRFQResponse`
DataBeans	`RFQBuyersDisplayRFQDataBeanResourceGroupTheyOwn`

Table 29. (continued)

	RFQBuyersDisplayRFQResponseDataBeans ViewabletoRFQOwnerResourceGroup
	RFQSalesViewersDisplayRFQDataBeanResourceGroup
	RFQSalesViewersDisplayRFQResponseDataBeanResourceGroup

Rules

Table 30.

Data Resources	
	StoreAdministratorsForOrgExecutePersonalization RuleServiceAdministrationCommandsOnStoreEntityResource
DataBeans	StoreAdministratorsForOrgDisplay PersonalizationRuleServiceAdministrationDataBeanResource

Scheduler

Table 31.

Data Resources	
	StoreAdministratorsForOrgExecute ScheduledJobManageCommandsOnStoreEntityResource
	StoreAdministratorsForOrgExecute ScheduledJobManageCommandsOnUserResource
DataBeans	StoreAdministratorsForOrgDisplay SchedulerDataBeansResourceGroup

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described 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:

IBM Director of Licensing IBM Corporation North Castle Drive Armonk, NY 10504-1785 U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created

Copyright License

Trademarks

Printed in U.S.A.