BEA WebLogic Server®

Securing WebLogic Resources

Version 9.1
Revised: February 14, 2006

Copyright

Restricted Rights Legend

Trademarks and Service Marks

Contents

4. Options for Securing Web Application and EJB Resources

5. Security Policies

6. Users, Groups, And Security Roles

Index

CHAPTER 1

Introduction and Roadmap

The WebLogic Security Service combines several layers of security features to prevent unauthorized access to your WebLogic Server® domains. This document describes the security resource feature, which uses roles and policies to determine who can access entities in a domain. The security resource feature fulfills the same function as the familiar Access Control List (ACL), but offers an improvement over ACLs: an ACL is static while roles and policies specify conditions under which users can access resources, and these conditions are evaluated at runtime.

The following sections describe the content and organization of this document:

Document Scope and Audience

This document contains information that is useful for security architects and security administrators who are designing a security strategy for resources within a WebLogic Server domain. It includes information about resource types, options for securing Web applications and EJBs, different types of security roles and policies, and the components of a role and policy.

It is assumed that the reader is familiar with J2EE security and the other features of the WebLogic Security Service.

The information in this document is relevant during the design and development phases of a software project. This document does not address production phase administration topics. For links to WebLogic Server documentation and resources related to these topics, see "Related Information" on page 1-2.

Guide to This Document

The document is organized as follows:

- This chapter, Introduction and Roadmap, introduces the organization of this guide.
- Chapter 2, "Understanding WebLogic Resource Security," introduces terms and concepts, provides a workflow summary, and outlines the main steps for securing WebLogic resources.
- Chapter 3, "Types of WebLogic Resources," describes the different types of WebLogic resources that can be secured using the WebLogic Server Administration Console.
- Chapter 4, "Options for Securing Web Application and EJB Resources," describes options for securing EJB and Web application resources using deployment descriptors and/or the WebLogic Server Administration Console.
- Chapter 6, "Users, Groups, And Security Roles," describes users and groups who access WebLogic resources, including WebLogic Server default groups. Also describes scoped security roles and global security roles, including WebLogic Server default global roles. A final section describes the components of a security role.
- Chapter 5, "Security Policies," describes security policies, including WebLogic Server default security policies. Also describes the components of a security policy.

Related Information

Other WebLogic Server documents that may be of interest to security administrators wanting to secure WebLogic resources are:

- *Understanding WebLogic Security*—Summarizes the features of the WebLogic Security Service, including an overview of its architecture and capabilities. It is the starting point for understanding WebLogic security.
- *Securing WebLogic Server*—Describes how to ensure that security is comprehensively configured for a WebLogic Server® installation, including information about security providers, identity and trust and SSL.

- Secure WebLogic Resources in *Administration Console Online Help*—Provides step-by-step instructions for using the WebLogic Server Administration Console to complete the tasks that this document describes.

These documents provide additional information about specific resource types:

- "Securing Web Applications," "Securing Enterprise JavaBeans (EJBs)" and "Using Java Security to Protect WebLogic Resources" in *Programming WebLogic Security.*
- "Configuring Access Control" in *Programming WebLogic jCOM* (COM resources).
- "Security" in *Programming WebLogic Resource Adapters* (EIS resources).
- "Configuring Security" in *Programming WebLogic Web Services* (Web Services resources).

Tutorials and Samples

Additional security documents are listed on the Sample Application Examples and Tutorials page.

New and Changed Features

See What's New in WebLogic Server 9.1 and Introduction and Roadmap in *Securing WebLogic Resources* for WebLogic Server 9.0. WebLogic Server 9.1 contains no changes in WebLogic resources, roles, or policies. However, several new and changed features were introduced in WebLogic Server 9.0

CHAPTER 2

Understanding WebLogic Resource Security

This chapter introduces terms and concepts, provides a workflow summary, and outlines the main steps for securing WebLogic resources:

Overview of Securing WebLogic Resources

A WebLogic **resource** is an object that the WebLogic Security Service creates to represent an underlying WebLogic Server entity and to determine who can access the entity. You create a security **policy** to specify which users, groups, or roles can access a resource under a set of conditions. A security **role**, like a security group, grants an identity to a user. Unlike a group, however, membership in a role can be based on a set of conditions that are evaluated at runtime.

For example, when you deploy a Web Service, the WebLogic Security Service creates a `weblogic.security.service.WebServiceResource` object. When a client attempts to interact with your Web Service, the application container forwards the request to the WebLogic Security Service. The Security Service refers to the security policy that you defined for the Web Service's `WebServiceResource`. Then it either grants the client permission to proceed or rejects the request. See Figure 2-1.

Figure 2-1 A Resource Determines Who Can Access an Entity

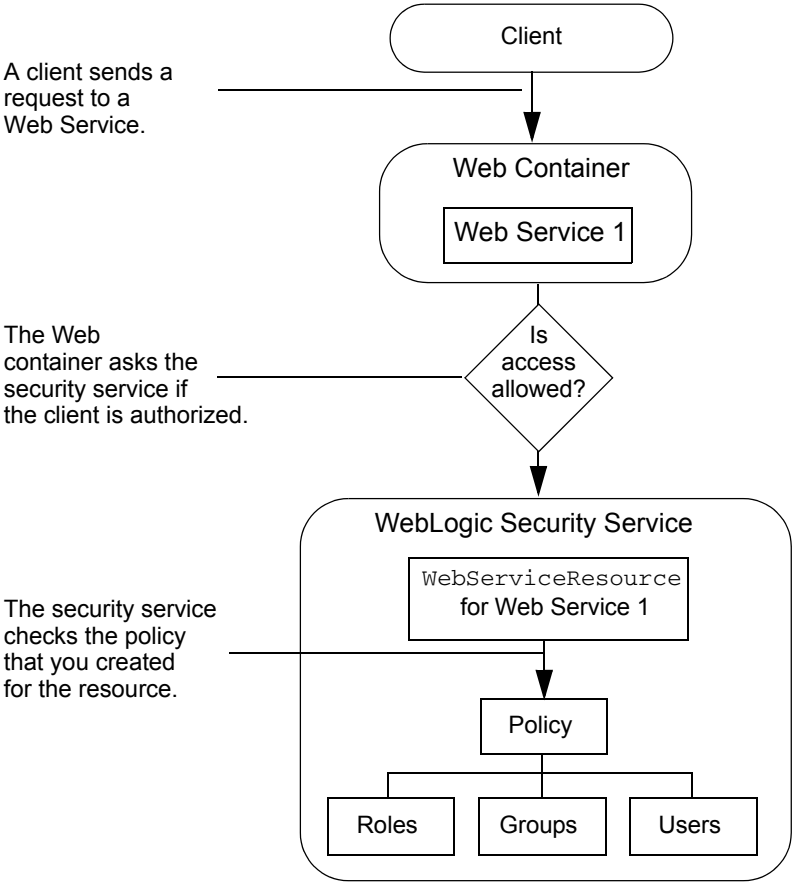
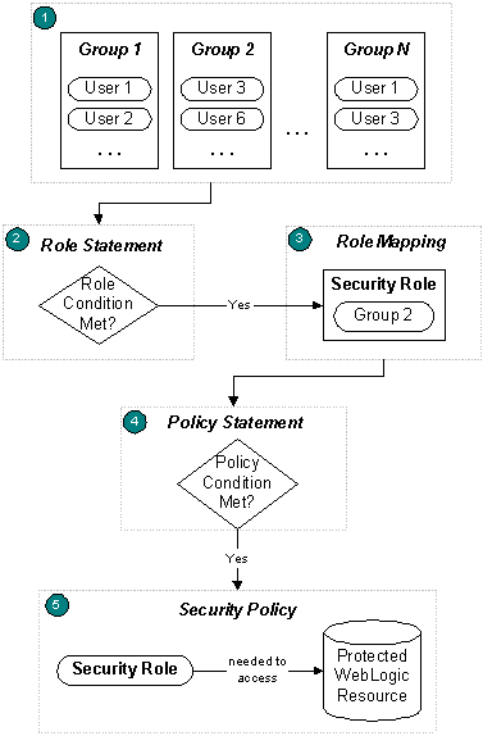


Figure 2-2 describes how you create roles and policies and how the Security Service uses them to determine whether a client can access a resource. A brief explanation follows the figure.

Figure 2-2 How a Policy Grants Access to a Resource



1. Before creating security policies and roles, Administrators statically assign users to groups, which can represent organizational boundaries. The same user can be a member of multiple groups. Figure 2-2 shows three groups with two users each. User 1 and User 3 are members of multiple groups.

   BEA recommends assigning users to groups because doing so increases efficiency for administrators who work with many users.

2. Administrators create a security role based on their organization's established business procedures. The security role consists of one or more conditions, which specify the circumstances under which a particular user, group, or other role should be granted the security role.

3. At runtime, the Security Service compares the groups against the role condition(s) to determine whether users in the group should be dynamically granted a security role. This process of comparing groups to roles is called role mapping. In Figure 2-2, Group 2 is the only group that is granted a security role.

Individual users can also be granted a security role, but this is a less typical practice.

4. Administrators create a security policy based on their organization's established business procedures. The security policy consists of one or more policy statements, each of which includes a policy condition. The policy condition specifies the circumstances under which a particular security role should be granted access to a protected WebLogic resource.
5. At runtime, the WebLogic Security Service uses the security policy and the WebLogic resource itself to determine whether access to the protected WebLogic resource should be granted. Only users who are members of the group that is granted the security role can access the WebLogic resource. In Figure 2-2, User 3 and User 6 can access the protected WebLogic resource because they are members of Group 2, and Group 2 is granted the necessary security role.

Designing Roles and Policies for WebLogic Resources: Main Steps

To design a set of roles and policies that can secure the resources in your domain:

1. List all of the resources that will be in your domain and determine which ones should be accessed only by specific users or groups.

   To see a list of all the types of resources that could be in any given domain, see "Types of WebLogic Resources" on page 3-1.

   For planning purposes, organize the resources into the following categories:

   – Server resources and administrative resources. Server resources determine who can start and stop server instances. Administrative resources determine who can complete such tasks as unlocking users who have been locked out of their accounts, uploading files (used during deployment), viewing the domain and server logs, and changing the configuration of servers, clusters, machines, and other components that are defined in the domain's configuration document (`config.xml`).

     For these tasks, WebLogic Server already provides a detailed, layered security scheme that grants different types of access to four security roles (Admin, Deployer, Operator, Monitor). For most environments, this security scheme is adequate and only requires you to assign users to the four default security roles appropriately (see step 3).

     While it is possible to modify some parts of this layered security scheme, such modifications are usually not needed and require careful planning to maintain consistency between the different layers. See "Administrative Resources" on page 3-2 and "Server Resources" on page 3-11.

- Web application resources and EJB resources, which determine who can access the Web applications and EJBs that you deploy in your domain.

  The J2EE platform already provides a standard model for securing Web applications and EJBs. In this standard model, developers define role mappings and policies in the Web application or EJB deployment descriptors.

  You can use the standard model or you can use the Administration Console to define polices and roles, which offers unified and dynamic security management. See "Options for Securing Web Application and EJB Resources" on page 4-1.

- All other resources, which determine who can access the business logic and business content in the enterprise applications and other modules that you deploy or otherwise configure for the domain.

  By default, these resources are not protected by policies; you must define policies to determine who can access them.

2. For each type of resource that you want to secure, determine if you need to create root-level policies, scoped policies, or a combination of both.

   A **root-level** policy applies to all instances of a resource type. For example, if you define a root-level policy for the Web Services resource type, then the policy will apply to all Web Services in your domain.

   A **scoped** policy applies to a specific resource instance and **overrides** a root-level policy. If the resource contains child resources, then the scoped policy applies to all children as well. For example, if you create a scoped policy for a Web Service named WebService1, then the policy applies only to WebService1 and all operations of WebService1. If you have also defined a root-level policy for the Web Services resource type, WebService1 uses its scoped policy and ignores the root-level policy.

   See "Security Policies" on page 5-1.

3. Analyze your users and the resources that you want them to access. Organize them into security groups and roles as follows:

   - Add any user that you want to start and stop servers or to engage in other administrative tasks to one of the four default global roles. The WebLogic Server security scheme allows only the four global roles to perform many of these tasks.
   - For other users (that you do **not** want to access administrative or server resources but you do want to access other resources for which you have defined policies), create additional security groups and roles. Because role membership can be granted at runtime, you can place users or groups in roles based on business rules or the context of the request.

You can create **global** roles, which can be used in any policy, or **scoped** roles, which can be used only in a policy for a specific resource instance.

See "Users, Groups, And Security Roles" on page 6-1.

4. Use the Administration Console to create users, groups, roles, and policies:

   a. To create the users and groups, see Manage Users and Groups in *Administration Console Online Help*.

   b. To create security roles, see Manage Security Roles in *Administration Console Online Help*.

   c. To create security policies, see Manage Security Policies in *Administration Console Online Help*.

Best Practices: Conditionalize Policies or Conditionalize Roles

Because both roles and policies can evaluate a set of conditions at runtime, you should consider which parts of your security data should be static and which should be dynamic. For example, you might want some policies to always allow one specific role to access a resource, and then you use conditions in the role's definition to move users in and out of the roles as needed. In other cases, you might want a static role definition and create a policy that allows access to different roles at different times of the day.

As a general guideline, if you base the authorization decision on the resource instead of the entities (roles) who can access the resource, you would add conditions to the security policy. If you base authorization on who can access the resource, then you would add conditions to the security role.

For an example of authorization based on who can access the resource, consider a manager who is going on vacation. You can temporarily place a user in a `Manager` security role. Dynamically granting this security role means that you do not need to change or redeploy your application to allow for such a temporary arrangement. You simply specify the hours between which the temporary manager should have special privileges. Further, you do not need to remember to revoke these special privileges when the actual manager returns as you would if you temporarily added the user to a management group.

Best Practices: Configure Entitlements Caching When Using WebLogic Providers

The WebLogic Authorization provider (`DefaultAuthorizer`) and the WebLogic Role Mapping provider (`DefaultRoleMapper`) improve performance by caching the roles, predicates, and resource data that they look up. If you modify your realm to use these WebLogic providers, you can configure the maximum number of items that they store in the caches.

Note: By default, security realms in newly created domains include the XACML Authorization and Role Mapping providers. The XACML providers use their own cache, but this cache is not configurable. WebLogic Server also includes the WebLogic Authorization provider (`DefaultAuthorizer`) and the Role Mapping provider (`DefaultRoleMapper`), which use a proprietary policy language. The `DefaultAuthorizer` and `DefaultRoleMapper` providers are the only BEA provider with configurable caches of entitlement data.

By default, the Weblogic Authorization and Role Mapping providers store the following number of items in each cache:

- 2000 items in the roles cache

  This cache contains the name of each role that has been looked up and the policy that protects it.

- 200 items in the predicates cache

  This cache contains each predicate that the WebLogic entitlements engine has looked up.

- 5000 items in the resources cache

  This cache contains the name of each resource that has been looked up and the policy that protects it.

If a cache exceeds its maximum size, the WebLogic entitlements engine removes the least recently used (LRU) item from the cache.

If the applications on a WebLogic Server instance use more than 2000 roles or 5000 resources, consider increasing the cache sizes. (The WebLogic providers include less than 50 predicates, so there is no need to increase the size of this cache.)

To change the maximum number of items that a cache contains, pass one of the following system properties in the `java` startup command for a WebLogic Server instance:

- `-Dweblogic.entitlement.engine.cache.max_role_count=`*`max-roles`*

where *max-roles* is the maximum number of roles that you want to cache.

- `-Dweblogic.entitlement.engine.cache.max_predicate_count=`*max-predicates*

  where *max-predicates* is the maximum number of predicates that you want to cache.

- `-Dweblogic.entitlement.engine.cache.max_resource_count=`*max-resources*

  where *max_resource_count* is the maximum number of resources that you want to cache.

By default, the WebLogic providers add items to the cache as they use them. With this configuration, the initial lookup of entitlement data takes longer than subsequent lookups. You can, however, decrease the amount of time needed for an initial lookup by configuring a WebLogic Server instance to load the caches during its startup cycle. To do so, pass the following system property to the server's `java` startup command:

- `-Dweblogic.entitlement.engine.cache.preload=true`

For example:

```
java -Dweblogic.entitlement.engine.cache.max_role_count=6001
     -Dweblogic.entitlement.engine.cache.max_resource_count=3001
     -Dweblogic.entitlement.engine.cache.preload=true
      weblogic.Server
```

CHAPTER 3

Types of WebLogic Resources

A WebLogic **resource** is an object that the WebLogic Security Service creates to represent an underlying WebLogic Server entity and to determine who can access the entity. You create a security **policy** to specify which users, groups, or roles can access a resource under a set of conditions.

The following sections describe the types of resources that you can secure using the WebLogic Server Administration Console:

- "Work Context Resources" on page 3-14

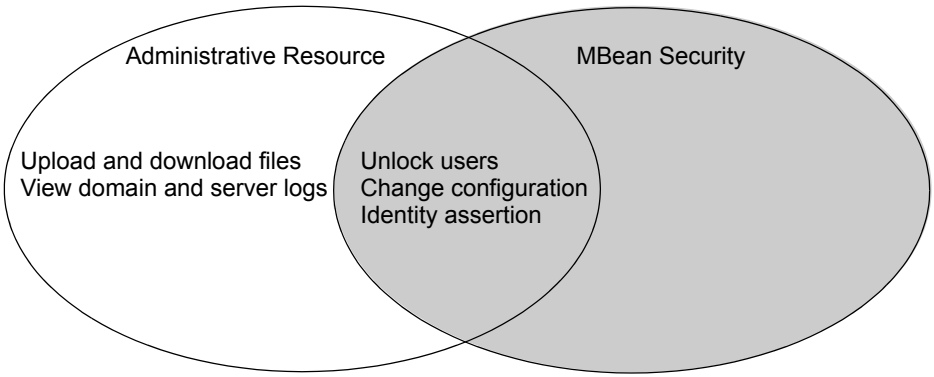
For information about APIs for WebLogic resources, see the `weblogic.security.service` package in the *WebLogic Server API Reference*.

Administrative Resources

Administrative resources determine who can complete such tasks as uploading files (used during deployment), viewing the domain and server logs, unlocking users who have been locked out of their accounts, and changing the configuration of servers, clusters, machines, and other components that are defined in the domain's configuration document (`config.xml`).

For many of these tasks, users must first be authorized by an additional security layer called the MBean security layer (see Figure 3-1).

Figure 3-1 Administrative Resource and MBean Security Layers Overlap



The following sections describe the scheme for securing administrative activity in a domain:

- "MBean Security Layer" on page 3-2
- "Administrative Resource Layer" on page 3-4
- "Maintaining a Consistent Security Scheme" on page 3-6

MBean Security Layer

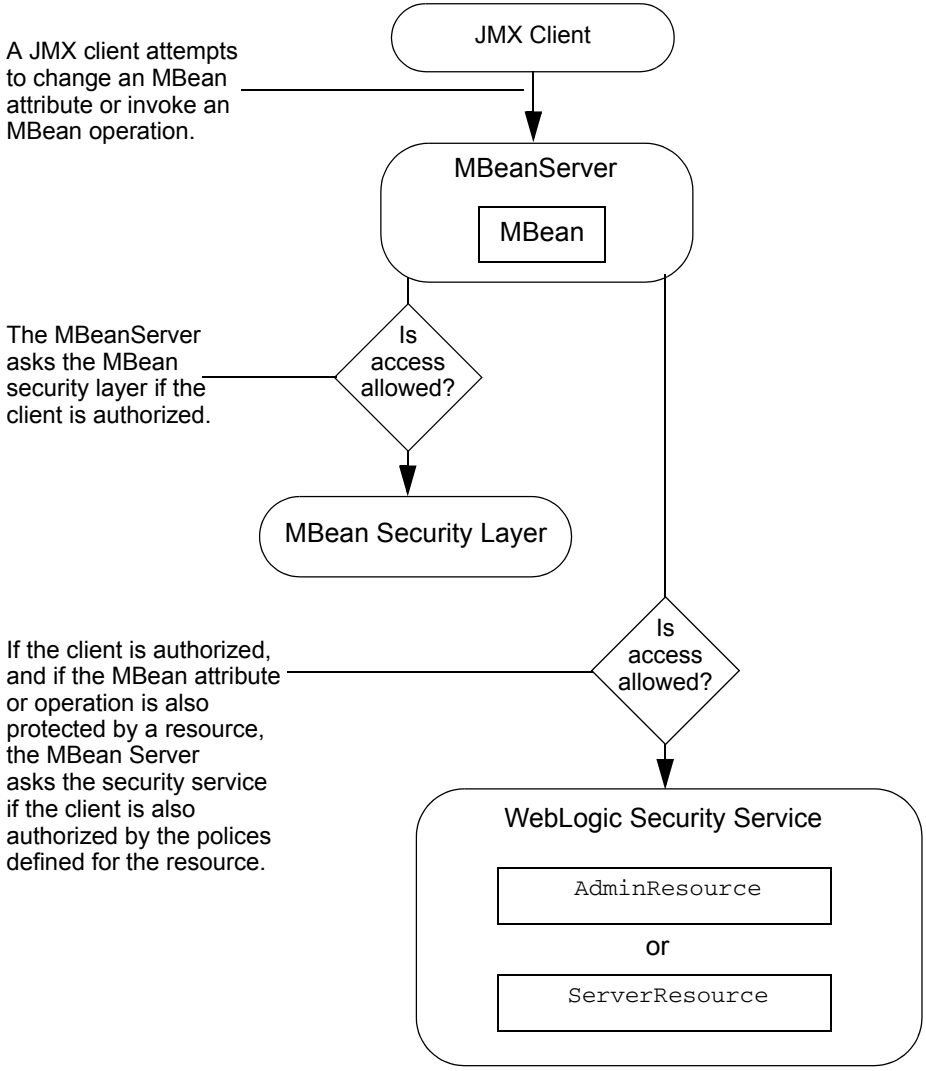
WebLogic Server uses managed beans (MBeans) in the implementation of its management system. Almost all administrative tasks require you to invoke an MBean operation or modify an MBean attribute using a Java Management Extensions (JMX) client. For example, the Administration Console is a JMX client. If you use it to change the value of a server's listen port,

the Administration Console changes the value of an MBean attribute. The WebLogic Scripting Tool is also a JMX client. For more information, see Understanding WebLogic Server MBeans in *Developing Custom Management Utilities with JMX*.

To secure access to its MBeans, WebLogic Server requires JMX clients to be in one of the four default global roles: Anonymous, Admin, Deployer, Operator (see "Default Global Roles" on page 6-4). In most cases, only the Admin role can modify MBean attributes, though there are exceptions. To see a detailed description of which roles can modify WebLogic Server MBeans, see Security Roles for MBeans in the *WebLogic Server MBean Reference*. The security requirements for MBeans are immutable; you cannot change which roles can invoke MBean operations or change attributes.

When a JMX client attempts to invoke an operation or change an attribute that is also secured by an administrative resource, the client must also satisfy the policies defined in an administrative resource (see Figure 3-2).

Figure 3-2 MBean Security Layer Checks with Security Service When Needed



Administrative Resource Layer

Table 3-1 describes the administrative activities that an administrative resource protects. Some of the protections overlap with the protections in the MBean security layer. For these overlaps,

the default policies in the administrative resource simply duplicate the protections in the MBean security layer.

Table 3-1 Activities And Default Policies For Administrative Resource

Administrative Activities	Default Policy Allows These Roles	Overlap with MBean Security?
Upload files for deployment.	`Admin, Deployer`	No
Control access to these methods in the file download servlet: • `ALL methods` • `wl_component_request` • `wl_ear_resource_request` • `ear_request` • `wl_xml_entity_request` • `wl_jsp_refresh_request` • `file` • `wl_init_replica_request` • `wl_file_realm_request` • `wl_managed_server_independence_request` **Note:** The file download servlet is used internally by WebLogic Server. BEA recommends that you do not modify the default policies for any of its methods. They are listed here only for completeness.	`Admin, Operator`	No
View domain and server logs through the Administration Console.	`Admin, Deployer, Operator, Monitor`	Yes
Enable applications to use identity assertion. The default policy for this activity specifies that an application must supply credentials for a user who is in the `Admin` role before it can successfully invoke the `Authentication.assertIdentity()` API. See `weblogic.security.services.Authentication` in the *WebLogic Server API Reference*.	`Admin`	No

Table 3-1 Activities And Default Policies For Administrative Resource

Administrative Activities	Default Policy Allows These Roles	Overlap with MBean Security?
Unlock users who have been locked out of their accounts.	`Admin`	Yes
Change the configuration of an active domain. **Note:** BEA recommends that you not add or remove roles for this policy. Modifying this policy could result in inconsistencies between the Administration resource security layer and the MBean security layer. If you need to change the default settings, modify the definition of the roles.	`Admin, Deployer, Operator, Monitor`	Yes

Maintaining a Consistent Security Scheme

The MBean security layer and the default configuration of groups, global roles, and security policies on Administrative and Server resources work together to create a consistent security scheme. You can, however, make modifications to that limit access in ways that you do not intend. Make sure that any modifications you make to the default security settings do not prevent a user from being authorized by both the MBean security layer and the policies on the Administrative and Server resources.

To keep MBean protections synchronized with security policies, consider taking the following actions when you create or modify a security policy:

- Always include the `Admin` and `Operator` global roles in policies for Server resources.

  Failure to use the `Operator` global role or a security role nested within this default global role may result in inconsistent behavior by the WebLogic Security Service.

- For a security policy on a deployable resource (such as an Web application or EJB module, Connector module, or startup/shutdown class), use the `Deployer` global role.

Application Resources

An Application resource is a type of WebLogic resource that represents an enterprise application packaged as an EAR (Enterprise Archive). Similar to the Web application resource (see "Web Application Resources" on page 3-14) the hierarchy of an application resource is a mechanism for containment, rather than a type hierarchy. You secure an Application resource when you want to protect multiple WebLogic resources that *constitute* the EAR; for example, Web application,

EJB, and Web Service resources. In other words, securing an enterprise application causes all WebLogic resources within that application to inherit its security configuration.

You can also secure, on an individual basis, the WebLogic resources that constitute an EAR. Securing a resource by both means causes the individual security configuration to override the security configuration inherited from the Enterprise Application for that WebLogic resource.

COM Resources

WebLogic jCOM is a software bridge that allows bidirectional access between Java/J2EE objects deployed in WebLogic Server, and Microsoft ActiveX components available within the Microsoft Office family of products, Visual Basic and C++ objects, and other Component Object Model/Distributed Component Object Model (COM/DCOM) environments.

A COM resource is a specific type of WebLogic resource that is designed as a program component object according to Microsoft's framework. To secure COM components accessed through BEA's bi-directional COM-Java (jCOM) bridging tool, you create security policies and security roles for packages containing multiple COM classes, or for individual COM classes.

For related information, see the "Configuring Access Control" section of *Programming WebLogic jCOM*.

Enterprise Information Systems (EIS) Resources

An EIS resource is a system-level software driver used by an application server, such as WebLogic Server, to connect to an Enterprise Information System. BEA supports resource adapters developed by EIS vendors and third-party application developers. Resource adapters. can be deployed in any application server supporting the applicable Sun Microsystems J2EE Platform Specification. Resource Adapters contain the Java code, and if necessary, the native components required to interact with the EIS.

To secure access to an EIS, create security policies and security roles for all resource adapters as a group, or for individual adapters.

For related information, see the "Security" section of *Programming WebLogic Resource Adapters*.

EJB Resources

An EJB (Enterprise JavaBean) resource is a specific type of WebLogic resource that is related to EJBs. To secure EJBs, you create security policies and security roles for EJB deployment modules, individual EJBs, or for individual methods on an EJB.

Because the J2EE platform standardizes EJB security in deployment descriptors, WebLogic Server integrates this standard mechanism with its Security Service to give you a choice of techniques for securing EJB resources. For more information, see Chapter 4, "Options for Securing Web Application and EJB Resources."

Java DataBase Connectivity (JDBC) Resources

A Java DataBase Connectivity (JDBC) resource is a specific type of WebLogic resource that is related to JDBC. You can secure JDBC resources that are deployed either as a service or an application. To secure JDBC database access, you can create security policies and security roles for all data sources as a group, individual data sources, and multi data sources.

JDBC Operations

When you secure an individual data source, you can choose whether to protect JDBC operations using one or more of the following administrator methods:

- `admin`—The following methods on the `JDBCDataSourceRuntimeMBean` are invoked as `admin` operations: `clearStatementCache`, `suspend`, `forceSuspend`, `resume`, `shutdown`, `forceShutdown`, `getProperties`, and `poolExists`.
- `reserve`—Applications reserve a connection in the data source by looking up the data source and then calling `getConnection`.

  Note: Giving a user the `reserve` permission enables them to execute vendor-specific operations. Depending on the database vendor, some of these operations may have database security implications.
- `shrink`—Shrinks the number of connections in the data source to the maximum of the currently reserved connections or to the initial size.
- `reset`—Resets the data source connections by shutting down and re-establishing all physical database connections. This also clears the statement cache for each connection. You can only reset data source connections that are running normally.
- `All`—An individual data source is protected by the union of the `Admin`, `reserve`, `shrink`, and `reset` administrator methods.

Note: If a security policy controls access to connections in a multi data source, access checks are performed at both levels of the JDBC resource hierarchy (once at the multi data source level, and again at the individual data source level). As with all types of WebLogic resources, this double-checking ensures that the most restrictive security policy controls access.

Note: If you use an Oracle database, you can also control access to JDBC resources using an Oracle Virtual Private Database (VPD). For more information, see "Programming with Oracle Private Virtual Databases" in *Using Third-Party Drivers with WebLogic Server*.

Java Messaging Service (JMS) Resources

A Java Messaging Service (JMS) resource is a specific type of WebLogic resource that is related to JMS. You can secure JMS resources that are deployed either as a service or an application. To secure JMS destinations, you create security policies and security roles for all destinations (JMS queues and JMS topics) as a group, or an individual destination (JMS queue or JMS topic) on a JMS server.

JMS Operations

When you secure a particular destination on a JMS server, you can protect operations on the destination. By default, destinations are not protected. This means that any valid user for a WebLogic server instance can send, receive, and browse messages on a destination. Only users defined by the Policy Condition can access control of the destination. Valid protection operations are:

- `send`—Required to send a message to a queue or a topic. This includes calls to the `MessageProducer.send()`, `QueueSender.send()`, and `TopicPublisher.publish()` methods, as well as the Messaging Bridge.
- `receive`—Required to create a consumer on a queue or a topic. This includes calls to the `Session.createConsumer()`, `Session.createDurableSubscriber()`, `QueueSession.createReceiver()`, `TopicSession.createSubscriber()`, `TopicSession.createDurableSubscriber()`, `Connection.createConnectionConsumer()`, `Connection.createDurableConnectionConsumer()`, `QueueConnection.createConnectionConsumer()`, `TopicConnection.createConnectionConsumer()`, and `TopicConnection.createDurableConnectionConsumer()` methods, as well as the Messaging Bridge and message-driven beans.
- `browse`—Required to view the messages on a queue using the `QueueBrowser` interface.
- `ALL`—Required to `send`, `receive`, and `browse` methods on a destination.

To protect operations for a destinations, do one of the following when creating Policy Conditions:

- "Select ALL Methods" on page 3-10

- "Select Individual Methods" on page 3-10

Select ALL Methods

To secure a resource for `send`, `receive`, and `browse` methods on a destination:

1. On the **Security > Policies** tab, select **ALL** from the **Methods** drop-down box.

   Note: You may need to update the Providers and Policy Conditions on this page. See "Security Policies" in *Securing WebLogic Resources*.

2. Click **Save**.

You should select the ALL method when you want to create a policy condition where users have access to the `send`, `receive`, and `browse` methods for a destination. You cannot mix ALL methods with individual methods when creating Policy Condition. If you mix ALL method protection with individual methods protection when creating your Policy Condition, users associated with ALL methods are ignored when the Policy Condition is created.

Select Individual Methods

To individually secure a resource for `send`, `receive`, or `browse` for a destination:

1. On the **Security > Policies** tab, select **send**, **receive**, or **browse** from the **Methods** drop-down box.

   Note: You may need to update the Providers and Policy Conditions on this page. See "Security Policies" in *Securing WebLogic Resources*.

2. Click **Save**.

You should select the `send`, `receive`, or `browse` methods when you want to create a policy condition where you individually associate `send`, `receive`, or `browse` methods for a destination. You cannot mix ALL methods with individual methods when creating Policy Condition. If you mix ALL method protection with individual methods protection when creating your Policy Condition, users associated with ALL methods are ignored when the Policy Condition is created.

Java Naming and Directory Interface (JNDI) Resources

A Java Naming and Directory Interface (JNDI) resource is a specific type of WebLogic resource that uses the industry-standard JNDI SPI to enable connectivity to heterogeneous enterprise naming and directory services.

A JNDI provides a common-denominator interface to many existing naming services, such as Lightweight Directory Access Protocol (LDAP) and Domain Name System (DNS). These naming services maintain a set of bindings, which relate names to objects and provide the ability to look up objects by name. JNDI allows the components in distributed applications to locate each other.

A JNDI is independent of any specific naming or directory service implementation. It supports the use of a number of methods for accessing various new and existing services. This support allows any service-provider implementation to be plugged into the JNDI framework using the standard service provider interface (SPI) conventions.

JNDI Operations

To secure access to the JNDI tree, create security policies and security roles for the entire JNDI tree, or for an individual branch of that tree. Regardless, you can protect all operations, or protect one of the following operations:

- `modify`—Whenever an application modifies the JNDI tree in any way (that is, adding, removing, changing) the current user must have permission to make the modification. This includes the `bind()`, `rebind()`, `createSubContext()`, `destroySubContext()`, and `unbind()` methods.
- `lookup`—Whenever an application looks up an object in the JNDI tree, the current user must have permission to perform the lookup. This includes the `lookup()` and `lookupLink()` methods.
- `list`—Whenever an application lists the contents of a context in JNDI, the current user must have permission to perform the listing operation. This includes the `list()` and `listBindings()` methods.

Server Resources

A Server resource determines who can control the state of a WebLogic Server server instance.

When users start server instances by directly invoking the `weblogic.Server` class in a Java command, the policy on the Server resource is the only security check that occurs. All other tasks that change the state of a WebLogic Server instance require the use of the Administration Console, WebLogic Scripting Tool, Node Manager, or some other JMX client, and therefore require users to be authorized first by an additional security layer called the MBean security layer. See "MBean Security Layer" on page 3-2.

You can create security policies that apply to all WebLogic Server instances in a domain or to individual servers. If you define a policy for an individual server, you can protect all of its life cycle operations or define individual policies for each of the following operations:

- `boot`—A user who tries to start a WebLogic Server instance, either an Administration Server or Managed Server, must have permission to do so. This action is typically initiated through a call to the `java weblogic.Server` command on the command line, by a configured start script (which in turn calls the `java weblogic.Server` command), or through the Node Manager capabilities that allow for remote start of WebLogic Server
- `shutdown`—A user who tries to shut down a running WebLogic Server instance, either an Administration Server or Managed Server, must have permission to do so. This action is typically initiated through the WebLogic Server Administration Console or the `WLST SHUTDOWN` or `FORCESHUTDOWN` commands.
- `suspend`—A user who tries to prohibit additional logins (logins other than for privileged administrative actions) to a running WebLogic Server instance, either an Administration Server or Managed Server, must have permission to do so. This action is typically initiated through the Administration Console.
- `resume`—A user who tries to re-enable non-privileged logins to a running WebLogic Server instance, either an Administration Server or Managed Server, must have permission to do so. This action is typically initiated through the Administration Console.

All server resources inherit a default security policy that gives permission to the `Admin` and `Operator` global security roles.

Note: If you enable the domain-wide administration port, then only the `Admin` role (and not `Operator`) can control the state of a WebLogic Server server instance. See Configure the domain-wide administration port in *Administration Console Online Help*.

Caution: Do not remove roles from the default security policies. Eliminating some of the existing security roles might negatively affect the functioning of WebLogic Server. However, if you like, you can make the default security policies more inclusive (for example, by adding new security roles). See "Maintaining a Consistent Security Scheme" on page 3-6.

Permissions for the weblogic.Server Command and the Node Manager

WebLogic Server provides two ways to start and shut down WebLogic Server instances (servers): the `weblogic.Server` command and the Node Manager. Because the underlying components

for the `weblogic.Server` command and the Node Manager are different, the two commands use different authorization methods.

Permissions for Using the weblogic.Server Command

The `weblogic.Server` command, which you can use to start both Administration and Managed Servers, calls methods that are protected by a security policy on the Server resource. To use this command, you must satisfy the requirements of the security policy on the Server resource.

Some `weblogic.Server` arguments set attributes for MBeans. However, because these arguments modify an MBean before the server is in the `RUNNING` state, the security policy on the Server resource, not the protection on the MBean, is the authorizer. For example, a user in the `Operator` global role can use the `-Dweblogic.ListenPort` argument to change a server's default listen port, but once the WebLogic Server instance is running, this user cannot change the listen port value.

For more information about `weblogic.Server`, see "weblogic.Server Command-Line Reference" in the *WebLogic Server Command Reference*.

Permissions for Using the Node Manager

The Node Manager uses both MBeans and the security policy on the Server resource to start a remote server.

If you configure a Node Manager on the host machine of a remote WebLogic Server instance, by default a user in the `Admin` or `Operator` global role can use the Node Manager to start the remote server.

For more information, see "Using Node Manager to Control Servers" in *Managing Server Startup and Shutdown*.

Shutting down a WebLogic Server instance involves both MBeans and the security policy on the Server resource. When a user issues a shutdown command, the server first determines whether that user is granted the `Admin` or `Operator` global role (per the MBean security layer). Then, after the MBean operations run, the server determines whether the security policy on the Server resource authorizes the user to shut down the server.

For more information about shutting down a WebLogic Server instance, see "Starting and Stopping Servers: Quick Reference" in *Configuring and Managing WebLogic Server*.

Web Application Resources

A Web application resource is a specific type of WebLogic resource accessed using the URL in a Web browser. To secure Web applications, you create security policies and security roles for a stand-alone Web application, the URL representing the Web Application in an EAR, or for individual components of a Web application. You can also secure specific HTTP methods for URL resources.

Because the J2EE platform standardizes Web application security in deployment descriptors, WebLogic Server integrates this standard mechanism with its Security Service to give you a choice of techniques for securing Web application resources. For more information, see Chapter 4, "Options for Securing Web Application and EJB Resources."

Web Service Resources

A Web Service resource is a specific type of WebLogic resource related to Web Services. To secure Web Services, you can create security policies and security roles for:

- The entire Web Service
- A subset of the operations of the Web Service
- The stateless session EJB that implements the Web Service
- A subset of the methods of the stateless session EJB

If your Web Service is implemented with a Java class, then the Web application WAR file contains the Java class files.

If the Web Service is implemented with a stateless session EJB, then the Enterprise Application EAR file contains the corresponding EJB JAR file.

Note: A Web Service can also be packaged as a stand-alone Web application WAR file if the Web Service is implemented with just a Java class. This type of packaging for a Web Service is uncommon; typically, Web Services are packaged as EAR files.

For more information, see "Configuring Security" in *Programming Web Services for WebLogic Server*.

Work Context Resources

Work Contexts enable J2EE developers to define properties as application context which implicitly flow across remote requests and allow downstream components to work in the context of the invoking client. Work Contexts allow developers to pass properties without including them

in a remote call. A Work Context is propagated with each remote call, allowing the called component to add or modify properties defined in the Work Context. Similarly, the calling component can access the Work Context to obtain new or updated properties.

For more information, see Best Practices for Application Design in *Programming WebLogic RMI*.

You can use the Administration Console to specify a path for a Work Context resource and secure `read`, `create`, `modify` and/or `delete` actions.

CHAPTER 4

Options for Securing Web Application and EJB Resources

The J2EE platform already provides a standard model for securing Web applications and EJBs. In this standard model, you define role mappings and policies in the Web application or EJB deployment descriptors.

Because this J2EE standard can be too inflexible for some environments, WebLogic Server offers a choice of other, more flexible models in addition to supporting the J2EE standard.

Note: If you are implementing security using JACC (Java Authorization Contract for Containers as defined in JSR 115), you must use the J2EE standard model. Other WebLogic Server models are not available and the security functions for Web applications and EJBs in the Administration Console are disabled. See Using the Java Authorization Contract for Containers in *Programming WebLogic Security*.

You choose a security model when you deploy each Web application or EJB, and your choice is immutable for the lifetime of the deployment. If you want to use a different model, you must delete and redeploy the Web application or EJB.

The following sections describe options for securing Web application and Enterprise Java Bean (EJB) resources:

Note: The instructions for EJB resources provided in this section also apply to Message-Driven Beans (MDBs).

Comparison of Security Models for Web Applications and EJBs

Each security model defines two types of behaviors for securing Web applications and EJBs: where roles and policies are defined and which URL patterns and EJB methods trigger the Security Service to perform security checks. Table 4-1 compares the models.

Table 4-1 Security Model Behaviors

This Model...	Uses Roles and Policies From...	And Performs Security Checks...
Deployment Descriptor Only (J2EE standard)	The `web.xml`, `weblogic.xml` and `ejb-jar.xml`, `weblogic-ejb-jar.xml` deployment descriptors. If roles have been defined for the application that contains the Web application or EJB, all roles are combined using a logical OR operation.	Only when clients request URLs or EJB methods that are protected by a policy in the deployment descriptor.
Custom Roles	This model uses role mappings from a role mapping provider that you configure for the security realm. You can use the Administration Console to configure the provider. Any role mappings in the deployment descriptors are ignored. The model uses the policies that are defined in the `web.xml` and `ejb-jar.xml` deployment descriptors.	Only when clients request URLs or EJB methods that are protected by a policy in the deployment descriptor.

Table 4-1 Security Model Behaviors

This Model...	Uses Roles and Policies From...	And Performs Security Checks...
Custom Roles and Policies	A role mapping provider and an authorization provider that you configure for the security realm. You can use the Administration Console to configure the providers. Any role mappings or policies in the deployment descriptors are ignored.	For all URLs and EJB methods.
Advanced	Configurable. You can configure this model to use only security data from deployment descriptors, use only the data from security providers, or import security data from deployment descriptors into the security provider databases to provide a baseline for further modifications.	Configurable.

Discussion of Each Model

The following sections describe each model and suggest scenarios under which each is appropriate.

Deployment Descriptor Only Model

This is the standard J2EE model and is therefore a widely known technique for adding declarative security to Web applications and EJBs. It uses only roles and policies defined by a developer in the J2EE deployment descriptor (DD) and the WebLogic Server DD. It requires the security administrator to verify that the security principals (groups or users) in the deployment descriptors exist and are mapped properly in the security realm.

If a developer changes roles or policies in a deployment descriptor, WebLogic Server recognizes the change as soon as you redeploy the Web application or EJB.

This model is appropriate if developers and security administrators can closely coordinate their work, both upon initial deployment of the Web application or EJB and upon subsequent redeployments. Table 4-2 shows a typical scenario:

Table 4-2 Deployment Descriptors Only: Typical Scenario

Company A, Developer	Company A, Admin/Deployer
• Maps EJBs and/or Web URLs to roles in the J2EE DD. • Maps roles to principals in the WebLogic Server DD. • Turns application over to the Admin/Deployer.	• Uses the WebLogic Server Administration Console to ensure that groups exist and are properly mapped to users.

Custom Roles Model

This security model uses policies defined in the J2EE DD and ignores any principal mappings in the WebLogic Server DD. The security administrator completes the role mappings using the Administration Console.

The model enables team members to focus on their areas of expertise. Web application and EJB developers need only to declare which URL patterns or EJB methods should be secured. Then the security administrator creates role mappings that fit within the existing hierarchy of roles and principals for a given realm.

If a developer changes policies in a deployment descriptor, WebLogic Server recognizes the change as soon as you redeploy the Web application or EJB. If an administrator changes role mappings, the changes take effect immediately without requiring a redeployment.

This model is appropriate if developers and administrators cannot closely coordinate their work or if role mappings change frequently. Table 4-3 shows a typical scenario:

Table 4-3 Customize Roles Only: Typical Scenario

Company A, ISV Developer or Company B, Developer	Company B, Admin/Deployer
• Maps EJBs/URLs to roles in J2EE deployment descriptor. • Turns application over to Admin/Deployer	• Uses the WebLogic Server Administration Console to define security roles.

Custom Roles and Policies Model

This security model offers unified and dynamic security management. It uses roles and policies that a security administrator has created using the Administration Console and ignores any roles and policies defined in deployment descriptors.

Instead of requiring developers to modify multiple deployment descriptors when organizational security requirements change, administrators can modify all security configurations from a centralized, graphical user interface. Users, groups, security roles, and security policies can all be defined using the Administration Console. As a result, the process of making changes based on updated security requirements becomes more efficient.

This model is appropriate if you require only that entire Web applications or EJBs be secured, but is less appropriate if you require fine-grained control of a large number of specific URL patterns or EJB methods. Such fine-grained control requires a developer to provide to administrators detailed information about the URL patterns or EJB methods that must be secured. If you require such fine-grained control, consider using the Custom Roles model (see "Custom Roles Model" on page 4-4).

The model also introduces a slight performance degradation because it checks permissions regardless of which URL a client requests or EJB method a client attempts to invoke.

Table 4-4 shows a typical scenario:

Table 4-4 Customize roles and Policies: Typical Scenario

Company A, Developer	Company A, Admin/Deployer
• Provides no mappings in the J2EE or WebLogic Server DD. • Turns application over to Admin/Deployer	• Uses the WebLogic Administration Console to define roles and policies for EJBs and Web applications.

Advanced Model

WebLogic Server provides this model primarily for backwards compatibility with releases prior to 9.0.

You can configure the following behaviors for this model (see "Understanding the Combined Role Mapping Enabled Setting" on page 4-10):

- Perform security checks for all URLs and EJB methods or only those that are protected in the deployment descriptors.

- (Not applicable if you configure this model to perform security checks only for URLs and EJB methods that are secured in deployment descriptors.) Use only roles and policies defined in the deployment descriptors, or use only roles and policies defined in the realm's security providers, or import security data from deployment descriptors into the realm's authorization provider or role mapping provider databases.

  BEA provides the ability to import security data for the following tasks:

  – As a step toward migrating to full security administration using the Administration Console. The import feature assumes that you want to use the WebLogic Server Administration Console exclusively to secure Web applications and EJBs, but you want to use the security data in deployment descriptors as a baseline.

  – To reinitialize security configurations for Web application and EJB resources to their original state, as specified in the deployment descriptors.

  Once the data is imported, you can use the Administration Console to modify the security data.

  Warning: Importing security data introduces risks to the integrity of your security data. Each time you import the data, the Security Service attempts to remove all associated data from the provider databases and re-imports data from the deployment descriptors. If you modified the imported security data, then your modifications could become invalid or could be removed. If you import security data, follow the recommendations in Manage Security for Web Applications and EJBs in the *Administration Console Help*.

- (Not applicable if you configure this model to use only roles and policies defined in the realm's security providers.) Combine roles in parent applications with roles in the Web application or EJB, or override roles in parent applications.

If you change the configuration of this model, the change applies to all Web applications and EJBs that use this model. For example, you configure the Advanced model to perform security checks for all URLs and methods, and then you deploy several EJBs and configure them to use the Advanced model. The EJB container will request a security check any time a client tries to invoke any method in any of the several EJBs. If you then modify the Advance model to perform security checks only for the EJB methods that are protected in deployment descriptors, then the EJB container immediately begins to request security checks only for protected methods for the several EJBs.

Understanding the Advanced Security Model

Note: This section applies only for those Web applications and EJBs that use the Advanced security model.

Three settings in the Administration Console configure the Advanced model: **Check Roles and Policies**, **When Deploying Web Applications or EJBs**, and **Combined Role Mapping Enabled**. Failure to understand these settings could result in incorrect or lost security data.

If you change the configuration of this model, the change applies to all Web applications and EJBs that use this model.

The following sections describe the settings for the Advanced security model:

- "Understanding the Check Roles and Policies Setting" on page 4-7
- "Understanding the When Deploying Web Applications or EJBs Setting" on page 4-8
- "How the Check Roles and Policies and When Deploying Web Applications or EJBs Settings Interact" on page 4-9
- "Understanding the Combined Role Mapping Enabled Setting" on page 4-10

Understanding the Check Roles and Policies Setting

The **Check Roles and Policies** setting determines whether the Security Service performs security checks for all URLs and EJB methods or only those that are protected in the deployment descriptors.

Set the value of **Check Roles and Policies** as follows:

- To perform security checks only on Web application and EJB resources that have security specified in their associated deployment descriptors (DDs), select **Web applications and EJBs Protected in DD**.

  Note: This selection is analogous to the Deployment Descriptor Only security model: the Security Service uses only roles and policies defined in a Web application or EJB's deployment descriptors.

- To perform security checks on all Web application and EJB resources, regardless of whether there are any security settings in the deployment descriptors for these WebLogic resources, select **All Web applications and EJBs**.

  Note: With this selection, you can also configure the **When Deploying Web Applications or EJBs** setting.

Understanding the When Deploying Web Applications or EJBs Setting

The **When Deploying Web Applications or EJBs** setting determines whether the Security Service ignores role and policy data in deployment descriptors or imports the data into role mapping and authorization provider databases **each time** you deploy a Web application or EJB.

Note: This setting is valid only if you have set **Check Roles and Policies** to **All Web applications and EJBs**.

Set the value of **When Deploying Web Applications or EJBs** as follows:

- To secure Web application and EJB resources using *only* the WebLogic Server Administration Console, select **Ignore Roles and Policies From DD** (Deployment Descriptors). At this point you can begin to use the Administration Console to secure the resources. See Create Scoped Security Roles and Create Policies for Resource Instances in *Administration Console Online Help.*
- To import security data from the deployment descriptors, select **Initialize Roles and Policies from DD**.

  Warning: Importing security data introduces risks to the integrity of your security data. Each time you import security data, the Security Service attempts to remove all associated security data from the provider databases and re-imports data from the deployment descriptors. If you modified the imported security data, then your modifications could become invalid or could be removed. If you import security data, follow the recommended procedures in Manage Security for Web Applications and EJBs in the *Administration Console Help*.

How the Check Roles and Policies and When Deploying Web Applications or EJBs Settings Interact

Table 4-5 shows how to achieve the behavior you want from the WebLogic Security Service using different combinations of the Check Roles and Policies and When Deploying Web Applications and EJBs settings.

Table 4-5 Interaction Between the Check Roles and Policies Setting and the When Deploying Web Applications or EJBs Setting

If you want to control security for...	and set security for Web application and EJB resources...	then set Check Roles and Policies to...	and set When Deploying Web Applications or EJBs to...
All Web application and EJB resources	using *only* the Administration Console	All Web applications and EJBs	Ignore Roles and Policies from DD
All Web application and EJB resources	by *copying* or *reinitializing* security data from the deployment descriptors into the configured Authorization and Role Mapping providers' databases when the Web application or EJB resource is deployed, then use one of the other techniques to modify security roles and security policies **Note:** Security data is copied/reinitialized *each time* the Web application or EJB resource is deployed.	All Web applications and EJBs	Initialize Roles and Policies from DD
Only on Web applications and EJB methods that are specified in the deployment descriptors (default configuration)	using *only* the deployment descriptors	Web applications and EJBs Protected in DD	--

Understanding the Combined Role Mapping Enabled Setting

The Combined Role Mapping Enabled setting determines how the role mappings in the Enterprise Application, Web application, and EJB containers interact.

WebLogic Server provides this setting for backwards compatibility with 8.x versions. For all applications initially deployed in version 9.x, the default value for this setting is "true" (enabled). For all applications previously deployed in version 8.1 and upgraded to version 9.x, the default value is "false" (disabled). If either of the following is true, consider changing the default value for Combined Role Mapping Enabled:

- You selected the Advanced security model for an 8.x application upgrade and want to use the combine role mapping behavior available in version 9.x.
- You selected the Advanced security model for a 9.x application and want to use the role mapping behavior in version 8.x.

Table 4-6 compares how this setting affects security for Web applications and EJBs:

Table 4-6 How Combined Role Mapping Affects Security for Web Applications and EJBs

When Combined Role Mapping is disabled...	When Combined Role Mapping is enabled...
Role mappings for each container are exclusive to other containers unless defined by the `<externally-defined>` descriptor element.	Application role mappings are combined with EJB and Web application mappings so that all principal mappings are included. The Security Service combines the role mappings with a logical `OR` operator.

If one or more policies in the `web.xml` file specifies a role for which no role mapping exists in the `weblogic.xml` file, the Web application container assumes that the undefined role is the name of a principal. It therefore maps the assumed principal to the role name. For example, if the `web.xml` file contains the following stanza in one of its policies:
`<auth-constraint>`
`<role-name>PrivilegedUser</role-name>`
`</auth-constraint>`
but the `weblogic.xml` file has no role mapping for PrivilegedUser, then the Web application container creates an in-memory mapping that is equivalent to the following stanza:
`<security-role-assignment>`
`<role-name>PrivilegedUser</role-name>`
`<principal-name>`
`PrivilegedUser`
`</principal-name>`
`</security-role-assignment>`
	If one or more policies in the `web.xml` file specifies a role for which no mapping exists in the weblogic.xml file, the Web application container creates an empty map for the undefined role (that is, the role is explicitly defined as containing no principal). Therefore, no one can access URL patterns that are secured by such policies.

Role mappings for EJB methods must be defined in the `weblogic-ejb-jar.xml` file. Role mappings defined in the other containers are not used unless defined by the `<externally-defined>` descriptor element.	If one or more policies in the `ejb-jar.xml` file specifies a role for which no mapping exists in the `weblogic-ejb-jar.xml` file, the EJB container creates an empty map for the undefined role (that is, the role is explicitly defined as containing no principal). Therefore, no one can access methods that are secured by such policies.

Usage Examples

The following examples show the differences in role mapping behaviors depending on whether Combined Role Mapping is enabled or disabled.

Example for EAR, WAR and EJB

MyAppEar contains MyAppWAR which contains MyEJB. Role to Principal mappings (p1 and p2) are as follows:

- EAR descriptor, myRole = p1
- WAR descriptor, myRole = p2
- EJB-JAR descriptor, myRole = empty

When Combined Role Mapping is enabled, the role mappings would be:

- For the Ear container, myRole maps to p1.
- For the WAR container, myRole maps to p1 or p2.
- For the EJB container, myRole maps to p1.

When Combined Role Mapping is disabled, the role mappings would be

- For the Ear container, myRole maps to p1.
- For the WAR container, myRole maps to p2.
- For the EJB container: Must be externally-defined or the deployment fails.

Example for EAR and WAR

MyAppEar contains MyAppWAR. Role to Principal mappings are as follows:

- In MyAppEAR descriptor, myRole = p1
- In MyAppWAR descriptor, myRole = (none defined)

When Combined Role Mapping is enabled, the role mappings would be:

- For the Ear container, myRole maps to p1.
- For the WAR container, myRole maps to p1.

  The mapping is the same because of the combined role behavior.

When Combined Role Mapping is disabled, the role mappings would be

- For the Ear container, myRole maps to p1.
- For the WAR container, myRole maps to MyRole.

  The mapping is the same because if there is no mapping defined for the Web application, WebLogic Server copies the EAR mapping to the WAR mapping.

Securing Web Applications and EJBs

You choose a security model when you deploy each Web application or EJB, and your choice is immutable for the lifetime of the deployment. If you want to use a different model, you must delete and redeploy the Web application or EJB.

For information on using the Administration Console to deploy applications, choose a security model, modify roles and polices, and complete other related tasks, see Manage Security for Web Applications and EJBs in the *Administration Console Help.*

If you plan to use deployment descriptors to secure Web applications or EJBs, see "Using Declarative Security With Web Applications" and "Using Declarative Security With EJBs" in *Programming WebLogic Security*.

CHAPTER 5

Security Policies

A security policy specifies who can access a WebLogic Server resource. You can create simple policies, such as "allow access if user is in Admin role," or more complex policies, such as "between the hours of 8 and 5, allow access if user is in Admin role."

The following sections describe the features and functions of security policies:

Security Policy Granularity and Inheritance

Security policies are always scoped to a WebLogic resource, but because WebLogic resources are hierarchical, the level at which you define a security policy is up to you.

You can create a **root-level policy** that applies to all instances of a specific resource type (all new instances of that WebLogic resource inherit the security policy). For example, you can define a root level policy that will be inherited by all JMS resources in your domain. This inheritance of security policies means that you can secure multiple WebLogic resources efficiently. WebLogic

Server provides a set of default, root-level policies. See "Default Root Level Security Policies" on page 5-3.

You can also create a policy that applies to a specific resource instance. If the instance contains other resources, the policy will apply to the included resource as well. For example, you can create a policy for an entire enterprise application (EAR), an EJB JAR containing multiple EJBs, a particular EJB within that JAR, or a single method within that EJB.

The policy of the narrower scope overrides the policy of a broader scope. For example, if you create a security policy for an EAR and a policy for an EJB that is in the EAR, the EJB will be protected by its own policy and will ignore the policy for the EAR.

Security Policy Storage and Prerequisites for Use

Security policies for all resources other than Web Application resources and EJB resources are always stored in the security provider database of the Authorization provider that is configured in the default (active) security realm. The security realm that WebLogic Server provides stores policies in the embedded LDAP server.

For Web Application resources and EJB resources, the location of policies depends on the following:

- If you implement security using JACC (Java Authorization Contract for Containers as defined in JSR 115), the policies are stored in the Web application or EJB deployment descriptors.
- If you use the DDOnly model to secure a Web application or EJB, the policies are stored in the deployment descriptors.
- If you use a security model that ignores the policies in the descriptors, then the Authorization provider determines where the policies are stored. The security realm that WebLogic Server provides stores policies in the embedded LDAP server.
- If you use the Advanced security model, the location of policies depends on how you configure the model.

See "Options for Securing Web Application and EJB Resources" on page 4-1.

Each user or group that you add to a security policy must be defined in the security provider database of the Authentication provider that is configured in the active security realm. Each role that you add must be defined in the security provider database of the Role Mapping provider that is configured in the active security realm. The security realm that WebLogic Server provides is

configured to use the WebLogic Authentication and WebLogic XACML Role Mapping providers, which store users, groups, and roles in the embedded LDAP server.

For more information about the WebLogic Authentication, Authorization, and Role Mapping providers, see "The WebLogic Security Providers" in *Introduction to WebLogic Security*.

Default Root Level Security Policies

A **root level policy** is inherited by all instances of a specific resource type. Table 5-1 describes the default root level policies that are defined in the security realm that WebLogic Server installs. For information about the roles and groups that are named in these policies, see "Users, Groups, And Security Roles" on page 6-1.

Note: You can access root level policies in the Administration Console. See Create root-level policies in *Administration Console Online Help.*

Table 5-1 Default Security Policies for WebLogic Resources

WebLogic Resource	Security Policy
Administrative resources	Default global role: `Admin`
Application resources	None
EIS (Resource Adapter) resources	Default group: `Everyone`
EJB resources	Default group: `Everyone`
COM resources	None
JDBC resources	Default group: `Everyone`
JNDI resources	Default group: `Everyone`
JMS resources	Default group: `Everyone`
Server resources	Default global roles: • `Admin` • `Operator`
Work Context	Default group: `Everyone`
URL resources	Default group: `Everyone`
Web Services resources	Default group: `Everyone`

Caution: Do not modify the default root level policies for Administrative and Server resources to make them more restrictive. Eliminating some of the existing security roles might negatively impact the functioning of WebLogic Server. However, if you like, you can make the default security policies more inclusive (for example, by adding new security roles). See "MBean Security Layer" on page 3-2.

Security Policy Conditions

To determine who can access a resource, a policy contains one or more conditions. The most basic policy simply contains the name of a security role or a principal. For example, a basic policy might simply name the "Admin" global role. At runtime, the WebLogic Security Service interprets this policy as "allow access if user is in Admin role." You can create more complex conditions and combine them using the logical operators AND and OR (which is an inclusive OR). You can also negate any condition, which would prohibit access under the specified condition.

The WebLogic Server Authorization providers display three kinds of built-in policy conditions in the Administration Console:

Note: These sections describe the conditions that are available in realms that use the WebLogic Authorization provider or the WebLogic XACML Authorization provider. If your security realm uses a third-party Authorization provider, refer to the third-party documentation for information on its capabilities.

- "Basic Policy Conditions" on page 5-4
- "Date and Time Policy Conditions" on page 5-5
- "Context Element Policy Conditions" on page 5-6

Basic Policy Conditions

The basic policy conditions that are available in this release of WebLogic Server are:

- `User`—Allows a specific user to access the resource. For example, you might create a condition indicating that only the user `John` can access the `Deposit` EJB.
- `Group`—Allows all users or groups in the specified group to access the resource unless a User or Role condition contradicts the Group condition.
- `Role`—Allows all users or groups in the specified role to access the resource unless a User or Group condition contradicts the Role condition. For example, if you create a Role

condition that specifies “Admin” and a User condition that negates “joe”, then user joe will be denied access even if he is in the Admin role.

- `Server is in Development Mode`—Allows access if the server that hosts the resource is running in development mode. See Differences Between Domain Startup Modes in *Creating WebLogic Domains Using the Configuration Wizard*.
- `Allow access to everyone`—Allows access for all users, groups, and roles.
- `Deny access to everyone`—Prohibits access for all users, groups, and roles.
- `Element requires signature by`—Creates a condition for a security policy based on who has digitally signed an element in the SOAP request message that invokes a Web Service operation. For example, you might create a condition that says the `getBalance` operation can only be invoked if the `AccountNumber` element in the incoming SOAP request has been digitally signed by the `BankTeller` security role.

  Note: This policy condition is used only when securing Web Services and individual Web Service operations.

Date and Time Policy Conditions

When you use any of the date and time conditions, the security policy grants access to *all users* for the date or time you specify, unless you further restrict the users by adding one of the other conditions. The date and time policy conditions available in this release of WebLogic Server are:

- `Access occurs between specified hours`—Allows access during a specified time period. For example, you might create a condition granting access to users only during business hours.
- `Access occurs after`—Allows access after a specified time. For example, you might create a condition that grants access to users after the business opens or after a certain date and time.
- `Access occurs before`—Allows access before a specified time. For example, you might create a condition that grants access to users before the business closes or before a certain date and time.
- `Access occurs on specified days of the week`—Allows access on specified days. For example, you might create a condition that grants access to users on week days.
- `Access occurs on the specified day of the month`—Allows access on an ordinal day of the month. For example, you might create a condition that grants access to users only the first day of each month.

- `Access occurs after the specified day of the month`—Allows access after an ordinal day in the month. For example, you might create a condition indicating that grants access to users after the 15th day of the month.
- `Access occurs before the specified day of the month`—Allows access before an ordinal day in the month. For example, you might create a condition that grants access to users before the 15th day of the month.

Context Element Policy Conditions

You can use the context element conditions to create security policies based on the value of HTTP Servlet Request attributes, HTTP Session attributes, and EJB method parameters. WebLogic Server retrieves this information from the ContextHandler object and allows you to defined policy conditions based on the values. When using any of these conditions, it is your responsibility to ensure that the attribute or parameter/value pairs apply to the context in which you are using them. For more information, see ContextHandlers and WebLogic Resources in *Developing Security Providers for WebLogic Server*.

The context element role conditions available in this release of WebLogic Server are:

- `Context element defined`—Allows access based on the existence of a specified attribute or parameter.
- `Context element's value equals a numeric constant`—Allows access based on a specified attribute or parameter's number value (or string representing a double number) being equal to a specified `double` number.
- `Context element's value is greater than a numeric constant`—Allows access based on a specified attribute or parameter's number value (or string representing a double number) being greater than a specified `double` number.
- `Context element's value is less than a numeric constant`—Allows access based on a specified attribute or parameter's number value (or string representing a double number) being less than a specified `double` number
- `Context element's value equals a string constant`—Allows access based on a specified attribute or parameter's string value being equal to a specified string.

Protected Public Interfaces

The WebLogic Server Administration Console, the WebLogic Scripting Tool (WLST), and MBean APIs are secured using the default security policies, which are based on the default global roles and default groups described in Table 6-2, "Default Global Roles, Privileges, and Default

Group Assignments," on page 6-4. Therefore, to use the Administration Console, a user must belong to one of these default groups or be granted one of these global roles. Additionally, administrative operations that require interaction with MBeans are secured using the MBean protections described in "MBean Security Layer" on page 3-2. Therefore, interaction with the following protected public interfaces typically must satisfy both security schemes.

- *The WebLogic Server Administration Console*—The WebLogic Security Service verifies whether a particular user can access the Administration Console when the user attempts to log in. If a user attempts to invoke an operation for which they do not have access, they see an Access Denied error.

  For information about using this public interface, see the The WebLogic Server Administration Console in *Administration Console Online Help*.

- *The* `WebLogic Scripting Tool`—The WebLogic Scripting Tool (WLST) is a command-line scripting interface that system administrators and operators can use to monitor and manage WebLogic Server instances and domains. The WebLogic Security Service verifies whether a particular user has permission to execute a WLST command when the user attempts to invoke the command. If a user attempts to invoke an operation for which the user does not have access, WebLogic Server throws a `weblogic.management.NoAccessRuntimeException`, which developers can catch explicitly in their programs. The server sends this exception to its log file, but you can also configure the server to send exceptions to standard out.

  For information about using this public interface, see *WebLogic Scripting Tool*.

  Note: WLST is a convenience utility that abstracts the interaction with the MBean APIs (described next). Therefore, for any administrative task you can perform using WLST, you can also perform using the MBean APIs.

- *MBean APIs*—The WebLogic Security Service verifies whether a particular user has permission to access the API when the user attempts to perform an operation on the MBean. If a user attempts to invoke an operation for which the user does not have access, WebLogic Server throws a `weblogic.management.NoAccessRuntimeException`, which developers can catch explicitly in their programs. The server sends this exception to its log file, but you can also configure the server to send exceptions to standard out.

  For information about using these APIs, see Understanding WebLogic Server MBeans in *Developing Custom Management Utilities with JMX*.

Using the Administration Console to Manage Security Policies

Note: This section describes the features and functions that are available in security realms that are using the WebLogic Authorization provider or the WebLogic XACML Authorization provider. If your security realm uses a third-party Authorization provider, refer to the third-party documentation for information on how to add polices to the provider database.

You can use the WebLogic Administration Console to access WebLogic resources for creating and modifying security policies. For more information, see Manage Security Policies in *Administration Console Online Help*.

CHAPTER 6

Users, Groups, And Security Roles

The following sections describe the features and functions of users, groups, and security roles:

Overview of Users and Groups

A user is an entity that can be authenticated. A user can be a person or a software entity, such as a Java client. Each user is given a unique identity within a security realm. For efficient security management, BEA recommends adding users to groups. A group is a collection of users who usually have something in common, such as working in the same department in a company.

Default Groups

Table 6-1 lists the groups that WebLogic Server defines in the security realm that it installs. By default, if you add a user to one of these groups, you also place the user in one of the default global security roles (see "Default Global Roles" on page 6-4).

Table 6-1 Default Groups

Group Name	Membership
`Administrators`	By default, this group contains the user information entered as part of the installation process (that is, the Configuration Wizard), and the `system` user if the WebLogic Server instance is running Compatibility security. Any user assigned to the `Administrators` group is granted the `Admin` security role by default. See "Best Practices: Add a User To the Administrators Group" on page 6-3.
`Deployers`	By default, this group is empty. Any user assigned to the `Deployers` group is granted the `Deployer` security role by default.
`Operators`	By default, this group is empty. Any user assigned to the `Operators` group is granted the `Operator` security role by default.
`Monitors`	By default, this group is empty. Any user assigned to the `Monitors` group is granted the `Monitor` security role by default.
`AppTesters`	By default, this group is empty. Any user assigned to the `AppTesters` group is granted the `AppTester` security role by default.

Runtime Groups

At runtime, WebLogic Server places all users in the following groups:

- `users`. This group contains all users who have been authenticated.
- `everyone`. This group contains all anonymous users and, because it contains the `users` group, all users who have been authenticated.

Unlike the groups in Table 6-1 (or other groups that you create), you cannot add or remove users directly to these groups; WebLogic Server assigns users to them dynamically. These groups do not appear in the Administration Console's Groups tab and they are not exported with the authentication database.

Best Practices: Add a User To the Administrators Group

BEA recommends that you add at least one user to the `Administrators` group *in addition to* the user you defined at installation (using the Configuration wizard). Having at least two administrators at all times helps protect against a single admin user being locked out from a potential security breach. Also, avoid using predictable user names like “system”, “admin”, or “Administrator”.

Overview of Security Roles

A security role is an identity granted to users or groups based on specific conditions. Multiple users or groups can be granted the same security role and a user or group can be in more than one security role. Security roles are used by policies to determine who can access a WebLogic resource. (See “Security Policies” on page 5-1.)

Like a security group, a role grants an identity to a user. Security roles differ from groups as follows:

- Security roles can be scoped to specific WebLogic resources within a single application in a WebLogic Server domain. Groups, on the other hand, are always scoped to an entire WebLogic Server domain. See “Types of Security Roles: Global Roles and Scoped Roles” on page 6-3.
- Security roles are computed and granted to users or groups dynamically, based on conditions such as user name, group membership, or the time of day. Groups are static.

  The process of computing and granting roles is referred to as role mapping and occurs just before the WebLogic Security Service renders an access decision for a protected WebLogic resource. An access decision is the component of an Authorization provider that determines whether a subject has permission to perform a given operation on a WebLogic resource. (See “Access Decisions” in *Developing Security Providers for WebLogic Server*.)

Types of Security Roles: Global Roles and Scoped Roles

There are two types of security roles in WebLogic Server:

- A **global** security role can be used in any security policy. BEA provides several default global roles that you can use out of the box to secure your WebLogic resources; these are described in “Default Global Roles” on page 6-4.

  Note: If you are implementing security using JACC (Java authorization Contract for Containers as defined in JSR 115) global security roles cannot be used.

- A **scoped** role can be used only in policies that are defined for a specific instance of a WebLogic resource (such as a method on an EJB or a branch of a JNDI tree). You might never need to use scoped roles. They are provided for their flexibility and are an extra feature for advanced customers.

Default Global Roles

Table 6-2 lists the global roles that WebLogic Server defines in the security realm that it installs. The table also summarizes the access that the default security policies grant to each role and indicates which groups are in each role by default.

Caution: Do not delete these roles. They are used in the default security policies that protect most types of WebLogic resources. In addition, they are used by the MBean security layer. If you delete the Admin role, no one will be able to modify the configuration of a running domain. See "MBean Security Layer" on page 3-2.

Table 6-2 Default Global Roles, Privileges, and Default Group Assignments

Global Role	Default Policies Grant Access To...	Default Conditions Include This Group...
`Admin`	• View the server configuration, *including* the encrypted value of some encrypted attributes. • Modify the entire server configuration. • Deploy Enterprise Applications and Web application, EJB, J2EE Connector, and Web Service modules. • Start, resume, and stop servers.	`Administrators`
`Anonymous`	All users (the group `everyone`) are granted this global role. **Note:** This global role is provided as a convenience, and can be specified in the `weblogic.xml` and `weblogic-ejb-jar.xml` deployment descriptors. See weblogic.xml schema and ejb-jar deployment descriptor reference.	`everyone`

Table 6-2 Default Global Roles, Privileges, and Default Group Assignments

Global Role	Default Policies Grant Access To...	Default Conditions Include This Group...
`Deployer`	• View the server configuration, including *some* encrypted attributes related to deployment activities. • Change startup and shutdown classes, Web applications, JDBC data pool connections, EJB, J2EE Connector, Web Service, and WebLogic Tuxedo Connector components. If applicable, edit deployment descriptors. • Access deployment operations in the J2EE 1.4 Deployment Implementation (JSR-88). See Deployment Standards in *Deploying Applications to WebLogic Server*.	`Deployers`
`Operator`	• View the server configuration, *except* for encrypted attributes. • Start, resume, and stop servers.	`Operators`
`Monitor`	View the server configuration, *except* for encrypted attributes. This security role effectively provides read-only access to the WebLogic Server Administration Console, `WLST`, and MBean APIs.	`Monitors`
`AppTester`	Access applications for testing purposes that are running in Administration mode. For more information, see Administration Mode for Isolating Production Applications in *Understanding WebLogic Server Deployment*.	`AppTesters`

Security Role Conditions

To determine who is in a security role at runtime, a role contains one or more conditions. For example, a basic role might simply name the "Administrator" group. At runtime, the WebLogic Security Service interprets this policy as "place the Administrator group in this role." You can create more complex conditions and combine them using the logical operators AND and OR (which is an inclusive OR). You can also negate any condition, which would make sure that a user is not in the role. The entire collection of conditions must be true for a user or group to be granted the security role. More restrictive expressions should come later in a role statement.

The WebLogic Server Role Mapping providers display three kinds of built-in policy conditions in the Administration Console:

Note: These sections describe the conditions that are available in realms that use the WebLogic Role Mapping provider or the WebLogic XACML Role Mapping provider. If your security realm uses a third-party Role Mapping provider, refer to the third-party documentation for information on its capabilities.

Basic Role Conditions

The basic role conditions available in this release of WebLogic Server are:

- `User`—Adds the specified user to the role. For example, you might create a condition indicating that only the user `John` can be granted the `BankTeller` security role.
- `Group`—Adds the specified group to the role. For example, you might create a condition indicating that only users in the group `FullTimeBankEmployees` can be granted the `BankTeller` security role.

  As a minimum requirement, BEA recommends this role condition for more efficient security management.
- `Server is in development mode`—Adds principals to the role only when the server is running in development mode. See Differences Between Domain Startup Modes in *Creating WebLogic Domains Using the Configuration Wizard*.
- `Allow access to everyone`—Adds all users and groups to the role.
- `Deny access to everyone`—Prevents any user or group from being in the role.

Date and Time Role Conditions

When you use any of the date and time role conditions, the security role is granted to *all users* for the date or time you specify, unless you further restrict the users by adding one of the other role conditions. The date and time role conditions available in this release of WebLogic Server are:

- `Access occurs between specified hours`—Adds principals to the role only during the specified time period. For example, you might create a condition indicating that the `BankTeller` security role can only be granted to users when the bank is open.
- `Access occurs after`—Adds principals to the role only if the current time is after a specified time. For example, you might create a condition indicating that the `BankTeller`

security role can only be granted to users after the bank opens or after a certain date and time.

- `Access occurs before`—Adds principals to the role only if the current time is before a specified time. For example, you might create a condition indicating that the `BankTeller` security role can only be granted to users before the bank closes or before a certain date and time.
- `Access occurs on specified days of the week`—Adds principals to the role only on specified days. For example, you might create a condition indicating that the `BankTeller` security role can only be granted to users on week days.
- `Access occurs on the specified day of the month`—Adds principals to the role only on an ordinal day of the month. For example, you might create a condition indicating that the `BankTeller` security role can only be granted to users on the first day of each month.
- `Access occurs after the specified day of the month`—Creates a condition for a security role based on a time after an ordinal day in the month. For example, you might create a condition indicating that the `BankTeller` security role can only be granted to users after the 15th day of the month.
- `Access occurs before the specified day of the month`—Adds principals to the role only if the current day is before an ordinal day in the month. For example, you might create a condition indicating that the `BankTeller` security role can only be granted to users before the 15th day of the month.

Context Element Role Conditions

You can use the context element conditions to create security roles based on the value of HTTP Servlet Request attributes, HTTP Session attributes, and EJB method parameters. WebLogic Server retrieves this information from the ContextHandler object and allows you to defined role conditions based on the values. When using any of these conditions, it is your responsibility to ensure that the attribute or parameter/value pairs apply to the context in which you are using them. For more information, see ContextHandlers and WebLogic Resources in *Developing Security Providers for WebLogic Server*.

The context element role conditions available in this release of WebLogic Server are:

- `Context element defined`—Adds principals to the role based on the existence of a specified attribute or parameter.

- `Context element's value equals a numeric constant`—Adds principals to the role based on a specified attribute or parameter's number value (or string representing a double number) being equal to a specified `double` number.
- `Context element's value is greater than a numeric constant`—Adds principals to the role based on a specified attribute or parameter's number value (or string representing a double number) being greater than a specified `double` number.
- `Context element's value is less than a numeric constant`—Adds principals to the role based on a specified attribute or parameter's number value (or string representing a double number) being less than a specified `double` number
- `Context element's value equals a string constant`—Adds principals to the role based on a specified attribute or parameter's string value being equal to a specified string.

Using the Administration Console to Manage Users, Groups, and Roles

Note: This section describes the features that are available in realms that use the WebLogic Authentication provider and the WebLogic Role Mapping provider or the WebLogic XACML Role Mapping provider. If your security realm uses a third-party Authentication or Role Mapping provider, refer to the third-party documentation for information on its capabilities.

For information on adding users and groups to a security realm, see Manage Users and Groups in *Administration Console Online Help*.

For information on creating security roles, see Manage Security Roles in *Administration Console Online Help*.

Index