BEA WebLogic Server®

Securing WebLogic Resources

Version 9.0
Revised: July 22, 2005

Contents

4. Options for Securing EJB and Web Application Resources

Introduction and Roadmap

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

- "Document Scope" on page 1-1
- "Guide to this Document" on page 1-2
- "Related Information" on page 1-3
- "New and Changed Features In This Release" on page 1-3

Document Scope

This document describes how to use security roles and security polices to protect different types of WebLogic resources. It includes information about resource types, options for securing EJB and Web application resources, different types of security roles and policies, and the components of a role and policy.

This document should be used in conjunction with:

- *Securing WebLogic Server* to ensure that security is comprehensively configured for a WebLogic Server® deployment.
- Secure WebLogic Resources in *Administration Console Online Help*. Provides instructions for accessing and securing WebLogic resources using the WebLogic Server Administration Console.

Documentation Audience

This document is intended for the these audiences:

- Server Administrators—Administrators work closely with Application Architects to design a security scheme for the server and the applications running on the server, to identify potential security risks, and to propose configurations that prevent security problems. Related responsibilities may include maintaining critical production systems, configuring and managing security realms, implementing authentication and authorization schemes for server and application resources, upgrading security features, and maintaining security provider databases.

- Application Administrators—Administrators who work with Server Administrators to implement and maintain security configurations and authentication and authorization schemes, and to set up and maintain access to deployed application resources. Application Administrators have general knowledge of security concepts and the Java Security architecture. They understand Java, XML, deployment descriptors, and can identify security events in server and audit logs.

Guide to this Document

The document is organized as follows:

- Chapter 2, "Understanding WebLogic Resource Security," introduces some 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 EJB and Web Application Resources,"describes options for securing EJB and Web application resources using deployment descriptors and/or the WebLogic Server Administration Console.

- Chapter 5, "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 6, "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 Server or Application 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*—Explains how to configure security for WebLogic Server®, including information about security providers, identity and trust, SSL, and Compatibility security.

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.*
- "Configure 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 Samples page.

New and Changed Features In This Release

WebLogic Server 9.0 introduces several changes to WebLogic Server resource security:

Resource Types

EJB and Web Application Resources

- **Security Deployment Models**—For each deployment, you can choose a model based on how you want to secure these resources; using the Administration Console, Deployment Descriptors, or a combination of descriptors and the console. You can still define a single model for all deployments in a security realm (now called the *Advanced Model*), which was the only option in the previous version. For more information, see "Choose a Security Model" on page 4-3.

- **Limited Support for JAAC**—If you are running servers within a WebLogic Server domain using JAAC (JSR-115) you are restricted to using the Deployment Descriptors security model.
- **Combined Role Mapping**—When you specify Deployment Descriptors for your security deployment model, WebLogic Server will ensure consistent behavior by combining application-defined role mappings with EJB and Web application role mappings. See "Understanding the Combined Role Mapping Enabled Setting" on page 4-14.

WorkContext

New resource type that can be secured using the Administration Console. See "Work Context Resources" on page 3-14.

Admin Resources

More methods (in addition to `unlockUser`) can be secured in the Administration Console. See "Administrative Resources" on page 3-2.

JDBC Resources

The list of `Admin` methods that can be accessed from the Administration Console has been updated. See "Java DataBase Connectivity (JDBC) Resources" on page 3-5.

Server Resources

Changes to methods that can be accessed and secured in the Administration Console: `suspend` and `resume` replace `lock` and `unlock`. See "Server Resources" on page 3-9.

Roles and Policies

Role and Policy Expressions

New built-in conditions are available for creating security role and policy expressions. See "Components of a Security Role: Conditions, Expressions, and Statements" on page 5-7 and "Components of a Security Policy: Conditions, Expressions, and Statements" on page 6-5.

Role and Policy Management

Use the Roles and Policies tables to obtain a central listing of all security roles and policies in a security realm. See List Security Roles and List Security Policies in *Administration Console Online Help*.

New Default Group and Role

The AppTesters default group and AppTester global security role have been added. See "Default Groups" on page 5-2 and "Default Global Roles" on page 5-5.

MBean Protections

The list of MBeans and methods secured by default global roles has been updated and expanded. See "Protected MBean Attributes and Operations" on page 5-6.

Understanding WebLogic Resource Security

This chapter describes how resource security works and outlines the main tasks for securing WebLogic Server resources:

- "Overview of Securing WebLogic Resources" on page 2-7
- "Securing WebLogic Resources: Main Steps" on page 2-9

Overview of Securing WebLogic Resources

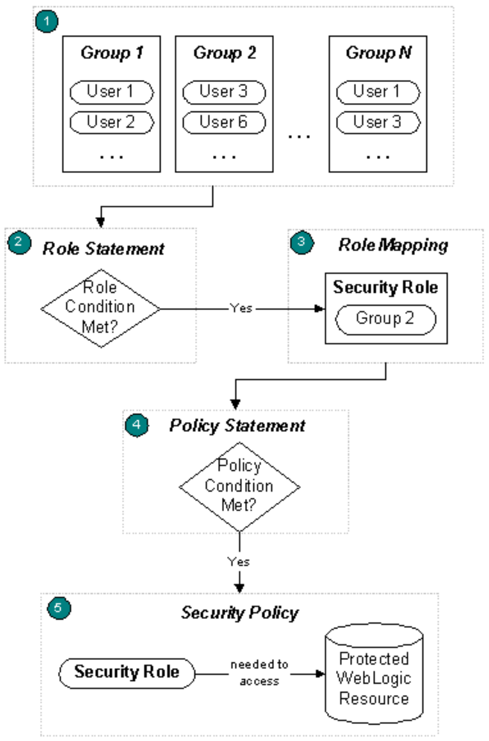
WebLogic Server security includes many unique terms and concepts. These terms and concepts, which you will encounter throughout the WebLogic Server security documentation, are defined in the "Security Fundamentals" and "Terminology" sections of *Understanding WebLogic Security*

A WebLogic resource represents an underlying WebLogic Server entity that can be protected from unauthorized access using security roles and security policies. Examples of WebLogic resources include enterprise applications (EARs), EJBs (JARs), and Web applications (WARs). For more information about the different types of WebLogic resources, see Chapter 3, "Types of WebLogic Resources."

Resource Security Process

Figure 2-1 illustrates the overall process for securing WebLogic resources, and a brief explanation follows.

Figure 2-1 Securing WebLogic Resources



1. Administrators statically assign users to groups, which can represent organizational boundaries. The same user can be a member of multiple groups. Figure 2-1 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 role statements, each of which include a role condition. The role condition specifies the circumstances under which a particular group should be granted the security role.

3. At runtime, the WebLogic 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 is referred to as role mapping. In Figure 2-1, 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 include 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 2-1, 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.

Securing WebLogic Resources: Main Steps

The main steps for securing a WebLogic resource are:

1. Determine which WebLogic resource to secure. For more information, see Chapter 3, "Types of WebLogic Resources.".
2. If you want to secure any Weblogic Resource except a Web application or EJB, go to step 4.
3. If you want to secure a Web application or EJB resource, you have a choice of security techniques and models. See Chapter 4, "Options for Securing EJB and Web Application Resources."
   a. Decide which security technique to use. See "Choose a Security Technique" on page 4-2.
   b. When you deploy your application, choose one of the security models. See "Choose a Security Model" on page 4-3

      If you choose the Advanced security model (compatible with WebLogic Server 8.x) you might need to reset come realm configurations. See "Using the Advanced Security Model" on page 4-6.
   c. If you want to use the WebLogic Server Administration Console to secure your Web application or EJB resource see step 4.
   d. If you want to use deployment descriptors to secure your Web application or EJB resource, see "Adding Declarative Security to Web Applications" or "Adding Declarative Security to EJBs" in *Programming WebLogic Security*, respectively.
4. Use the Administration Console to secure your WebLogic resource:

a. Create users and groups—representations of individuals and collections of individuals—who may be granted a security role. For more information, see For more information, see Manage Users and Groups in *Administration Console Online Help*.

b. Create security roles—dynamically computed privileges granted to users or groups based on specific conditions—which are used to restrict access to WebLogic resources. For more information, see Manage Security Roles in *Administration Console Online Help*.

   For information about expressions that you can use to define security roles, see "Components of a Security Role: Conditions, Expressions, and Statements" on page 5-7.

c. Create a security policy—an association between the WebLogic resource and a user, group, or security role—that specifies who has access to the WebLogic resource. For more information, see Manage Security Policies in *Administration Console Online Help*.

   For information about expressions that you can use to define security policies, see "Components of a Security Policy: Conditions, Expressions, and Statements" on page 6-5.

Types of WebLogic Resources

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

Overview of WebLogic Resource Types

A WebLogic resource represents any software component, such as a server, a service, an application, or an application artifact. In the context of this document, a WebLogic resource is any type of resource that can be secured using security roles and policies. WebLogic resources are hierarchical. The level at which you define security roles and security policies is up to you. For example, you can define security roles and security policies for an entire enterprise application (EAR), an Enterprise JavaBean (EJB) JAR containing multiple EJBs, a particular EJB within that JAR, or a single method within that EJB.

Administrative Resources

An Administrative resource is a type of WebLogic resource that allows users to perform administrative tasks. Examples of Administrative resources include the WebLogic Server Administration Console, the WebLogic Scripting Tool (WLST), and MBean APIs.

Administrative resources are similar to Server resources in that both are controlled by a layered security scheme that combines security policies and MBean protections. See “Layered Security Scheme for Server Resources” on page 3-10

Administrative Operations

When you secure Administrative resources, you can choose to protect all or any one of the operation categories defined in the WebLogic server's realm `AdminResource` object. These are described in Table 3-1.

Table 3-1 Categories And Default Groups For Administrative Resources

Administrative Category	Accessible Operations	Default Group Access
`UserLockout`	Allows user to access `unlockuser` to unlock users who have been locked out of their accounts.	`Administrators`
`Configuration`	Allows user access to configuration operations **Note:** BEA recommends that you not add groups for this category but rather add users to the default groups. Adding groups might interfere with the Layered security scheme. See "Layered Security Scheme for Server Resources" on page 3-10.	`Administrators, Deployers, Operators Monitors`
`FileUpload`	Allows user access to file upload operations on specified resources in the AdminResource object.	`Administrators, Deployers`
`FileDownload`	Allows users to access these methods in the AdminResource object: • `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`	`Administrators, Operators`

Table 3-1 Categories And Default Groups For Administrative Resources

Administrative Category	Accessible Operations	Default Group Access
`ViewLog`	Allows user access to view log operations on specified resources in the AdminResource object.	`Administrators, Deployers, Operators Monitors`
`Identity Assertion`	Allows user access to the `assertIdentity` action in the AdminResource object.	`Administrators`

For more information, see "Default Groups" on page 5-2 and "Default Global Roles" on page 5-5.

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 3-13) 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 EJB and Web Application 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 are an Oracle user, 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 3-7
- "Select Individual Methods" on page 3-8

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 is a specific type of WebLogic resource that is used to control the state of a WebLogic Server server instance. To secure Server resources, you create security policies and security roles for all WebLogic Server instances as a group, or individual servers.

Before securing WebLogic servers you need to understand the layered security scheme. See "Layered Security Scheme for Server Resources" on page 3-10.

Server Operations

When you are ready to secure a particular server, you can protect all operations on the server, or protect any 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.

Layered Security Scheme for Server Resources

The WebLogic scheme for securing server resources consists of two parts (or layers): security policies and MBean protections.

Default Security Policies for Server Resources

Like other types of WebLogic resources, access to Server resources is controlled with security policies created in the WebLogic Server Administration Console.

All server resources inherit a default security policy that is based on the `Admin` and `Operator` default global security roles. As described in "Default Global Roles" on page 5-5, the `Admin` and `Operator` global roles are granted specific privileges required for administrators to interact with administrative interfaces such as the Administration Console or the WLST utility. These default global roles are based on the default groups (described in "Default Groups" on page 5-2). Therefore, administrators who need access to Server resources must be members of either the `Administrators` or `Operators` default groups.

Notes: Because WebLogic Server grants the four default global roles to four default groups, adding a user to one of these groups automatically grants the user the global role.

If you enable the Domain-Wide Administration port, then only members of the `Administrators` (and not `Operators`) can access the securable server operations. see "Server Operations" on page 3-9 and Configure the domain-wide administration port in *Administration Console Online Help*.

Caution: Do not modify the default security policies for Server resources to make them more restrictive. 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).

MBean Protections

Each type of WebLogic resource (including a Server resource) exposes a set of its operations through its own implementation of the `weblogic.security.spi.Resource` interface (the `weblogic.security.service.ServerResource` class for Server resources). Therefore, the `ServerResource` class is the entity that is actually secured by the security policy described in "Default Security Policies for Server Resources" on page 3-10.

In WebLogic Server, the configuration of a Server resource is exposed through a set of MBeans. As such, the actions that the `ServerResource` class protects correspond to underlying MBean attributes and operations. For example, the `Resource` interface's `start()` method maps directly to the `start` operation of the `ServerRuntime` MBean.

The MBeans that expose the configuration of a Server resource are protected using one of the four default global roles. This protection is *in addition to* the security policy on the Server resource and is currently an unconfigurable protection supplied by the WebLogic Security Service. Therefore, although you can create your own global roles for securing Server resources, only users granted one of the default global roles can view or change the configuration of a server.

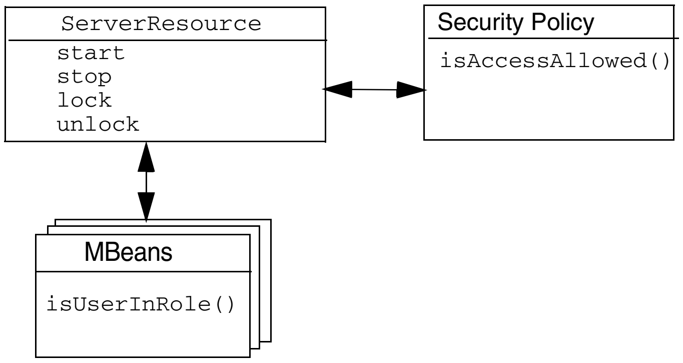
How the WebLogic Security Service Enforces Layered Protections

When a user tries to interact with a Server resource, the WebLogic Security Service:

- Determines whether the user is granted one of the default global roles permitted to change the attributes of the MBean or invoke the operations of the MBean.
- Checks the default security policy for the Server resource to verify that the global role is associated with the requested operation for this resource.

Therefore, a user must satisfy both of these security schemes for the request to be successful. Figure 3-1 shows how a security policy on the Server resource interacts with the security role-based protections on the underlying MBeans.

Figure 3-1 Layered Protections for Server Resources



Because the privileges given by the MBean protections are immutable, it is necessary to maintain security policies in a way that ensures consistency.

Maintaining a Consistent Security Scheme

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

For example, if you use the WebLogic Server Administration Console to add a user to the `Operator` global role (which is not one of the default policies for the server resource) but fail to add the `Operator` global role in the security policy defined for a Server resource, the user can call MBean operations that are used in the startup and shutdown sequence, but cannot use any Server resource operations to start or stop a server.

Similarly, if you use the Administration Console to remove the `Operator` global role from a security policy on the Server resource, a user granted the `Operator` global role can still call MBean operations but cannot call the Server resource. This result occurs because MBean protections for the default global roles are part of the WebLogic Security Service and are not currently configurable.

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

- Always give the `Admin` global role access to a Server resource.
- For a security policy on a server, use the `Operator` global role.

  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.

Permissions for Starting and Shutting Down Servers

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 protection). 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 EJB and Web Application 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 allow 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.

Accessing Resources Using the Administration Console

The way you access resources in the Administration console depends on how it is deployed. In addition, you can access security roles and policies for resources using the Domain Structure in the Administration console or, for central management you can use the Roles and Policies tables. For more information, see Create scoped security roles and Create security policies in *Administration Console Online Help.*

Options for Securing EJB and Web Application Resources

The following sections describe resource security configuration options for Enterprise Java Bean (EJB) and Web Application resources:

- "Security Techniques and Security Models" on page 4-1
- "Choose a Security Technique" on page 4-2
- "Choose a Security Model" on page 4-3
- "Using the Advanced Security Model" on page 4-6

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

Security Techniques and Security Models

Because the J2EE platform standardizes Web application and EJB 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 and EJB resources. You can use deployment descriptors exclusively, the Administration Console exclusively, or you can combine deployment descriptors and the console for certain situations. See "Choose a Security Technique" on page 4-2.

When you deploy your application, you select a security model based on the technique you want to use. WebLogic supports three different security models for individual deployments, and an Advanced security model for realm-wide configurations. See "Choose a Security Model" on page 4-3.

JAAC Security

If you are implementing security using JACC (Java authorization Contract for Containers) as defined in JSR 115) the security functions for Web applications and EJBs in the Administration Console are disabled. You must define security using deployment descriptors only. See "Using Deployment Descriptors" on page 4-2.

Choose a Security Technique

You can choose from among three techniques for securing Web application and EJB resources:

- "Using the WebLogic Server Administration Console" on page 4-2
- "Using Deployment Descriptors" on page 4-2
- "Using the Administration Console and Deployment Descriptors" on page 4-2

Using the WebLogic Server Administration Console

The primary benefit of using the Administration Console to secure Web application and EJB resources is unified security management. 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.

Using Deployment Descriptors

The primary benefit of securing your Web application and EJB resources through J2EE and WebLogic deployment descriptors is that it is a widely known, standard technique for adding declarative security to Web applications and EJBs. It may also be the technique with which most organizations are familiar. When using this technique, security roles and security policies are specified in the `web.xml`, `weblogic.xml` and `ejb-jar.xml`, `weblogic-ejb-jar.xml` deployment descriptors.

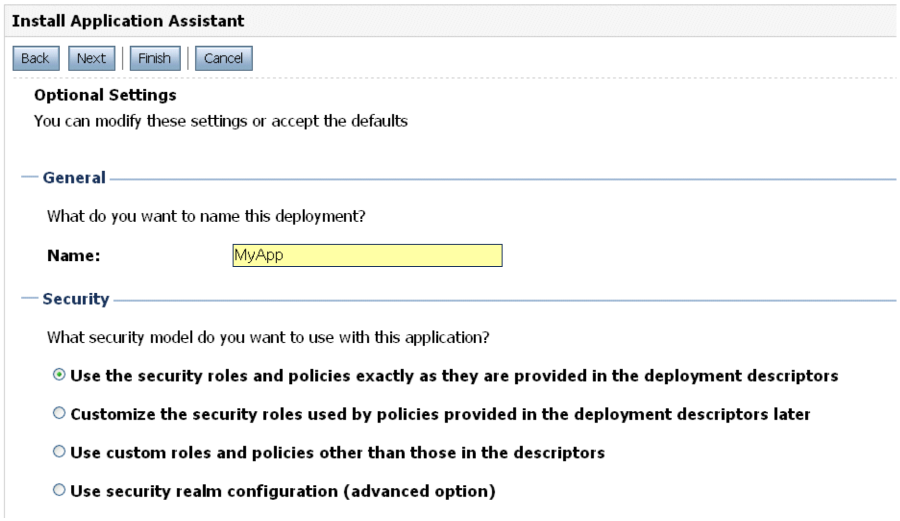
Using the Administration Console and Deployment Descriptors

Some organizations currently using deployment descriptors to secure their Web application and EJB resources may want to take advantage of the unified security management capabilities that the WebLogic Server Administration Console provides. The Administration Console can copy security configurations from existing deployment descriptors upon deployment of a Web application or EJB module. Once these security configurations are copied into the Role Mapping and Authorization providers' databases, you use the Administration Console for subsequent updates to security roles and security policies.

Choose a Security Model

A security model depends on the security technique that you choose (see "Choose a Security Technique" on page 4-2). When you are ready to deploy an application you can choose from among several security models. If you are using the Deployment tool in the WebLogic Server Administration Console, the options appear as follows:

Figure 4-1 Administration Console: Security Section of the Deployment Assistant



Optional Settings

You can modify these settings or accept the defaults

General

What do you want to name this deployment?

Name: MyApp

Security

What security model do you want to use with this application?

- [✓ **Use the security roles and policies exactly as they are provided in the deployment descriptors**]
- **Customize the security roles used by policies provided in the deployment descriptors later**
- **Use custom roles and policies other than those in the descriptors**
- **Use security realm configuration (advanced option)**

The security model you choose applies for the lifetime of the deployment. To change a security model you need to delete the deployment and create a new one.

Note: You can set the default deployment security model using the **Security Model** realm setting. For more information, see Set the default security deployment model in *Administration Console Online Help*.

the following sections describe each model:

- "Deployment Descriptors Only Model" on page 4-4

- "Customize Roles Model" on page 4-4
- "Customize Roles and Policies Model" on page 4-5
- "Security Realm Configuration (Advanced Model)" on page 4-5

Deployment Descriptors Only Model

This model corresponds to the first Security option shown in Figure 4-1 above.

This security model uses only roles and policies defined by a developer in the J2EE deployment descriptor (DD) and the WebLogic Server DD. The administrator verifies that the security Principals (groups or users) are mapped properly. Table 4-1 shows a typical scenario:

Table 4-1 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.

For more information, see the "Using Declarative Security With Web Applications" and "Using Declarative Security With EJBs" sections of *Programming WebLogic Security*.

If you select Deployment Descriptors Only, WebLogic Server uses combined role mapping by default. For more information, see "Understanding the Combined Role Mapping Enabled Setting" on page 4-14.

Customize Roles Model

This model corresponds to the second Security option shown in Figure 4-1 above.

This security model uses policies defined in the J2EE DD, and ignores any Principal mappings in the WebLogic Server DD. The administrator completes the role mappings using the Administration Console. Table 4-2 shows a typical scenario:

Table 4-2 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.

For more information, see Create scoped security roles in *Administration Console Online Help.*

Customize Roles and Policies Model

This model corresponds to the third Security option shown in Figure 4-1 above.

This security model ignores any roles and policies defined in deployment descriptors The administrator uses the Administration Console to secure the resources.Table 4-3 shows a typical scenario:

Table 4-3 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.

For more information, see Create scoped security roles and Create security policies in *Administration Console Online Help.*

Security Realm Configuration (Advanced Model)

This model corresponds to the fourth Security option shown in Figure 4-1 above.

This security model lets you define a single security model for all deployments containing EJB and Web application resources in the security realm. With this model you can choose one of the following security techniques:

- All EBJ and Web application deployments will be secured using deployment descriptors only.
- All EBJ and Web application deployments will be secured using the Administration Console only.
- On the initial deployment, security roles and policies will be copied from the deployment descriptors, but the Administration Console will control security thereafter: The deployment descriptors will be ignored for subsequent deployments. (Combined technique).

Note: Prior to the current release of WebLogic Server, this was the only security model available. Select Advanced Security if you want to continue to secure EJBs and Web Applications as in the previous release.

If you choose the Advanced option you need to set the following realm security configurations after the initial deployment:

- Check Security Roles and Security Policies.
- On Future Redeploys

For details see "Using the Advanced Security Model" on page 4-6.

Using the Advanced Security Model

Note: This section applies only if you choose to control security for EJBs and Web Applications at the security realm level rather than for each deployment (Advanced security option).

Whether using the WebLogic Server Administration Console or deployment descriptors to secure your Web application and EJB resources, there are two important settings in WebLogic Server that you need to understand: the **Check Security Roles and Policies** setting and the **On Future Redeploys** setting. Failure to understand these settings could result in incorrect or lost security configurations.

Before attempting to secure Web application or EJB resources using the Advanced security option, read the following sections:

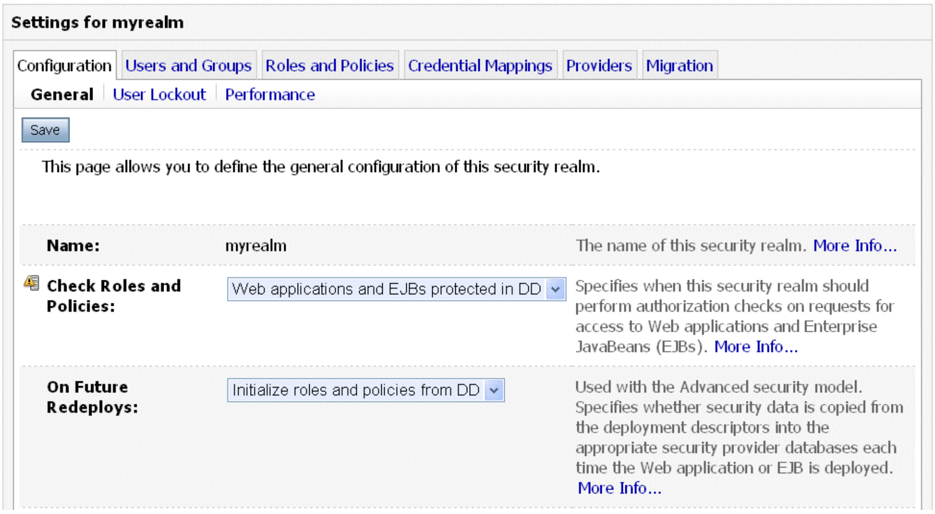
- "Understanding How to Check Security Roles and Security Policies" on page 4-7
- "Understanding the On Future Redeploys Setting" on page 4-8
- "How The Check Roles and Policies and On Future Redeploys Settings Interact" on page 4-10

- "How The Check Roles and Policies and On Future Redeploys Settings Interact" on page 4-10
- "Understanding the Combined Role Mapping Enabled Setting" on page 4-14

Understanding How to Check Security Roles and Security Policies

To give you control over performance, the WebLogic Server Administration Console requires that you specify how the WebLogic Security Service should control security. You specify this preference through the Check Roles and Policies drop-down menu highlighted in Figure 4-2.

Figure 4-2 Administration Console: Realms Settings for Advanced Security Model



This page allows you to define the general configuration of this security realm.

Name:	myrealm	The name of this security realm. More Info...
Check Roles and Policies:	Web applications and EJBs protected in DD	Specifies when this security realm should perform authorization checks on requests for access to Web applications and Enterprise JavaBeans (EJBs). More Info...
On Future Redeploys:	Initialize roles and policies from DD	Used with the Advanced security model. Specifies whether security data is copied from the deployment descriptors into the appropriate security provider databases each time the Web application or EJB is deployed. More Info...

- When the value of the Check Roles and Policies setting is `Web applications and EJBs Protected in DD`, the WebLogic Security Service *only* performs security checks on Web application and EJB resources that have security specified in their associated deployment descriptors (DDs). This is the default Check Roles and Policies setting.
- When the value of the Check Roles and Policies setting is `All Web applications and EJBs`, the WebLogic Security Service performs 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.

If you change the value of the Check Roles and Policies drop-down menu to `All Web applications and EJBs`, you also need to specify what the WebLogic Security Service should do when the Web application or EJB module is redeployed. For more information, see "Understanding the On Future Redeploys Setting" on page 4-8.

Note: The Check Roles and Policies setting affects all the WebLogic Server instances (servers) in the WebLogic Server domain in which it is set.

Using the fullyDelegateAuthorization Flag

You can also specify how to control security on Web application and EJB resources using the `fullyDelegateAuthorization` flag, a command line argument that you set when you start WebLogic Server.

Note: This setting applies only when using the Advanced security model.

When the value of the `fullyDelegateAuthorization` flag is false, the WebLogic Security Service only performs security checks on Web application and EJB resources that have security specified in their associated deployment descriptors.

To set this flag, ttype:

`-Dweblogic.security.fullyDelegateAuthorization` = *boolean_value*

where *boolean_value* is `true` or `false` on the command line each time you start a WebLogic Server instance.

Note: You should ensure that the `fullyDelegateAuthorization` flag is set the same way for both your Administration and Managed Servers.

Understanding the On Future Redeploys Setting

If you decide that the WebLogic Security Service should control security on All Web applications and EJBs in the Check Roles and Policies drop-down menu, you also need to tell WebLogic Server which technique you want to use to secure these Web application and EJB resources. (See "Choose a Security Technique" on page 4-2 for more information.) Specify this preference in the Future Redeploys drop-down menu highlighted in Figure 4-2.

Set the value of the Future Redeploys drop-down menu 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 security policies in *Administration Console Online Help*.

- To secure Web application and EJB resources using *only* the deployment descriptors (that is, the `web.xml`, `weblogic.xml`, `ejb-jar.xml`, and `weblogic-ejb-jar.xml` files), select Initialize Roles and Policies from DD and refer to the "Using Declarative Security With Web Applications" and "Using Declarative Security With EJBs" sections of *Programming WebLogic Security*, respectively.

Warning: Switching the value of the Future Redeploys setting is not advised because it can lead to incorrect or lost security configurations. If you need to switch between these settings (specifically for the situations described in "Using the Administration Console and Deployment Descriptors" on page 4-2), follow the instructions in "Using the Combined Technique with the Advanced Security Model" on page 4-11.

How The Check Roles and Policies and On Future Redeploys Settings Interact

Table 4-4 shows how to achieve the behavior you want from the WebLogic Security Service using different combinations of the Check Roles and Policies and Future Redeploys settings.

Table 4-4 Interaction Between Check Roles and Policies Setting and On Future Redeploys 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 On Future Redeploys 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	--

How to Modify Check Roles and Policies and On Future Redeploys Settings

Use the Administration Console to change this setting. For more information, see Revert the On Future Deploys setting in *Administration Console Online Help.*

Using the Combined Technique with the Advanced Security Model

As described "Using the Administration Console and Deployment Descriptors" on page 4-2, you can combine settings in the WebLogic Server Administration Console and J2EE/WebLogic deployment descriptors. You would typically do so for two reasons:

- To copy security configurations from deployment descriptors into the configured Authorization and Role Mapping providers' databases, upon initial deployment of Web application and EJB resources. This process enables you to use the Administration Console for subsequent modifications to security roles and security policies.
- To reinitialize security configurations for Web application and EJB resources to their original state, as specified in the deployment descriptors.

Caution: Use of the combined technique for any other purposes is not recommended.

The following sections provide step-by-step instructions for using the combined technique to secure your Web application and EJB resources:

- "Copying Security Configurations From a Deployment Descriptor" on page 4-11
- "Reinitializing Security Configurations" on page 4-13

Copying Security Configurations From a Deployment Descriptor

These instructions are intended for administrators using the Advanced security model or upgrading from WebLogic server version 8.x who presently secure Web application and EJB resources using J2EE and WebLogic deployment descriptors, but want to exclusively use the WebLogic Server Administration Console from this point forward.

Note that BEA does *not* recommend maintaining security configurations in both the deployment descriptors and the Administration Console—this is a step toward migrating to full security administration using the Administration Console.

Caution: When using the combined technique, it is possible to override security configurations for Web application and EJB resources. Therefore, take extra care to ensure that the appropriate security configuration is in place Web application and EJB resources.

To copy security configurations for a Web application or EJB resource so that you can use the WebLogic Server Administration Console for subsequent modifications, follow these steps:

- Step 1: Modify the Security Realms settings and deploy the resource.

  - See Copy security configurations from a deployment descriptor in *Administration Console Online Help.*

- "Step 2: Verify the Copied Security Policies (Optional)" on page 4-12
- "Step 3: Verify the Copied Security Roles (Optional)" on page 4-13
- Step 4: Revert the On future Deploys setting.
  - See Revert the On Future Deploys setting in *Administration Console Online Help.*
- Step 5: Modify security roles and policies using the administration Console.
  - See Create scoped security roles and Create security policies in *Administration Console Online Help*

Step 2: Verify the Copied Security Policies (Optional)

To verify the copied security policies, follow the instructions shown in the appropriate column of Table 4-5.

Table 4-5 Verifying Copied Security Policies, Based on Resource Type

Step	Web Application Resources	EJB Resources
1	Open the `web.xml` deployment descriptor for the Web application, and record the content of any `<url-pattern>` and `<http-method>` elements, as well as any `<role-name>` subelements of the `<auth-constraint>` element.	Open the `ejb-jar.xml` deployment descriptor for the EJB, and record the content of any `<method-permission>` elements, specifically focusing on the `<role-name>`, `<ejb-name>`, and `<method-name>` subelements.
2	Using the Administration console to view the security policies and associated security roles for each resource. See Access policies for Web application resources in *Administration Console Online Help*.	Using the Administration console to view the security policies and associated security roles for each resource. See Access policies for EJB resources in *Administration Console Online Help*.

Step 3: Verify the Copied Security Roles (Optional)

To verify the copied security roles, follow the instructions shown in the appropriate column of Table 4-6.

Table 4-6 Verifying Copied Security Roles, Based on Resource Type

Step	Web Application Resources	EJB Resources
1	Open the `weblogic.xml` deployment descriptor for the Web application, and record the content of any `<security-role-assignment>` elements, specifically focusing on the `<role-name>` and `<principal-name>` subelements. **Note:** If the deployment descriptor uses the `<externally-defined>` element for a Web application, no scoped roles are actually defined; therefore no scoped roles for the EJB can be copied.	Open the `weblogic-ejb-jar.xml` deployment descriptor for the EJB, and record the content of any `<security-role-assignment>` elements, specifically focusing on the `<role-name>` and `<principal-name>` subelements. **Note:** If the deployment descriptor uses the `<externally-defined>` element for an EJB, no scoped roles are actually defined; therefore no scoped roles for the EJB can be copied.
2	Using the Administration console to view the security roles for each resource. See Access roles for Web application resources in *Administration Console Online Help*.	Using the Administration console to view the security roles for each resource. See Access roles for EJB resources in *Administration Console Online Help*.

Reinitializing Security Configurations

To reinitialize security configurations for Web application and EJB resources to their original state as specified in their deployment descriptors, follow these steps:

- Step 1: Modify the Security Realms settings and deploy the resource.
  - See Reinitialize security configurations from a deployment descriptor in *Administration Console Online Help.*
- Step 2: Verify the reinitialized security policies and security roles (Optional)
  - Follow the instructions shown in the appropriate column of Table 4-5, "Verifying Copied Security Policies, Based on Resource Type," on page 4-12 and Table 4-6, "Verifying Copied Security Roles, Based on Resource Type," on page 4-13
- Step 3: Revert the On future Deploys setting.

  - See Revert the On Future Deploys setting in *Administration Console Online Help.*

- Step 4: Modify security roles and policies using the Administration Console.
  - See Create scoped security roles and Create security policies in *Administration Console Online Help*

Understanding the Combined Role Mapping Enabled Setting

The CombinedRoleMappingEnabled setting is provided for backward compatibility with the previous version (8.x) of WebLogic Server. For all applications initially deployed in version 9.x, the default value for this setting "true." (enabled). For all applications previously deployed in version 8.1 and upgraded to version 9.x, the default value is "false" (disabled).

The Combined Role Mapping Enabled setting determines how the role mappings in the Enterprise Application, Web application, and EJB containers interact. Table 4-7 shows a summary of the contrasting behaviors:

Table 4-7 Contrasting Behaviors For CombinedRoleMappingEnabled Flag

When CombinedRoleMapping is disabled...	When CombinedRoleMapping is enabled...
By default, 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 such that all principal mappings are included.
Unless a role mapping is defined in the Web application, the Web application container assumes any role mapping defined in `web.xml`.	The Web application container uses the base reference for roles in web.xml to create an empty role map.
Role mappings for EJBs must be defined in web.xml. Role mappings defined in the other containers are not used unless `<externally-defined>`	The EJB container uses the base reference for roles in web.xml to create an empty role map

Usage examples

The following examples show the differences in role mapping behaviors depending on whether CombinedRoleMappingEnabled 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.

Changing the default setting

There are two reasons why you might want to change the default value for Combined Roles 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.

CHAPTER 5

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

WebLogic Server defines the groups shown in Table 5-1. By default, WebLogic Server assigns some of these groups to corresponding global security roles. See "Default Global Roles" on page 5-5.

Table 5-1 Default Groups

Group Name	Membership
`users`	Users, when they log in (for example, through a Web page). The `users` group includes all users except the `<anonymous>` user.
`everyone`	Every user is a member of this group. The `users` group is nested within the `everyone` group.
`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 "Adding Users To Administrators Group" on page 5-2.
`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.

Adding Users To Administrators Group

You should 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".

Using the Administration Console to Manage Users and Groups

You can use the WebLogic Administration Console to add and mange users and group for a security realm using WebLogic Authentication provider. For more information, see Manage Users and Groups in *Administration Console Online Help*.

Note: The instructions for the using the Administration Console apply to users and groups in the WebLogic Authentication provider only. If you customize the default security configuration to use a custom Authentication provider, use the administration tools supplied by that security provider to create a user.

If you are upgrading to the WebLogic Authentication provider, you can load existing users and groups into its database. For more information, see Migrating Security Data in *Securing WebLogic Server*.

Overview of Security Roles

A security role is a privilege granted to users or groups based on specific conditions. Like groups, security roles allow you to restrict access to WebLogic resources for several users at once. Security roles differ from groups as follows:

- 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.
- 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 5-4.

Granting a security role to a user or a group confers the defined access privileges to that user or group, as long as the user or group is "in" the security role. For example, an administrator may define a security role called `AppAdmin`, which has write access to a particular Web application's resources. Any user or group granted the `AppAdmin` security role would then have write access to that Web application resource. Multiple users or groups can be granted a single security role.

Dynamic Role Mapping

At runtime, the WebLogic Security Service compares users or groups against a role condition to determine whether they should be dynamically granted a security role. This process is referred to as role mapping, and occurs just prior to when 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.

Note: For more information about role conditions and access decisions, see "Components of a Security Role: Conditions, Expressions, and Statements" on page 5-7 and "Access Decisions" in *Developing Security Providers for WebLogic Server*, respectively.

This dynamic mapping of security roles to users or groups provides a very important benefit: users or groups can be granted a security role based on business rules, or the context of the request. For example, a user may be allowed to be in a `Manager` security role only while the actual manager is away. Dynamically granting this security role means that you do not need to change or redeploy your application to allow for such a temporarily 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.

Dynamic Role Mapping and Security Policies

Security policies also work dynamically, in that authorization decisions are made at runtime. (See Chapter 6, "Security Policies.") The use of dynamic role mapping in conjunction with security policies provides additional flexibility. Generally speaking, if your authorization decision is based on the resource rather than the entities (roles) accessing the resource, you would add security expressions to the security policy. If authorization is focused on who is accessing the resource, then you would add expressions to the security role.

Types of Security Roles: Global Roles and Scoped Roles

There are two types of security roles in WebLogic Server: global roles and scoped roles.

- A security role that applies to all WebLogic resources deployed within a security realm (and thus the entire WebLogic Server domain) is called a global role.

  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 security role that applies to a specific instance of a WebLogic resource deployed in a security realm (such as a method on an EJB or a branch of a JNDI tree) is called a scoped role. You define scoped security roles for any of Weblogic resources described in Chapter 3, "Types of WebLogic Resources."

Combining role types (either global or scoped) is allowed when creating a security policy for a WebLogic resource. (For more information, see Chapter 6, "Security Policies.")

While BEA recommends creating and using security roles (rather than users or groups) to secure WebLogic resources, you do not need to use a particular type of security role. 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 5-5.

You may never need to use scoped roles. They are provided for their flexibility and are an extra feature for advanced customers.

Default Global Roles

By default, WebLogic Server defines the global roles shown in Table 5-2. The table summarizes the privileges that users or groups in these security roles are granted, and the default group assignments for each role. When a user is added to one of the groups, that user is automatically granted the global role.

The default global roles are used in the default security policies that protect most types of WebLogic resources. In addition, the default global roles are used to provide additional security for Server resources that are exposed as MBeans. For details, see "Protected MBean Attributes and Operations" on page 5-6.

Table 5-2 Default Global Roles, Privileges, and Default Group Assignments

Global Role	Privileges	Default Group Assignments
`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`
`Admin`	• View the server configuration, *including* the encrypted value of some encrypted attributes. • Modify the entire server configuration. • Deploy Enterprise Applications, startup and shutdown classes, and Web application, EJB, J2EE Connector, and Web Service modules. • Start, resume, and stop servers by default.	`Administrators`

Table 5-2 Default Global Roles, Privileges, and Default Group Assignments

Global Role	Privileges	Default Group Assignments
`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 JSR-88 deployment operations.	`Deployers`
`Operator`	• View the server configuration, *except* for encrypted attributes. • Start, resume, and stop servers by default.	`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`

Protected MBean Attributes and Operations

WebLogic server protects access to MBeans by assigning one or more default global roles to individual MBeans and MBean attributes (See "Default Global Roles" on page 5-5.) Note that the Admin role grants full access to all MBean resources.

For a complete listing of global role access to MBeans, see Security Roles for MBeans in *WebLogic Server MBean Reference*.

Components of a Security Role: Conditions, Expressions, and Statements

Role Conditions

A role condition is a condition under which a security role (global or scoped) is granted to a user or group at runtime. Weblogic provides three kinds of built-in role conditions. You can use these conditions in any logical combination:

- "Basic Role Conditions" on page 5-7
- "Date and Time Role Conditions" on page 5-8
- "Context Element Role Conditions" on page 5-8

Basic Role Conditions

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

- `Group`—Creates a condition for a security role based on a group. 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.

- `User`—Creates a condition for a security role based on a user name. For example, you might create a condition indicating that only the user `John` can be granted the `BankTeller` security role.
- `Server is in development mode`—Creates a condition for a security policy based on whether the server is running in development mode.
- `Allow access to everyone`—Creates a condition for a security policy that includes all users and groups.
- `Deny access to everyone`—Creates a condition for a security policy that excludes all users and groups.

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`—Creates a condition for a security role based on a 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`—Creates a condition for a security role based on a time 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`—Creates a condition for a security role based on a time 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`—Creates a condition for a security role based 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`—Creates a condition for a security role based 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`—Creates a condition for a security role based on a time 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`—Creates a condition for a security role based on the existence of a specified attribute or parameter.
- `Context element's value equals a numeric constant`—Creates a condition for a security 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`—Creates a condition for a security 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`—Creates a condition for a security 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`—Creates a condition for a security role based on a specified attribute or parameter's string value being equal to a specified string.

Role Expressions

Role conditions, combined with specific information you supply for the condition (such as an actual user name, group, or start/stop times), are called expressions. One or more expressions that define the role are known collectively as a role statement (see "Role Expressions" on page 5-9). The simplest role statement would have one expression: for example, you might have a group as the only condition for the role.

Role Statement

A role statement is a collection of expressions that define how a security role is granted. The ability to use multiple expressions means that you can create complex security roles that meet your organization's security requirements. You can also use `and or` operators between expressions, and you can reorder and negate expressions. For details, see Create scoped security roles in *Administration Console Online Help*

The entire role statement must be true in order for a user or group to be granted the security role. More restrictive expressions should come later in a role statement.

Using the Administration Console to Manage Security Roles

You can use the WebLogic Administration Console to create global security roles and to access Weblogic resources for creating, modifying, and removing scoped security roles. For more information, see Manage Security Roles in *Administration Console Online Help*.

CHAPTER 6

Security Policies

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

Overview of Security Policies

A security policy is an association between a WebLogic resource and one or more users, groups, or security roles and is designed to protect the WebLogic resource against unauthorized access.

Note: Security policies replace the access control lists (ACLs) and permissions that were used to protect WebLogic resources in previous releases of WebLogic Server.

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. For example, you can define security policies on an entire enterprise application (EAR), an EJB JAR containing multiple EJBs, a particular EJB within that JAR, or a single method within that EJB.

If you create a security policy for a *type* of WebLogic resource (for example, EJB resources), all new instances of that WebLogic resource inherit the security policy. This inheritance of security policies means that you can secure multiple WebLogic resources efficiently. Out of the box, WebLogic Server secures each WebLogic resource type with a default security policy that is inherited by all instances of that WebLogic resource. For more information, see "Default Security Policies" on page 6-2.

A security policy created for a specific instance of a WebLogic resource will override any security policy assigned to the WebLogic resource type. So, if you create a security policy for a particular EJB, this security policy (and not the one you created for the EJB resource type) will be used.

Security Policy Storage and Prerequisites for Use

Security policies are stored in the security provider database of the Authorization provider that is configured in the default (active) security realm. By default, the WebLogic Authorization provider is configured, and security policies are stored in the embedded LDAP server.

When creating a security policy with a user or group, the user or group must be defined in the security provider database of the Authentication provider that is configured in the default security realm. When creating a security policy with a security role, the security role (whether global or scoped) must be defined in the security provider database of the Role Mapping provider that is configured in the default security realm. By default, the WebLogic Authentication and Role Mapping providers are configured, and the default groups and default global roles are stored in the databases for these security providers (also 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 Security Policies

By default, WebLogic Server defines the security policies shown in Table 6-1. These security policies are defined for each type of WebLogic resource described in Chapter 3, "Types of WebLogic Resources," and are based on the default global roles and default groups.

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

Table 6-1 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 security 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).

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 5-2, “Default Global Roles, Privileges, and Default Group Assignments,” on page 5-5. 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 Protections” on page 3-10. 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 *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 “Protected MBean Attributes and Operations” on page 5-6 and Managing Security Realms with JMX in *Developing Custom Management Utilities with JMX*.

Components of a Security Policy: Conditions, Expressions, and Statements

Policy Conditions

A policy condition is a condition under which a security policy will be created. WebLogic provides three kinds of built-in policy conditions. You can use these conditions in any logical combination:

- "Basic Policy conditions" on page 6-5
- "Date and Time Policy Conditions" on page 6-6
- "Context Element Policy Conditions" on page 6-7

Basic Policy conditions

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

- `Role`—Creates a condition for a security policy based on a security role. For example, you might create a condition indicating that only users and groups in the `BankTeller` security role can access the `Deposit` EJB.

  For some WebLogic resources, WebLogic server assigns a default security policy based on one of the built-in global security roles. For more information, see "Default Global Roles" on page 5-5 and "Default Security Policies" on page 6-2.

- `User`—Creates a condition for a security policy based on a user name. For example, you might create a condition indicating that only the user `John` can access the `Deposit` EJB.
- `Group`—Creates a condition for a security policy based on a group. When a group is used to create a security policy, the security policy is assigned to all members of the group. For example, you might create a condition indicating that only users in the group `FullTimeBankEmployees` can access the `Deposit` EJB.
- `Server is in Development Mode`—Creates a condition for a security policy based on whether the server is running in development mode.
- `Allow access to everyone`—Creates a condition for a security policy that includes all users and groups.
- `Deny access to everyone`—Creates a condition for a security policy that excludes all users and groups.

- `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 role conditions. The date and time policy conditions available in this release of WebLogic Server are:

- `Access occurs between specified hours`—Creates a condition for a security policy based on a specified time period. For example, you might create a condition granting access to users only during business hours.
- `Access occurs after`—Creates a condition for a security policy based on a time 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`—Creates a condition for a security policy based on a time 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`—Creates a condition for a security policy based 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`—Creates a condition for a security policy based 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`—Creates a condition for a security policy based on a time 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`—Creates a condition for a security policy based on a time 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`—Creates a condition for a security policy based on the existence of a specified attribute or parameter.
- `Context element's value equals a numeric constant`—Creates a condition for a security policy 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`—Creates a condition for a security policy 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`—Creates a condition for a security policy 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`—Creates a condition for a security policy based on a specified attribute or parameter's string value being equal to a specified string.

Policy Expressions

Policy conditions, combined with specific information you supply for the condition (such as an actual role name, or start/stop times), are called expressions. One or more expressions that define the policy are known collectively as a policy statement (see next section). For example: only users who are included in the `BankTeller` security role and during regular banking hours can access a business Web application.

Policy Statement

A policy statement is a collection of expressions that define how a security role is granted. The ability to use multiple expressions means that you can create complex security roles that meet your organization's security requirements. You can also use `and` `or` operators between expressions, and you can reorder and negate expressions. For details, see Create Security Policies in *Administration Console Online Help.*

The entire role statement must be true in order for a user, group, or security role to be granted the security role. More restrictive expressions should come later in a policy statement.

Using the Administration Console to Manage Security Policies

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*.

Index