IBM® WEBSPHERE® APPLICATION SERVER V6.1 – LAB EXERCISE

Lab 06 - Fine-grained Administrative Security

What this exercise is about

The objective of this lab is to provide you with an understanding of the new fine-grained administrative security feature in WebSphere Application Server V6.1. You will gain hands on experience managing users, resources, and authorization groups using the wsadmin commands. Specifically, you will create new authorization groups and add resources and users to the authorization groups. You will also test user access and management capabilities based on the new security configuration.

Lab Requirements

List of system and software required for the student to complete the lab.

- WebSphere Application Server Network Deployment V6.1 installed.
- You will need to have a cell environment created that contains:
  - A deployment manager with administrative security enabled.
  - A node with the default application installed.
  - Note that this environment is created as a part of the installation lab.

What you should be able to do

At the end of this lab you should be able to:

- Create authorization groups.
- Add users and resources to authorization groups using the wsadmin commands.

- Restrict user access to specific application servers using the new fine-grained administrative security capabilities.

Introduction

In versions of WebSphere Application Server prior to WAS V6.1, only a single user registry is available for the WAS security realm associated with a WAS cell. In WAS V6.1 the Virtual Member Manager allows for the aggregation of multiple underlying registries into a single logical registry for the WAS security realm.

This lab will demonstrate the configuration of a federated repository by adding a LDAP based registry to the file based registry that is used by default in WAS V6.1 when security is enabled.

In previous releases of WebSphere Application Server, users with administrative privileges had authority to administer all of the resources in a cell. In other words, administrative security was maintained at the cell level. WebSphere Application Server V6.1 offers a more fine-grained approach to administrative security. Users can now be defined to administrative roles on a specific set of resources, such as a cell, node group, node, cluster, server, or even an application.

To achieve this instance-based fine-grained security, resources that require the same privileges are placed in a group called the administrative authorization group or authorization group. Users can be granted access to the authorization group by assigning to them the required administrative role.

In this lab, we will be using the wsadmin command to configure fine-grained administrative security. **Fine-grained administrative security functionality is not implemented in the Administrative Console; it is only available through the wsadmin scripting interface**.

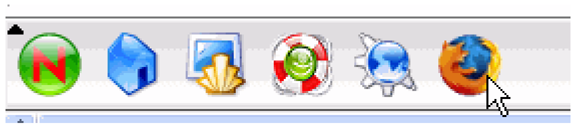
In this lab, we will create authorization groups, allocate resources to those authorization groups, and assign the users to the proper groups. This will all be accomplished using wsadmin commands.

Part 1: Change the Registry to LDAP

In this part of the lab, we change the security registry from the default file based registry to a LDAP server.
.

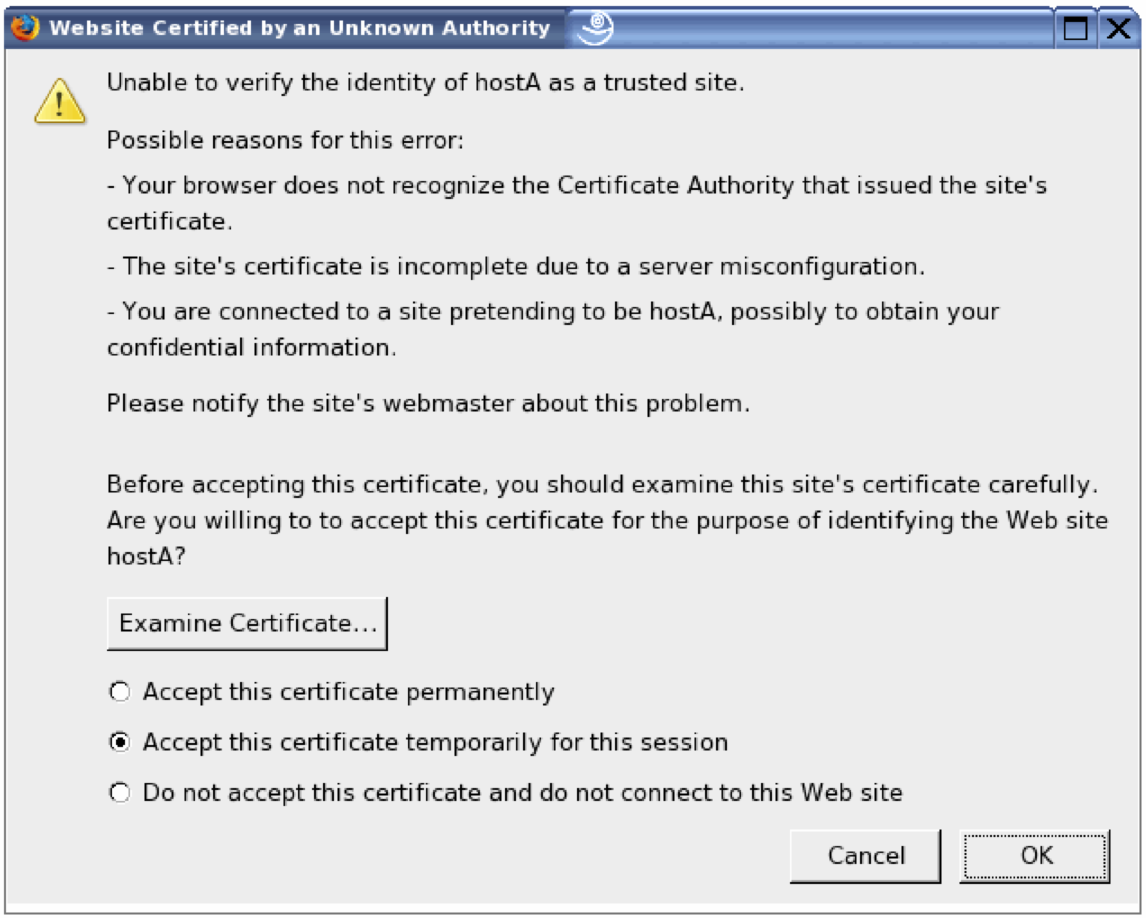
NOTE: If you have stopped the deployment manager or the node agent you will need to start both of these prior to running this lab. Please refer to the instructions in the Systems Management lab if you don't know how to do so.

____ 1. From the SLES desktop, locate the panel at the bottom of the workspace and select the Firefox browser icon and click on the icon as shown below.



Note: The browser home page should already be set for https://localhost:9043/ibm/console If the WAS adminconsole doesn't start automatically you will need to enter the URL

____ **2.** When the browser starts if you're presented with the Unknown Authority for the Cert warning as shown below and click on **OK.**



Website Certified by an Unknown Authority

Unable to verify the identity of hostA as a trusted site.

Possible reasons for this error:

- Your browser does not recognize the Certificate Authority that issued the site's certificate.

- The site's certificate is incomplete due to a server misconfiguration.

- You are connected to a site pretending to be hostA, possibly to obtain your confidential information.

Please notify the site's webmaster about this problem.

Before accepting this certificate, you should examine this site's certificate carefully. Are you willing to to accept this certificate for the purpose of identifying the Web site hostA?

Examine Certificate...

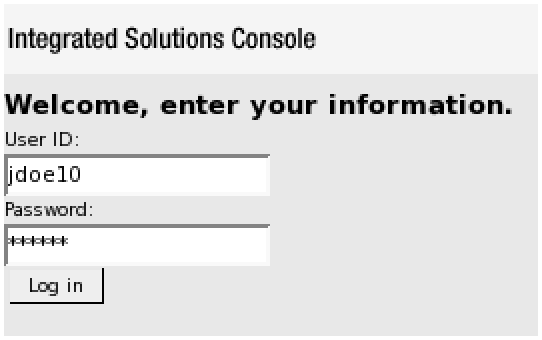
○ Accept this certificate permanently

◉ Accept this certificate temporarily for this session

○ Do not accept this certificate and do not connect to this Web site

Cancel OK

____ 3. Login as the user "jdoe10" with the password "jdoe10" as shown below and click **Login in**


Integrated Solutions Console

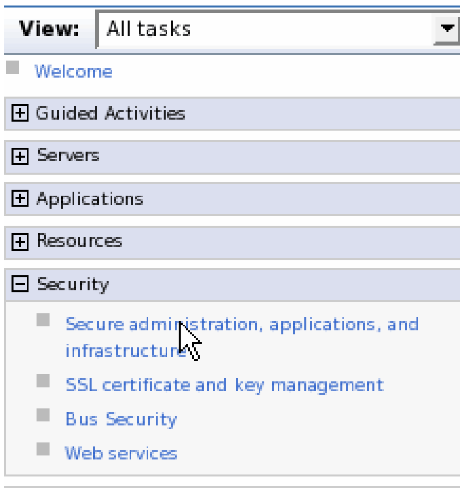
Welcome, enter your information.

User ID: jdoe10

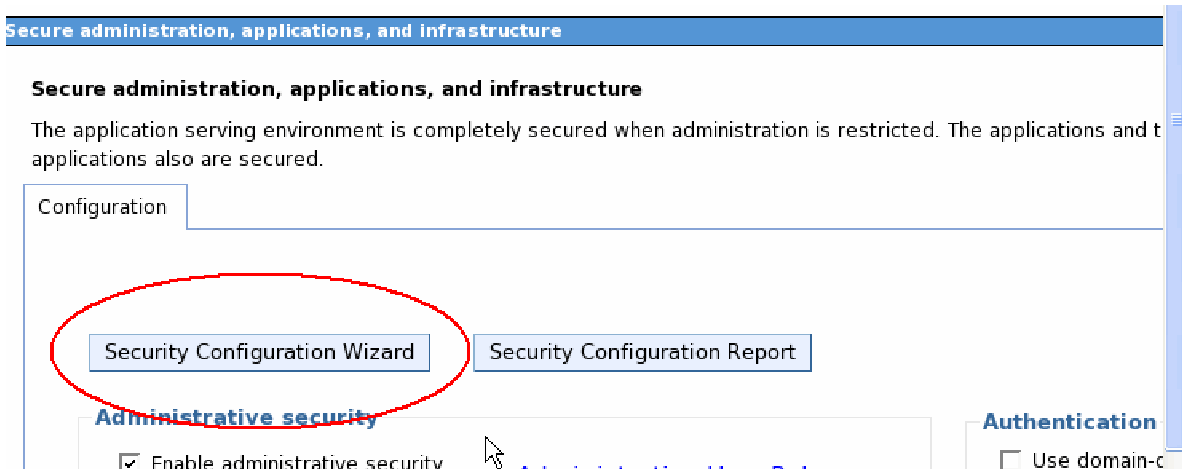
Password: ******

NOTE: While you'll likely be tempted to do so since the passwords are the same as the User ID, when entering the User ID and password in the console or in command line dialog boxes do not attempt to copy and paste the User ID into the password. In developing this lab material and delivering it, there have been problems using copy and paste for this purpose. It's not clear if this is related to Linux or VMWare or a combination of the two, but it has occurred often enough and slowed completion of the exercises that it's best to avoid the problem.

____ 4. Expand the **Security** tasks in the left panel, and click **Security administration, applications and infrastructure** as shown below



____ 5. In the right hand panel, Click **Security Configuration Wizard** as shown below


Secure administration, applications, and infrastructure

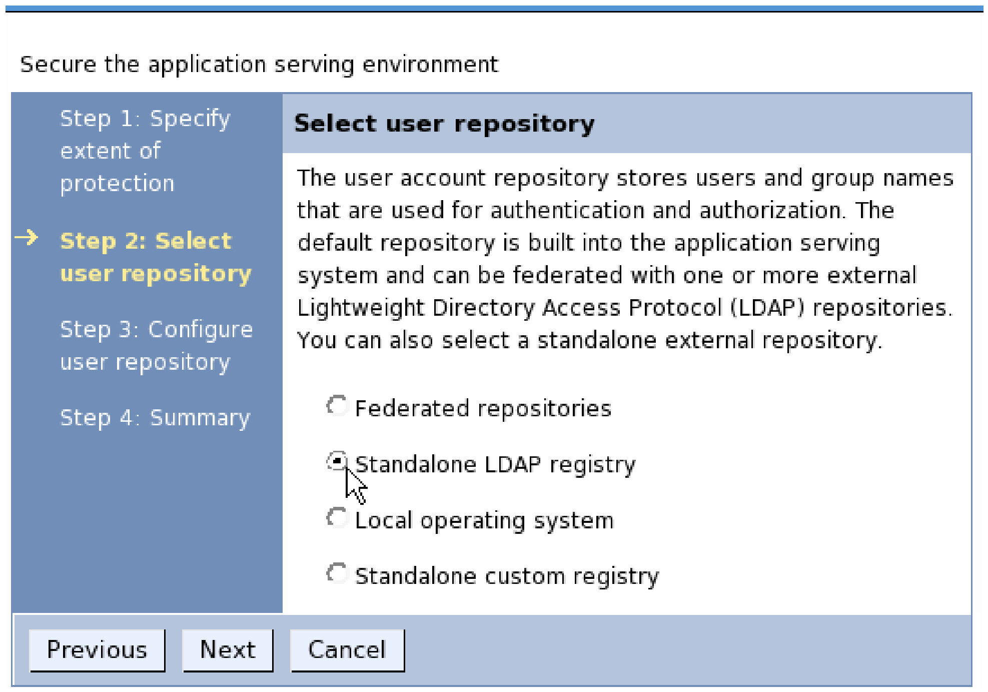
Secure administration, applications, and infrastructure

The application serving environment is completely secured when administration is restricted. The applications and t
applications also are secured.

Enable administrative security [✓]

____ 6. Click **Next** for Step 1 of the Security Configuration Wizard (we will not be enabling application security or Java 2 security)

____ 7. In Step 2 of the Security Configuration Wizard select **Standalone LDAP Registry** and click **Next** as shown below


Secure the application serving environment

Step 1: Specify extent of protection
→ Step 2: Select user repository
Step 3: Configure user repository
Step 4: Summary

Select user repository

The user account repository stores users and group names that are used for authentication and authorization. The default repository is built into the application serving system and can be federated with one or more external Lightweight Directory Access Protocol (LDAP) repositories. You can also select a standalone external repository.

- Federated repositories
- [✓ Standalone LDAP registry]
- Local operating system
- Standalone custom registry

____ 8. Specify the following values as shown below

__ a. **jdoe9** as the Primary Administrative User

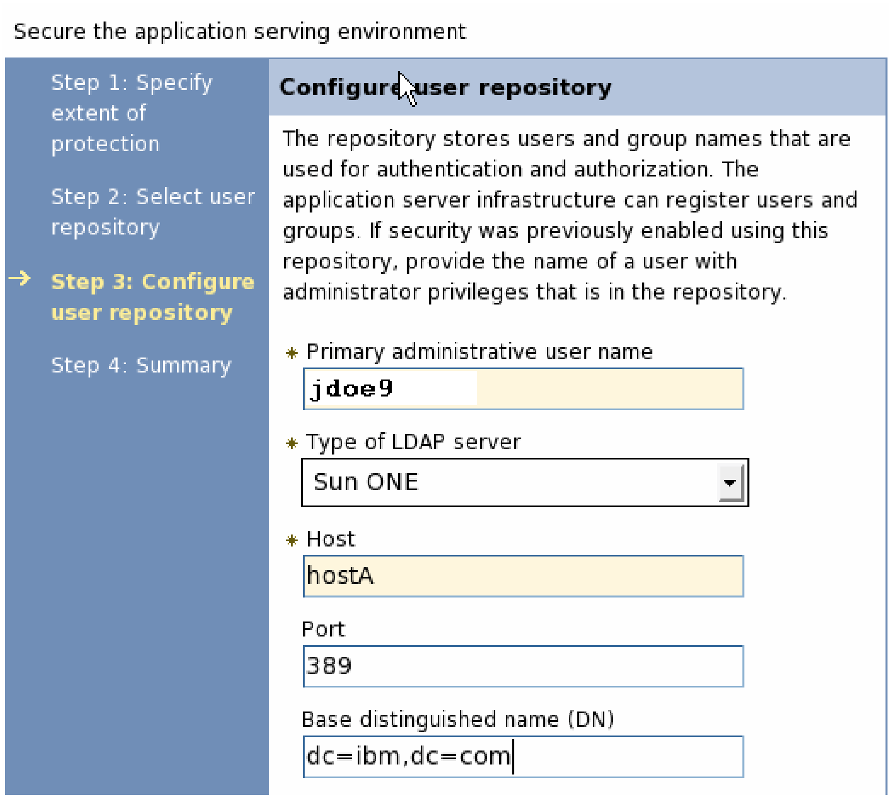
__ b. **Sun ONE** as the Type of LDAP Server

.

NOTE: We are **not** actually using Sun ONE as the LDAP server. Open LDAP, which is included with the SUSE Linux distribution, is being used. In order to simplify the configuration of LDAP in WAS, the LDAP structure created looks like Sun ONE to WAS If we hadn't created the directory to look like Sun ONE we would have needed to perform additional configuration in order to use Open LDAP.

__ c. **hostA** as the host

__ d. **dc=ibm,dc=com** as the Base Distinguished Name


Secure the application serving environment

Step 1: Specify extent of protection
Step 2: Select user repository
→ Step 3: Configure user repository
Step 4: Summary

Configure user repository

The repository stores users and group names that are used for authentication and authorization. The application server infrastructure can register users and groups. If security was previously enabled using this repository, provide the name of a user with administrator privileges that is in the repository.

* Primary administrative user name: jdoe9

* Type of LDAP server: Sun ONE

* Host: hostA

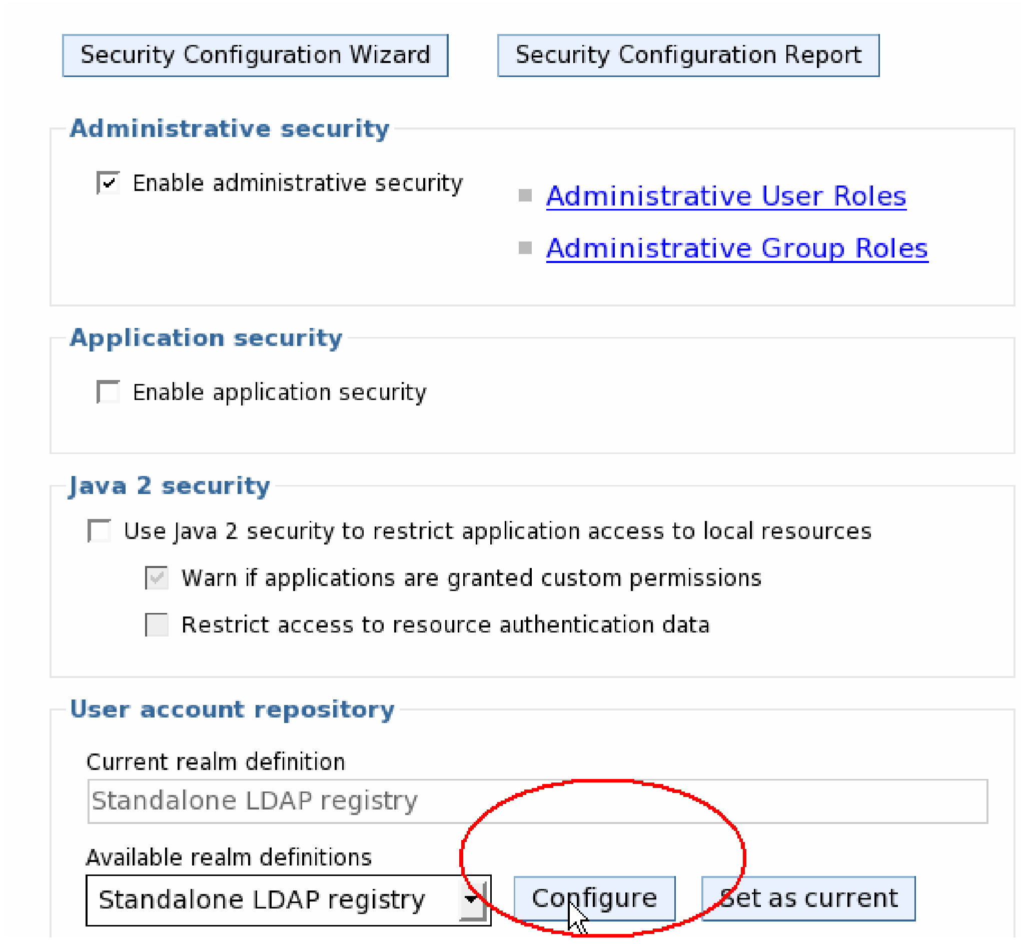
Port: 389

Base distinguished name (DN): dc=ibm,dc=com

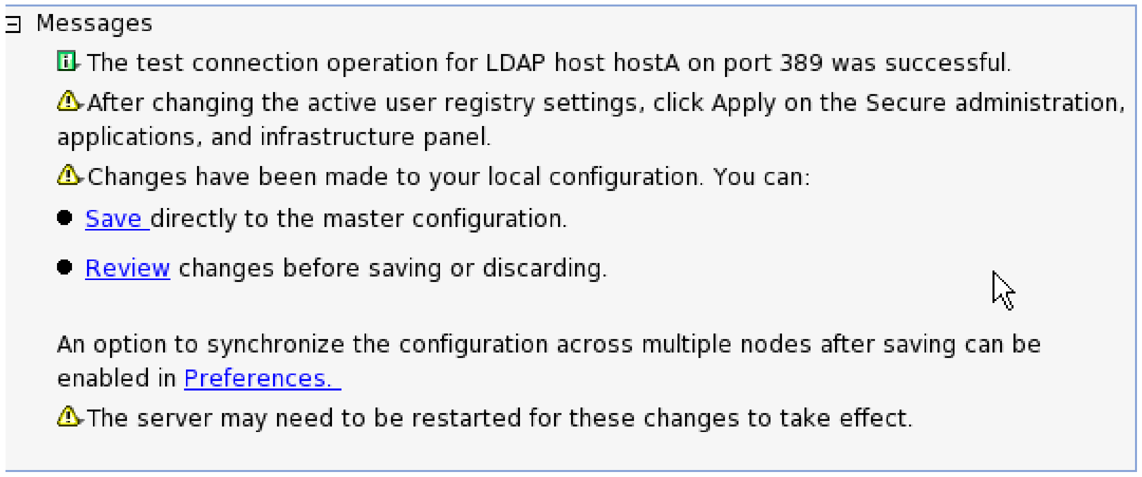
__ e. Click **Next**

____ 9. Click **Finish** on the Summary page for the Security Configuration Wizard

____ 10. Click **Configure** for the Standalone LDAP Registry as shown below


Security Configuration Wizard
Security Configuration Report
Administrative security
Enable administrative security
Administrative User Roles
Administrative Group Roles
Application security
Enable application security
Java 2 security
Use Java 2 security to restrict application access to local resources
Warn if applications are granted custom permissions
Restrict access to resource authentication data
User account repository
Current realm definition
Standalone LDAP registry
Available realm definitions
Standalone LDAP registry
Configure
Set as current

____ 11. Click **Test Connection,** you should see results as shown below indicating the test connection was successful. If this is not the case then you will need to validate your entries for LDAP from Step 8



Messages

The test connection operation for LDAP host hostA on port 389 was successful.

After changing the active user registry settings, click Apply on the Secure administration, applications, and infrastructure panel.

Changes have been made to your local configuration. You can:

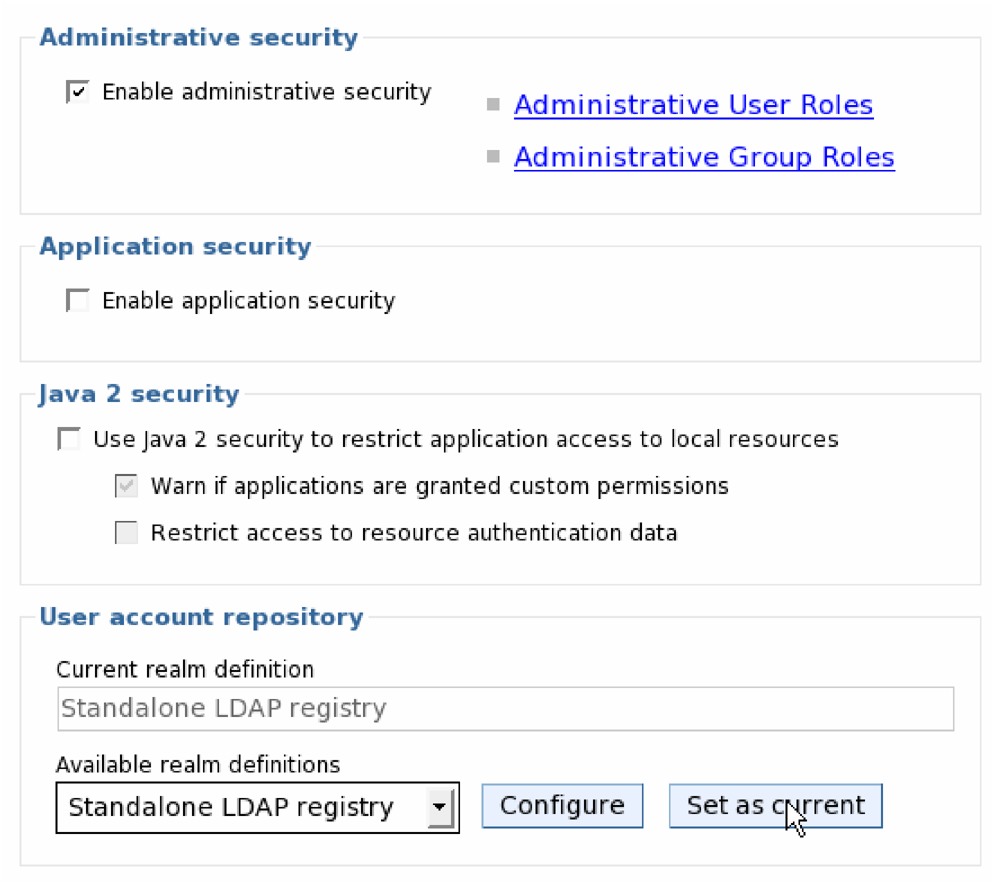
- Save directly to the master configuration.
- Review changes before saving or discarding.

An option to synchronize the configuration across multiple nodes after saving can be enabled in Preferences.

The server may need to be restarted for these changes to take effect.

____ 12. Click **OK** at the bottom of the page to return to the Security Configuration Dialog

____ 13. Click **Set as current** as shown below. Doing so changes the current registry from the file based registry to the LDAP registry we just configured.



Administrative security

[✓] Enable administrative security

- Administrative User Roles
- Administrative Group Roles

Application security

[ ] Enable application security

Java 2 security

[ ] Use Java 2 security to restrict application access to local resources

[✓] Warn if applications are granted custom permissions

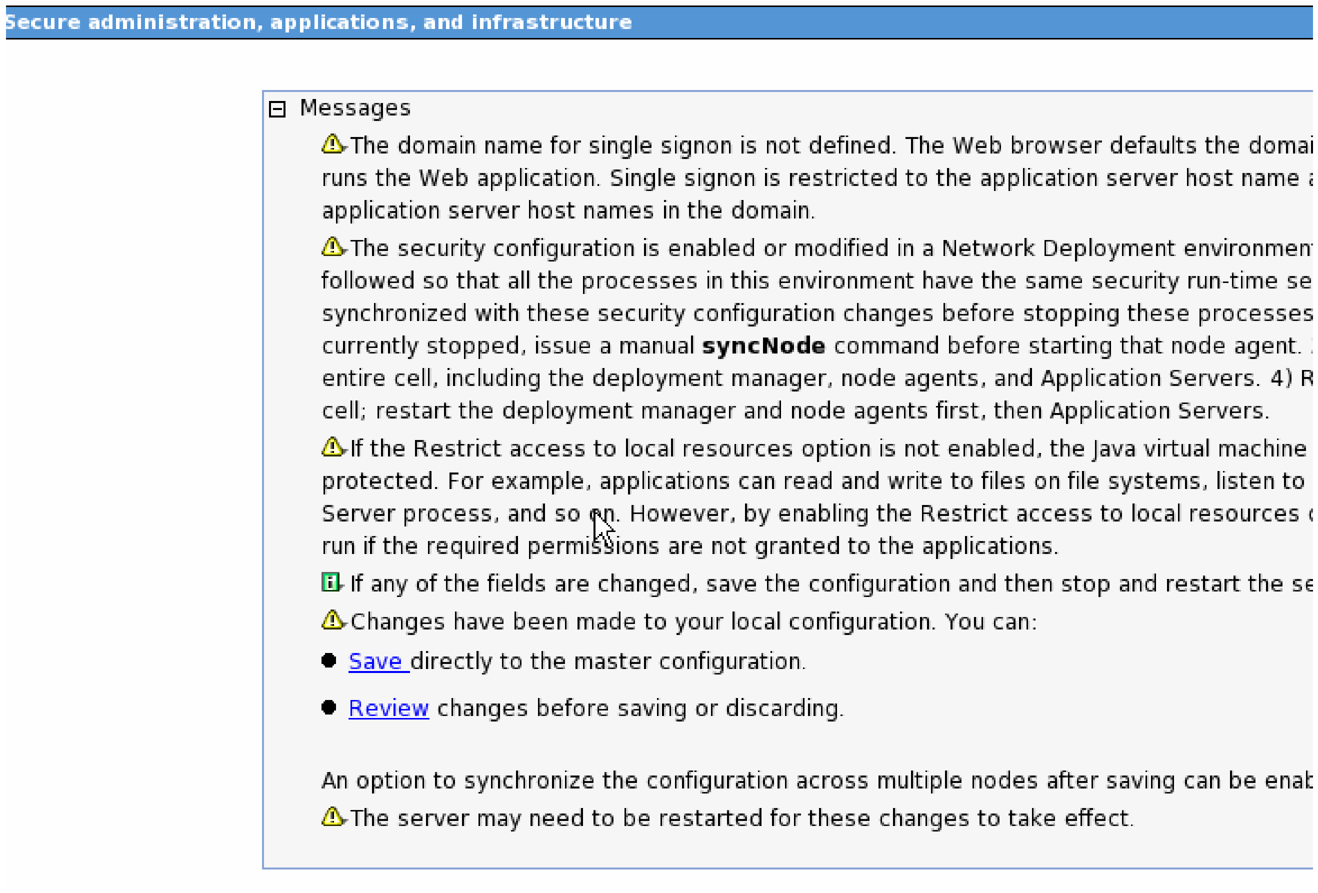
[ ] Restrict access to resource authentication data

User account repository

Current realm definition: Standalone LDAP registry

Available realm definitions: Standalone LDAP registry

____ 14. Click **Apply**, you should see messages as shown below.


Secure administration, applications, and infrastructure

⊟ Messages

⚠ The domain name for single signon is not defined. The Web browser defaults the domai
runs the Web application. Single signon is restricted to the application server host name a
application server host names in the domain.

⚠ The security configuration is enabled or modified in a Network Deployment environmen
followed so that all the processes in this environment have the same security run-time se
synchronized with these security configuration changes before stopping these processes
currently stopped, issue a manual **syncNode** command before starting that node agent.
entire cell, including the deployment manager, node agents, and Application Servers. 4) R
cell; restart the deployment manager and node agents first, then Application Servers.

⚠ If the Restrict access to local resources option is not enabled, the Java virtual machine
protected. For example, applications can read and write to files on file systems, listen to
Server process, and so on. However, by enabling the Restrict access to local resources o
run if the required permissions are not granted to the applications.

ℹ If any of the fields are changed, save the configuration and then stop and restart the se

⚠ Changes have been made to your local configuration. You can:

- Save directly to the master configuration.
- Review changes before saving or discarding.

An option to synchronize the configuration across multiple nodes after saving can be enab

⚠ The server may need to be restarted for these changes to take effect.

____ 15. **Save** your changes and **synchronize** your changes with the node.

.

NOTE: If you don't' synchronize the cell configuration at this point, when you attempt to restart the Node Agent it will not have the correct security configuration and will not start. You can correct this once the Deployment Manage is restarted by executing the command line **./synchNode.sh hostA** from the node agent profile bin directory. When you do so, you will be prompted for a userid and password. Use **jdoe9** and **jdoe9**, since that is the userid and password for the administrative user we're now using from the LDAP directory. Do not use jdoe10, since that user is in the file based registry which is not longer in use.

____ 16. Log out of the browser.

____ 17. Stop and start the Deployment Manager

__ a. Issue the command **./stopManager.sh** from the **/opt/IBM/WAS61/AppServer/profiles/Dmgr01/bin** directory

__ b. When prompted for a user ID use **jdoe10** and **jdoe10** for the password.

NOTE: jdoe10 is used since the current running process was booted using the file based registry. After we stop the deployment manager and node agent using jdoe10 the file based registry will no longer be used.

__ c. Issue the the command **./startManager.sh** from the **/opt/IBM/WAS61/AppServer/profiles/Dmgr01/bin** directory

____ 18. Stop and start the Node Agent

__ a. Issue the command **./stopNode.sh** from the **/opt/IBM/WAS61/AppServer/profiles/AppSvr01/bin** directory

__ b. When prompted for a user ID use **jdoe10** and **jdoe10** for the password.

__ c. Issue the command **./startNode.sh** from the **/opt/IBM/WAS61/AppServer/profiles/AppSvr01/bin** directory

____ **19.** Open Firefox and log in as the user ID **jdoe9** with the password **jdoe9.** Recall that you're no longer using the file based registry which had the user ID **jdoe10**.

____ **20.** Log out and close the browser.

Part 2: Start wsadmin

____ 1. Start the wsadmin environment.

__ a. Open a command shell and cd to the bin directory for the deployment manager.

cd /opt/IBM/WAS61/AppServer/profiles/Dmgr1/bin.

__ b. Start a wsadmin shell with the command

```
wsadmin –user jdoe9 –password jdoe9
```

NOTE: Optionally you can leave off the user and password information and a use jdoe9 for the user ID and password when prompted. .

__ c. If this is the first time that you've invoked the wsadmin client with a secured deployment manager, you will see an SSL signer exchange prompt that allows you to add the signer to the trust store. If this prompt appears, reply with **y** to add the signer to the trust store.

__ d. Check to see if the DefaultApplication is installed with the following command

$AdminApp list

__ e. If the DefaultApplication is installed, uninstall it using the commands

$AdminApp uninstall DefaultApplication

$AdminConfig save

NOTE: The reason for uninstalling the DefaultApplication and reinstalling it is to make sure it's installed on the server that corresponds to the ResourceGroups that will be created later in this lab.

__ f. Install the DetaultApplication using installInteractive command as shown below

$AdminApp installInteractive /opt/IBM/WAS61/AppServer/installableApps/DefaultApplication.ear

Accept the Defaults EXCEPT for the Application name and module to server mapping

__1) Enter **DefaultApplication** for the Application name as shown below

Application name: [DefaultApplication.ear]: **DefaultApplication**
Setting "Application name" to "DefaultApplication"

__2) Enter **[WebSphere:cell=hostACell01,node=hostANode01,server=STEWServer1]** as the server name for the Increment Enterprise Java Bean Module as shown below

__3) Enter **[WebSphere:cell=hostACell01,node=hostANode01,server=STEWServer1]** as the server name for the Default Web Module was shown below

```
Task[2]: Selecting servers

Specify targets such as application servers or clusters of application servers where you want
to install the modules that are contained in your application. Modules can be installed on the
same application server or dispersed among several application servers.
Also, specify the Web servers as targets that serve as routers for requests to this application.
The plug-in configuration file (plugin-cfg.xml) for each Web server is generated based on the
applications that are routed through.

Module:  Increment Enterprise Java Bean
URI:  Increment.jar,META-INF/ejb-jar.xml
Server:  [WebSphere:cell=hostACell01,node=hostACellManager01,server=dmgr]:
[WebSphere:cell=hostACell01,node=hostANode01,server=STEWServer1]
Setting "Server" to "WebSphere:cell=hostACell01,node=hostANode01,server=STEWServer1"
Module:  Default Web Application
URI:  DefaultWebApplication.war,WEB-INF/web.xml
Server:  [WebSphere:cell=hostACell01,node=hostACellManager01,server=dmgr]:
[WebSphere:cell=hostACell01,node=hostANode01,server=STEWServer1]
Setting "Server" to "WebSphere:cell=hostACell01,node=hostANode01,server=STEWServer1"
```

__4) Continue to accept the default values until the application is installed as shown below

```
ADMA5016I: Installation of DefaultApplication started.
ADMA5058I: Application and module versions are validated with versions of deployment
targets.
ADMA5005I: The application DefaultApplication is configured in the WebSphere Application
Server repository.
ADMA5053I: The library references for the installed optional package are created.
ADMA5005I: The application DefaultApplication is configured in the WebSphere Application
Server repository.
ADMA5001I: The application binaries are saved in
/opt/IBM/WAS61/AppServer/profiles/Dmgr01/wstemp/Script10bcf7fa9ec/workspace/cells/host
ACell01/applications/DefaultApplication.ear/DefaultApplication.ear
ADMA5005I: The application DefaultApplication is configured in the WebSphere Application
Server repository.
SECJ0400I: Successfuly updated the application DefaultApplication with the
appContextIDForSecurity information.
ADMA5011I: The cleanup of the temp directory for application DefaultApplication is complete.
ADMA5013I: Application DefaultApplication installed successfully.
```

__ g. Save the configuration with the command

$AdminConfig save

__ h. Keep this command prompt open because we will be using this wsadmin scripting environment to complete the rest of the lab. From this point on, all of the commands in the lab should be run in this scripting environment, unless otherwise noted.

Part 3: Create authorization groups

____ 2. Create the authorization groups **server1group** and **server2group**.

__ a. In these steps, we will be creating authorization groups; we will be adding resources to these authorization groups later. Only a user ID with cell wide administrative security has the ability to create, add, or change authorization groups.

NOTE: To see help for the authorization group commands, enter the following command at the wsadmin prompt: `$AdminTask help AuthorizationGroupCommands`

__ b. Create the new authorization group server1group.

```
$AdminTask createAuthorizationGroup {-authorizationGroupName
server1group}
```

__ c. Create the new authorization group server2group.

```
$AdminTask createAuthorizationGroup {-authorizationGroupName
server2group}
```

____ 3. Verify that the correct authorization groups have been created.

```
$AdminTask listAuthorizationGroups
```

The output from the command should list server1group and server2group.

____ 4. Save the configuration

```
$AdminConfig save
```

NOTE: As an alternative to entering the commands as shown in a wsadmin shell, you might find it quicker to invoke wsadmin using the –c (command) option, which will allow you to recall the command in the Linux shell using the up arrow key to recall, using the left and right arrow keys to navigate to the portion of text to be changed and then only changing the portion of the command that needs to change. **Using the "-c" option as show is recommended; it is much easier to correct mistakes and will allow you to complete the lab much faster**. Below is the output from doing so.

hostA:/opt/IBM/WAS61/AppServer/profiles/Dmgr01/bin # **./wsadmin.sh -c '$AdminTask createAuthorizationGroup { -authorizationGroupName server1group } ' -username jdoe9 -password jdoe9**

WASX7209I: Connected to process "dmgr" on node hostACellManager01 using SOAP connector; The type of process is: DeploymentManager
cells/hostACell01/authorizationgroups/server1group|authorizationgroup.xml#AuthorizationGroup_1150230572408

hostA:/opt/IBM/WAS61/AppServer/profiles/Dmgr01/bin # **./wsadmin.sh -c '$AdminTask createAuthorizationGroup { -authorizationGroupName server2group } ' -username jdoe9 -password jdoe9**

WASX7209I: Connected to process "dmgr" on node hostACellManager01 using SOAP connector; The type of process is: DeploymentManager
cells/hostACell01/authorizationgroups/server2group|authorizationgroup.xml#AuthorizationGroup_1150230614390

```
hostA:/opt/IBM/WAS61/AppServer/profiles/Dmgr01/bin # ./wsadmin.sh -c '$AdminTask listAuthorizationGroups ' -username jdoe9 -password jdoe9

 WASX7209I: Connected to process "dmgr" on node hostACellManager01 using SOAP connector;  The type of process is: DeploymentManager

server1group

server2group

hostA:/opt/IBM/WAS61/AppServer/profiles/Dmgr01/bin # ./wsadmin.sh -c '$AdminConfig save ' -username jdoe9 -password jdoe9

WASX7209I: Connected to process "dmgr" on node hostACellManager01 using SOAP connector;  The type of process is: DeploymentManager
```

Part 4: Configure authorization groups

Now that we've created our authorization groups, we can add resources to those authorization groups and assign users to specific roles. .

____ 1. Add server resources to the activation groups.

__ a. The resources that we are configuring security for are the servers in the cell. Add the **STEWServer1** server to **server1group** and the second server **STEWServer2** to **server2group**. Use the following command to add the resource **STEWServer1** to **server1group**.

```
$AdminTask addResourceToAuthorizationGroup {-authorizationGroupName
server1group -resourceName Server=STEWServer1 }
```

__ b. Use the following command to add the resource **STEWServer2** to **server2group**.

```
$AdminTask addResourceToAuthorizationGroup {-authorizationGroupName
server2group -resourceName Server=STEWServer2}
```

____ 2. Assign users roles in the activation groups.

__ a. We will be assigning two roles to each of the server administrator users that exist in the LDAP directory. Assigning a user the **administrator** role allows that user to manage the resources in the activation group; assigning the user the **adminsecuritymanager** role allows the user to manage the roles of other users in the activation group. The following two commands will assign both the **administrator** and the **adminsecuritymanager** roles to **srv1admin** for **server1group**.

```
$AdminTask mapUsersToAdminRole {-authorizationGroupName server1group
-roleName administrator -userids jdoe1}


$AdminTask mapUsersToAdminRole {-authorizationGroupName server1group
-roleName adminsecuritymanager -userids jdoe1}
```

__ b. The following two commands will assign both the **administrator** and the **adminsecuritymanager** roles to **srv2admin** for **server2group**.

```
$AdminTask mapUsersToAdminRole {-authorizationGroupName server2group
-roleName administrator -userids jdoe2}


$AdminTask mapUsersToAdminRole {-authorizationGroupName server2group
-roleName adminsecuritymanager -userids jdoe2}
```

__ c. In this step, we will give the **celladmin** user the authority to manage all of the resources and activation groups in the cell. This is accomplished by giving the user special authorities in the **CellActivationGroup**. Note that we did not have to create this activation group; it already existed as a part of the cell. As above, we will give the **jdoe3** user both **administrator** and **adminsecuritymanager** authority. The **adminsecuritymanager** role is required for the **celladmin** user to be able to manage authorization groups. Use these two commands to configure the roles for the **jdoe3** user:

```
$AdminTask mapUsersToAdminRole {-authorizationGroupName
CellAuthorizationGroup -roleName administrator -userids jdoe3}


$AdminTask mapUsersToAdminRole {-authorizationGroupName
CellAuthorizationGroup -roleName adminsecuritymanager -userids jdoe3
}
```

____ 3. Add the default application to server1group.

__ a. In order for a user to be able to manage an application installed on a server, that application resource needs to be added to the authorization group. It is not enough just for the server resource to be a part of the authorization group; the application must be explicitly added. Applications do not inherit security roles from the server on which they are deployed.

__ b. Add the default application to the **server1group** authorization group using the following command:

```
$AdminTask addResourceToAuthorizationGroup {-authorizationGroupName
server1group -resourceName Application=DefaultApplication}
```

____ 4. Save the configuration to the master repository.

```
$AdminConfig save
```

____ 5. Refresh the memory on all running servers.

NOTE: The commands in this step cannot be run with a –c option in wsadmin, if you have been using the wsadmin –c method shown above, you will need to open a wsadmin shell for the following two commands, then you can exit wsadmin for the remainder of the exercise and use wsadmin –c ' <commands>".

__ a. After making any changes to authorization groups, the AuthorizationGroupManager must be notified. This can be done while a server is running without requiring a restart. First, submit a query to gather the information on what needs to be refreshed.

```
set a [$AdminControl queryNames
type=AuthorizationGroupManager,process=dmgr,*]
```

__ b. Next, perform the actual refresh, based on the query results.

```
$AdminControl invoke $a refreshAll
```

____ 6. Verify the authorization group configuration.

__ a. Check to see that the user **jdoe1** has authorization to the appropriate resources. You should see that srv1admin has both **adminsecuritymanager** and **administrator** authority to: the server1group authorization group, the server STEWServer1, and the default application installed on that server.

```
$AdminTask listResourcesForUserID {-userid jdoe1 }
```

__ b. Verify that the **server1group** contains the appropriate resources. It should contain three resources: the authorization group itself, the server, and the default application.

```
$AdminTask listResourcesOfAuthorizationGroup {-authorizationGroupName
server1group}
```

__ c. Check to see that the user **jdoe2** has authorization to the appropriate resources. You should see that jdoe2 has both **adminsecuritymanager** and **administrator** authority to the server2group authorization group and the server server2. Notice that jdoe2 does not have any administrative privileges for the default application because we did not add the default application to the server2group authorization group.

```
$AdminTask listResourcesForUserID {-userid jdoe2}
```

Part 5: Test fine-grained administrative security settings

____ 1. Use wsadmin commands to manage application servers.

__ a. Use the following command to list the servers to which jdoe3 is authorized. This ID should be able to list all of the servers in the cell, including the deployment manager, the node agent, and the two application servers.

```
wsadmin –user jdoe3 –password jdoe3  -c ‘$AdminConfig list Server’
```

__ b. Use the following command to list the servers to which **jdoe1** is authorized. The only server that should show up in the list is **STEWServer1**.

```
wsadmin –user jdoe1 –password jdoe1  -c ‘AdminConfig list Server’
```

__ c. Similarly, use the following command to show that the only server to which **jdoe2** is authorized is **STEWServer2**.

```
wsadmin –user jdoe2 –password jdoe2 -c "$AdminConfig list Server"
```

__ d. Try to start STEWServer2 using the jdoe1 ID. This operation should fail because jdoe1 does not have the appropriate authority to start the server.

```
./wsadmin.sh –user jdoe1 –password jdoe1 –c ‘$AdminControl
startServer STEWServer2 hostANode01’
```

```
WASX7209I: Connected to process "dmgr" on node hostACellManager01 using SOAP connector;
The type of process is: DeploymentManager

WASX7015E: Exception running command: "$AdminControl startServer STEWServer2
hostANode01"; exception information:

com.ibm.ws.scripting.ScriptingException: WASX7255E: Cannot find server "STEWServer2" in
configuration data.
```

__ e. Now start STEWServer1 using the jdoe1 admin ID. This operation should complete successfully.

```
wsadmin –user jdoe1 –password jdoe1 –c ‘ $AdminControl startServer
STEWServer1 hostANode01 ‘
```

What you did in this exercise

In this lab, you performed the following activities:

- Configured WAS to use LDAP
- Created user IDs to manage portions of a cell.
- Created authorization groups, added existing resources to the authorization groups, and assigned the newly created users to specific roles in the activation groups.
- Examined security settings in the administrative console.
- Tested security settings using wsadmin commands to:
    - List authorization groups.
    - List resources within authorization groups.
    - Manage an application server, as appropriate, based on user authority.