Redpaper

Ganesan Karuppaiah

# WebSphere Application Server V7 Flexible Management

## Security Configuration Requirements

IBM® WebSphere® Application Server V7 introduces an administration function called *flexible management* that is designed to facilitate administration of many WebSphere Application Server stand-alone profiles and Federated Network Deployment topologies.

This IBM Redpaper™ publication provides an overview to a flexible management topology. It focuses on administrative security aspects in a flexible management topology that consists of WebSphere Application Server profiles, administrative agents, node agents, a deployment manager, and job managers. It also includes information about the underlying cross-realm trust, job submission in a cross-realm environment, and the Rivest-Shamir-Adleman algorithm (RSA) authentication mechanism that enables cross-cell authentication. In addition, this paper contains information about preferred practices when using a flexible management topology.

This paper is intended for experienced WebSphere system administrators who have a basic understanding of WebSphere Application Server and Network Management solutions. The discussion here assumes that you are familiar with WebSphere Application Server system management infrastructure, WebSphere security configuration, public key infrastructure (PKI), and the RSA public and private key mechanism.

This paper assumes that you have installed WebSphere Application Server V7 with the latest fix packs and have created one or more WebSphere Application Server, administrative agent, deployment manager, and job manager profiles. Also, this paper assumes that you have more than one LDAP server that is available for configuring WebSphere global security.

**Testing note:** The sample commands included in this paper were tested in WebSphere V7.0.0.15 across multiple operating systems.

## Flexible management overview

With the *flexible management* topology, you can manage applications, modify configurations, and control the application server run time. Flexible management complements rather than replaces any existing WebSphere Application Server and Network Deployment scenarios using the administrative agent and job manager processes. Both processes are management type profiles that are created by selecting the Management profile option.

The administrative agent (also known as the *admin agent*) is a WebSphere Application Server offering. To create and enable an administrative agent profile, you do not need WebSphere Application Server Network Deployment. However, the job manager is available only from the WebSphere Application Server Network Deployment offering.

Flexible management uses the RSA authentication mechanism, which is an authentication mechanism that is available in WebSphere Application Server V7. It is not available in earlier versions of WebSphere Application Server, and WebSphere Application Server V7 does not support profiles from earlier versions in flexible management.

When using flexible management, keep in mind the following version notes:

- If V6 or V6.1 nodes are federated with a V7 deployment manager cell, the V7 job manager can submit jobs to V6 or later nodes using the V7 deployment manager. Certain restrictions might apply to job types based on the target nodes. For more information, see the IBM WebSphere Application Server Information Center:

  `http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/index.jsp`
- Make sure that the nodes that you want the administrative agent to manage have the same products as the administrative agent and that the products are at the same version levels on these nodes as the administrative agent. Starting with V7.0.0.13, this requirement is enforced because the administrative agent must have a matching environment to handle all of the administrative capabilities of the registered node. A node is not allowed to register with an administrative agent unless that node has an identical set of products and versions.
- If you were previously running on V7.0.0.11 or earlier and have an administrative agent with a managed node that has mismatched products or versions, when you apply V7.0.0.13 or later to the administrative agent, that administrative agent cannot start the subsystem for any mismatched nodes. You must update these nodes to have the same products and versions as the administrative agents. Restart the servers on the node and then restart the administrative agent, before the administrative agent can resume managing these registered nodes.

## Authentication mechanism overview

*Authentication* is the process of establishing whether a client is who or what it claims to be in a particular context. A client can be a user, a computer, or an application. Authentication mechanisms in WebSphere Application Server typically collaborate closely with a user registry.

In the deployment manager topology environment, all the federated WebSphere Application Server profiles become part of the deployment manager cell. All the nodes in the deployment manager environment share the same cell name and LTPA keys. However, in the flexible management topology, all the participating nodes retain their original cell-level configuration. For example, the job manager cell, deployment manager cell, administrative agent cell, and the registered WebSphere Application Server cells are all individual cells. Even after registerNode or registerWithJobManager operations, all the cells retain their original cell name, Lightweight Third Party Authentication (LTPA) keys, and security configurations. The cells are always individual, separate, and independent cells. At any time, you can use the deregisterNode or unregisterWithJobManager operation to separate the WebSphere Application Server profile from the administrative agent and the job manager without any configuration loss.

Establishing an authentication mechanism between disparate cells in a flexible management topology can be a challenging task, because LTPA key share is not used for cross-cell authentication as it is with a deployment manager topology. The change in an LTPA key in a cell affects the trust relation in the entire topology. Also, no centralized configuration or synchronization function is available between the nodes in a flexible management topology.

## Introduction to the RSA authentication mechanism

The RSA authentication mechanism used with WebSphere Application Server in a flexible management topology is not an RSA SecureId Token mechanism. Instead, this RSA authentication mechanism provides a way for WebSphere Application Server management servers (the deployment manager, administrative agent, node agent, and job manager) to authenticate with each other before sending messages. This mechanism is an alternative to current LTPA or Kerberos models and is different from LTPA where keys are shared and, if one side changes, all sides need to change.

With WebSphere Application Server V7, this RSA authentication mechanism simplifies the security environment for a flexible management topology and provides advanced security features. In the current LTPA model, all the WebSphere Application Server management servers in a cell need to share the same LTPA keys. With RSA token authentication, after the RSA root signer certificate (with a 20-year lifetime) is exchanged between two administrative processes, no other synchronization of security information is needed between disparate profiles in order for the security tokens to be trusted.

**WebSphere Application Server profiles:** The RSA token authentication mechanism aids the flexible management objective to preserve the application server profiles configurations and to isolate these profiles from a security perspective. This mechanism permits the application server profiles that are managed by an administrative agent to have different LTPA keys, user registries, and administrative users.

The RSA token authentication mechanism uses PKI and RSA public and private key to establish trust and for certificate signing. It supports the ability to register new servers to the flexible management topology securely and easily.

The RSA token authentication mechanism has its own root certificates and personal certificates. Do not confuse RSA certificates with regular SSL certificates that are used for the AppServer end-point security. RSA certificates are not the same as regular SSL certificates and exist for a different purpose. The RSA root certificate and RSA personal certificates are created during profile creation for each profile and are used only between WebSphere nodes.

Also, the RSA token authentication mechanism uses a separate keystore and truststore that are different from the SSL keystore and truststore. For each profile, a root certificate is stored in the `rsatoken-root-key.p12` keystore. This RSA root certificate signs the RSA personal certificate that is stored in the `rsatoken-key.p12` keystore. Trusted signers from other realms are stored in the `rsatoken-trust.p12` truststore.

The RSA root signer certificate (20-year lifetime) is exchanged automatically between two administrative processes during the following operations. This process is an important process and is how the trust between the WebSphere nodes is established:

- registerNode

  RSA root signers are exchanged between the WebSphere Application Server profile and the registering administrative agent profile.

- AdminTask.registerWithJobManager

  RSA root signers are exchanged between the job manager and the registering nodes as follows:

  - Between the job manager and the deployment manager in a federated deployment manager topology
  - Between the job manager and the administrative agent in an administrative agent topology

- deregisterNode and AdminTask.unregisterWithJobManager

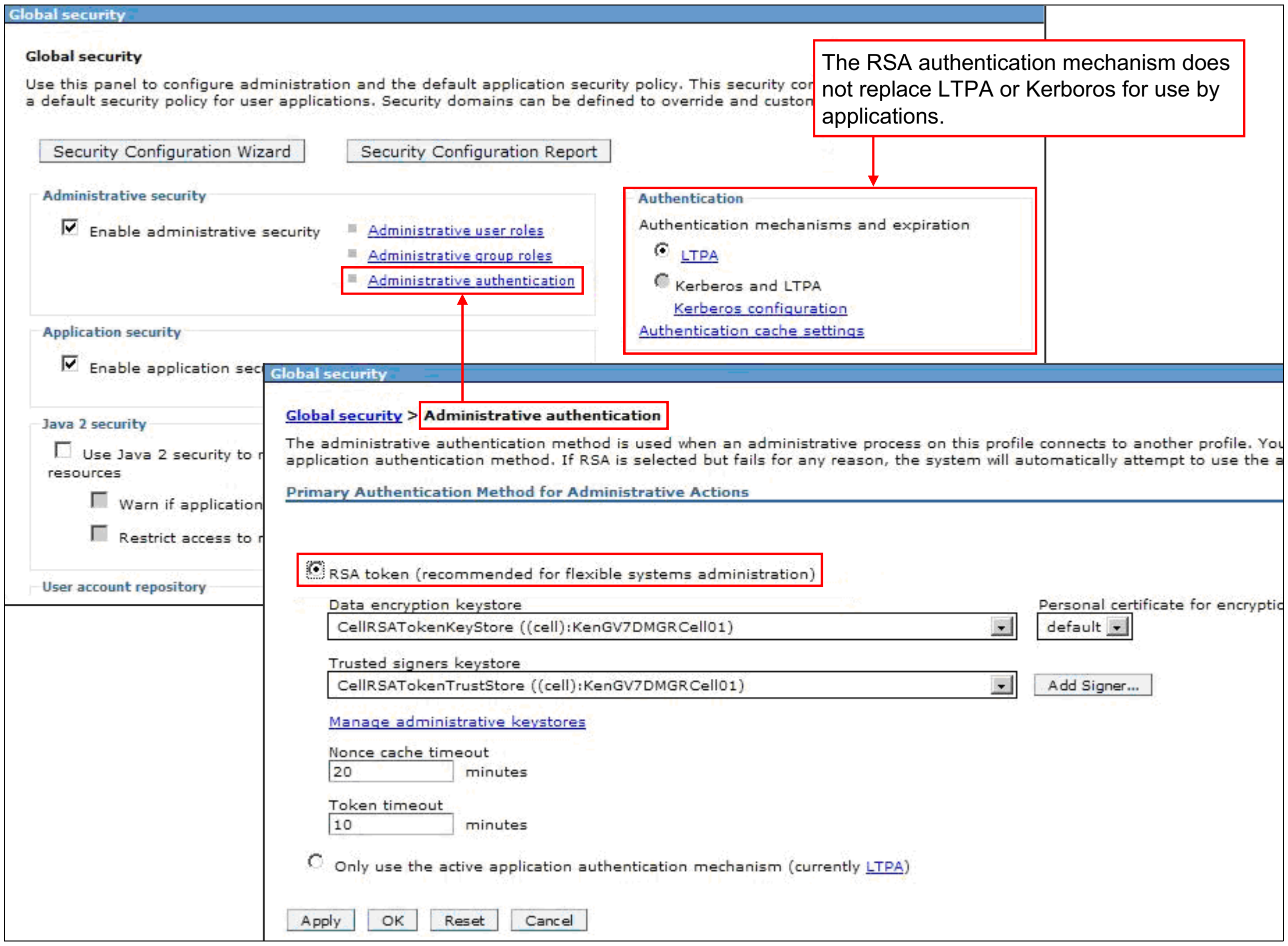
  Removes the signers automatically from the respective nodes' RSA truststores.

**RSA note**: RSA does not use the same certificates that are used by SSL.

## Enabling RSA administrative authentication

Use the RSA *authentication mechanism* for flexible systems administration. The job manager, administrative agent, and WebSphere Application Server profiles are enabled with RSA as a default administrative authentication mechanism. Deployment manager profiles are not RSA enabled by default. The deployment manager uses the LTPA or the active application authentication mechanism (LTPA or Kerberos) as the default administrative authentication mechanism.

As a preferred practice and to explore the RSA administrative authentication mechanism across the entire topology, use the RSA authentication mechanism for the deployment manager profile as well. Because it is not enabled by default, before you add the deployment manager to a flexible management topology (that is, before you add the deployment manager to the job manager or job managers), enable the RSA authentication mechanism as an administrative authentication mechanism (Figure 1). Select **Administrative console** → **Security** → **Global security**. Then save and synchronize the changes, and restart the deployment manager.


Global security
Security Configuration Wizard
Security Configuration Report
Administrative security
Enable administrative security
Administrative user roles
Administrative group roles
Administrative authentication
Application security
Enable application sec
Java 2 security
Use Java 2 security to r resources
Warn if application
Restrict access to r
User account repository
Authentication
LTPA
Kerberos and LTPA
Kerberos configuration
Authentication cache settings
Global security > Administrative authentication
Primary Authentication Method for Administrative Actions
Manage administrative keystores
Add Signer...
Apply
OK
Reset
Cancel

Global security

Use this panel to configure administration and the default application security policy. This security cor
a default security policy for user applications. Security domains can be defined to override and custon

Authentication mechanisms and expiration

The RSA authentication mechanism does not replace LTPA or Kerberos for use by applications.

The administrative authentication method is used when an administrative process on this profile connects to another profile. You
application authentication method. If RSA is selected but fails for any reason, the system will automatically attempt to use the a

RSA token (recommended for flexible systems administration)

Data encryption keystore: CellRSATokenKeyStore ((cell):KenGV7DMGRCell01)

Personal certificate for encryptic: default

Trusted signers keystore: CellRSATokenTrustStore ((cell):KenGV7DMGRCell01)

Nonce cache timeout: 20 minutes

Token timeout: 10 minutes

Only use the active application authentication mechanism (currently LTPA)

*Figure 1 RSA authentication mechanism*

The trust establishment for SSL and RSA are different. SSL allows trust in pure clients, whereas RSA does not. RSA is meant only for server-to-server trust. RSA certificates should not use SSL certificates and vice versa.

Changing the administrative authentication mechanism from RSA to any other active application authentication mechanism (LTPA or Kerberos) requires additional configurations and needs to be executed with caution. For example, a change in the LTPA key or the Kerberos realm will cause trust failure between the cells in a flexible management topology.

**Changing the default RSA authentication mechanism:** For the job manager, administrative agent, and WebSphere Application Server, RSA is the default administrative authentication mechanism. Do not change this authentication mechanism unless you have a strong reason.

RSA authentication process starts with the client process obtaining a public key from the target server. In cases where the target RSA certificate cannot be obtained, the fallback mechanism is LTPA. The fallback occurs automatically, and no configuration is required. However, if the LTPA keys are not shared and a fallback occurs, LTPA will fail as well.

The basic goal of an RSA authentication mechanism is to convert the client subject into a subject at the target by securely propagating the information that is necessary to do so. After the subject is generated at the target, the RSA authentication mechanism process is complete.

Figure 2 illustrates the process that takes place when a request is sent from a server-as-client to a target server.

**RSA authentication:** In an RSA authentication, there is no pure client involved. RSA authentication is always between two server processes, where one server acts as a client.

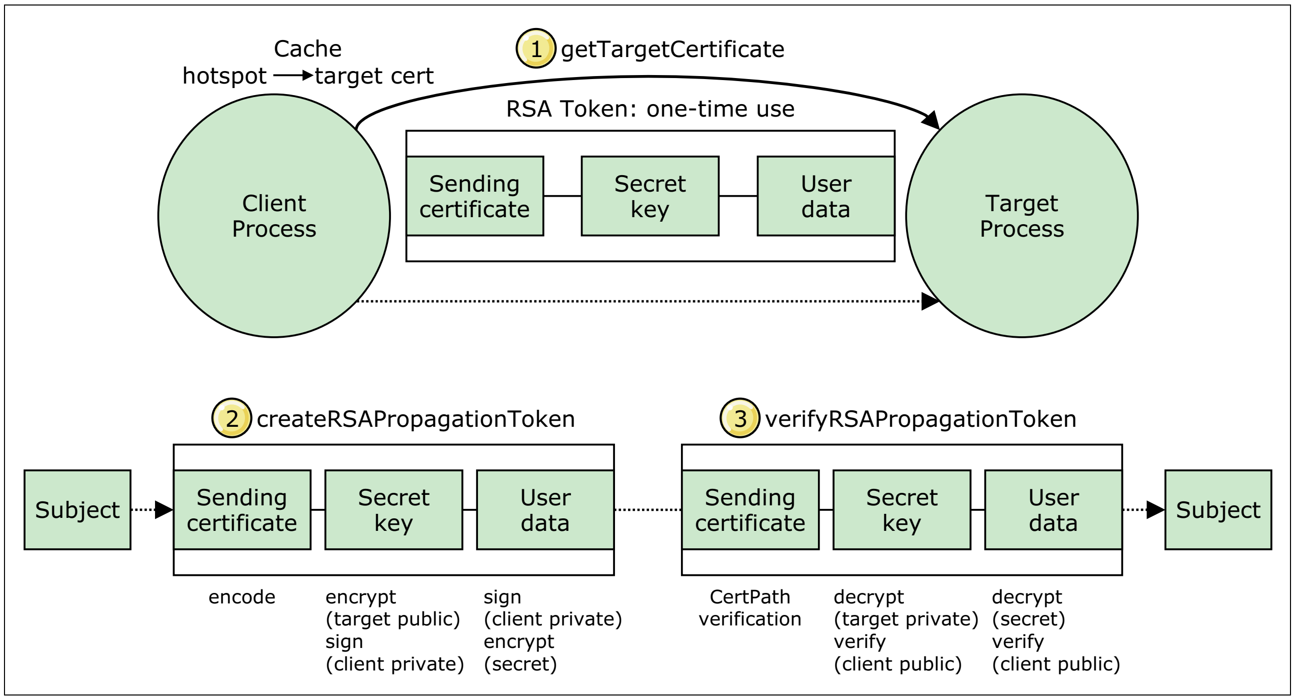


*Figure 2 RSA propagation*

The RSA authentication process helps for job submissions (job manager topology) and administrative activities (administrative agent topology). With this RSA administrative authentication, disparate profiles retain their cell-level security configuration.

## Points to consider

Consider the following points when using an RSA authentication mechanism:

- The RSA authentication mechanism is used only for server-to-server administrative authentication, such as administrative connector and file transfer requests from the administrative agent to the registered application server node.
- The RSA authentication mechanism does not replace LTPA or Kerberos for use by applications.

- Even if security is not enabled, RSA signer certificates are exchanged between the administrative agent and the registering application server profiles. During deregisterNode, RSA signer certificates are removed automatically.
- During registerWithJobManager, RSA signer certificates are exchanged between the job manager and the registering deployment manager or registering application server profiles (the administrative agent topology).
- After the subject is generated at the target, the RSA authentication mechanism process is complete.

**RSA token note:** The RSA token is not related to the RSA SecureId token. WebSphere Application Server does not support SecureId.

For more information about the RSA token authentication mechanism, see this website:

http://www14.software.ibm.com/webapp/wsbroker/redirect?version=compass&product=was-nd-dist&topic=csec_7rsa_token_auth

## Administrative agent overview

The *administrative agent* (also referred to as *admin agent*) is a management profile introduced in WebSphere Application Server V7. It is not the deployment manager topology's node agent and is not a replacement of the deployment manager (dmgr). After creating an application server profile, you have all the following options, as with the previous version of WebSphere Application Server:

- Running the base application server profile as a stand-alone server
- Using the `addNode` command to federate the base application server profile to a dmgr cell

If you run the WebSphere Application Server profile as a stand-alone server, that application server profile houses all the necessary administration services for managing and configuring the server. If you run multiple WebSphere Application Server stand-alone profiles on the same workstation, you cannot manage the multiple profiles together. However, you can federate the multiple profiles (using the `addNode` command) into the deployment manager. This method can be problematic if you do not want to federate to the deployment manager and want to keep the stand-alone server identity as it is with a tool to manage multiple application server nodes collectively.

In addition, if you create multiple application server profiles in a workstation, each WebSphere Application Server profile in that workstation has its own administration services and console application. This setup duplicates the administration services in every server and adds to the memory requirements and initialization time. In addition, a user cannot administer (stop or start) the server remotely.

The administrative agent addresses these challenges in the WebSphere Application Server stand-alone environment. Whereas the application sever continues to serve the application requests, the administrative agent takes care of the administrative aspects of the application server.

For more information about planning administrative agents, see this website:

http://www14.software.ibm.com/webapp/wsbroker/redirect?version=compass&product=was-nd-zos&topic=tins_planningaacell

## Registering a WebSphere Application Server stand-alone profile to an administrative agent

You can create WebSphere Application Server profiles and administrative agent profiles in a workstation using the **`manageprofiles`** command or using the Profile Management Tool.

As an example, Figure 3 illustrates the configuration of workstation A before using the **`registerNode`** command.

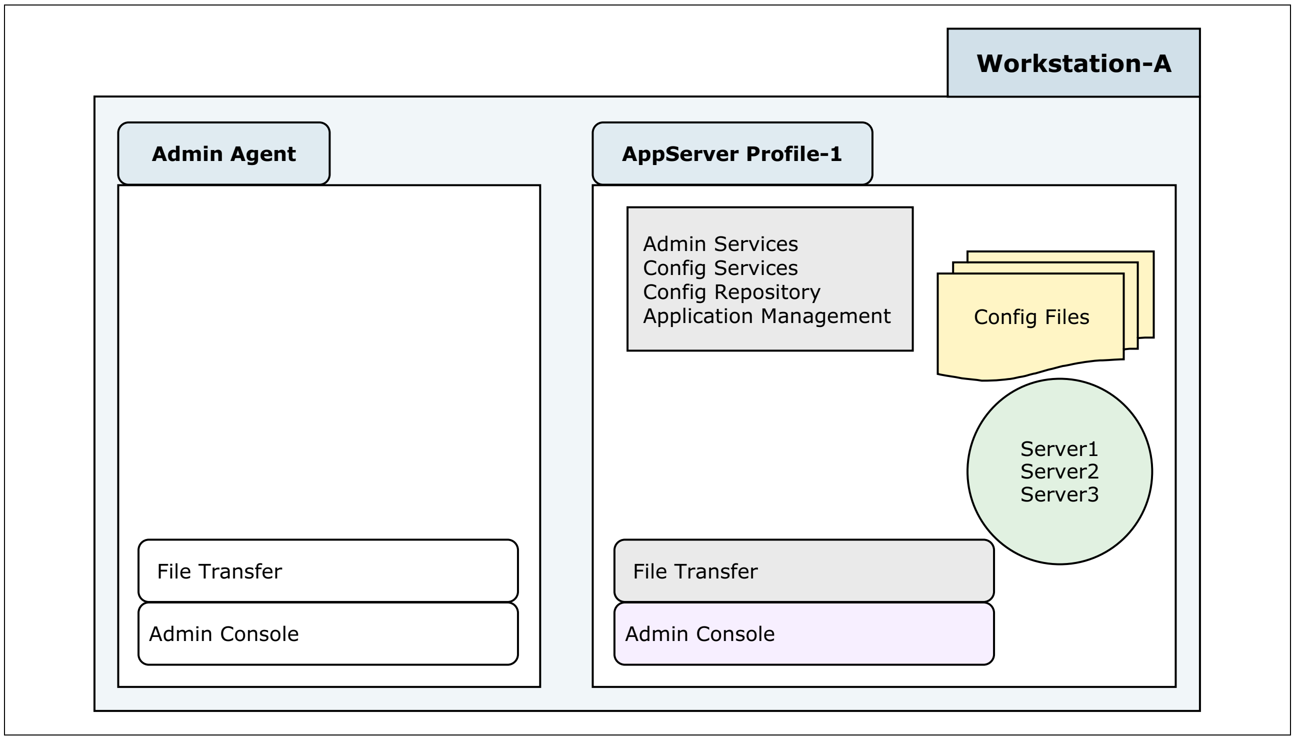


*Figure 3 Before registerNode*

After you create the WebSphere Application Server and administrative agent profiles, you can run the application server profile as a stand-alone profile, or you can register the profile to an administrative agent in that same workstation.

**Using the registerNode command:** You execute the **`registerNode`** command from the administrative agent profile's bin directory. Use the following command:

```
C:\IBM\WebSpherev7\AppServer\profiles\adminagent01\bin>registerNode -conntype
SOAP -port <SOAP Port of Admin Agent> -profilePath
C:\IBM\WebSpherev7\AppServer\profiles\AppServer01
```

When registered, the console application (the *administrative console*) for the WebSphere Application Server profile is disabled, and it will share the administrative agent's console application and other administrative services.

Even after registering the WebSphere Application Server profile with the administrative agent, the registering process preserves existing management experience as follows:

- WebSphere Application Server profiles retain their configurations.
- There is no master repository in the administrative agent or node synchronization.
- Unlike a node in a deployment manager cell, the configuration repository of a WebSphere Application Server profile that is registered with an administrative agent is *not* federated into the administrative agent's repository.

Figure 4 shows the configuration of workstation A after using the **`registerNode`** command.

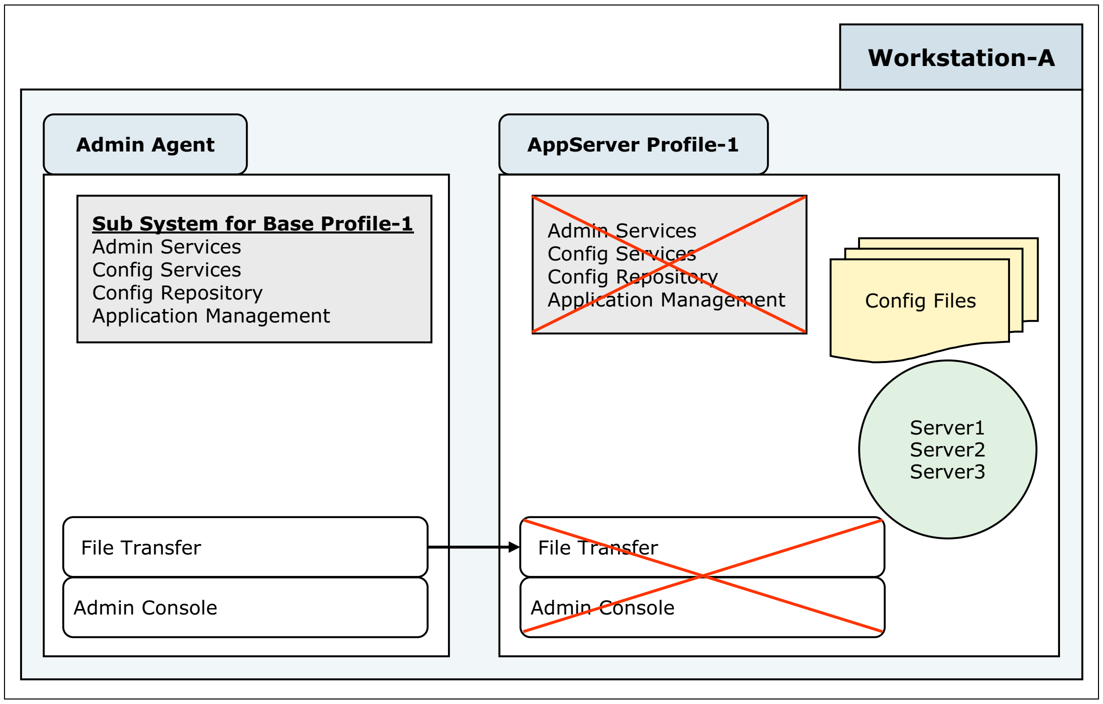


*Figure 4 After registerNode*

Administrative agent registration reduces the WebSphere Application Server footprint and migrates the following administrative functions to the administrative agent:

- Configuration management
- Application management
- Command Manager
- File transfer
- Admin console

## Registering multiple WebSphere Application Server profiles with the administrative agent

When multiple WebSphere Application Server stand-alone profiles are registered with a single administrative agent, the administrative services of the registered WebSphere Application Server profile are consolidated into the administrative agent, thus reducing the memory footprint from previously duplicated services (Figure 5).

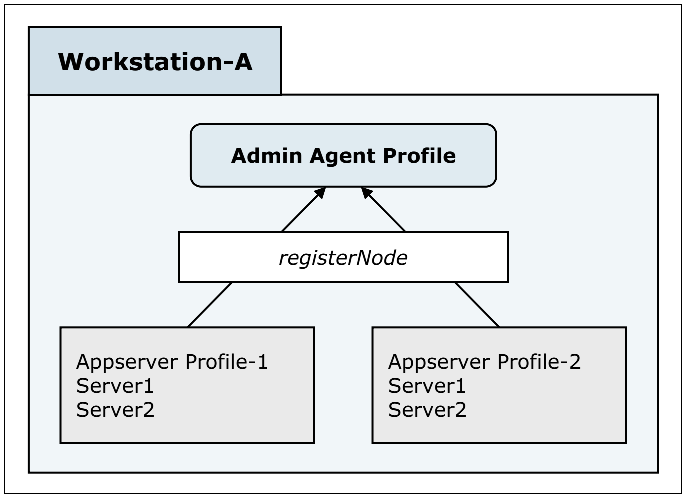


*Figure 5   Registering multiple nodes to the administrative agent*

The administrative agent becomes the central point of entry for administering multiple WebSphere Application Server stand-alone profiles on the same workstation. For every WebSphere Application Server profile that is registered with the administrative agent, an administrative subsystem is created within the administrative agent to represent the new administrative entry point for that profile. Assume a subsystem as an entry point for a WebSphere Application Server profile from the administrative agent.

## The administrative agent topology with z/OS systems

With IBM z/OS® systems, you can register WebSphere Application Server profiles from one logical partition (LPAR) with the administrative agent in another LPAR. However, you can register WebSphere Application Server profiles with only one administrative agent.

Figure 6 illustrates a z/OS system with a cross-registered LPAR.

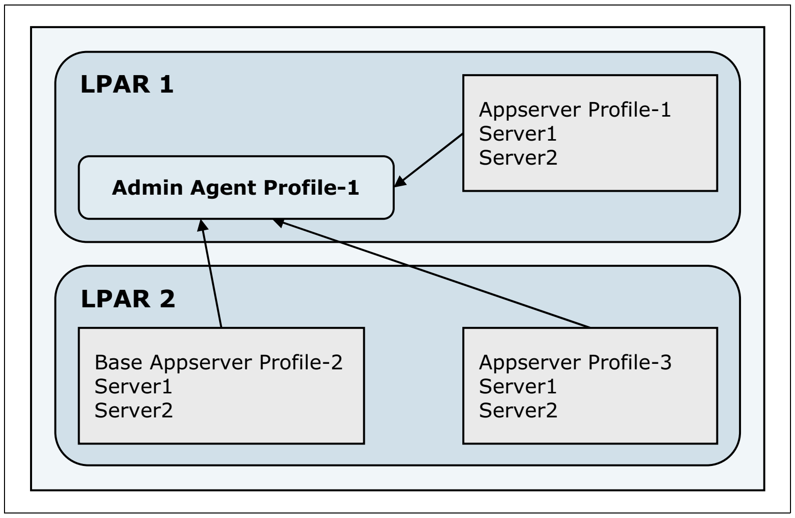


*Figure 6 z/OS system with a cross-registered LPAR*

Figure 7 illustrates the configuration of workstation A and workstation B in an administrative agent topology for a distributed operating system environment.

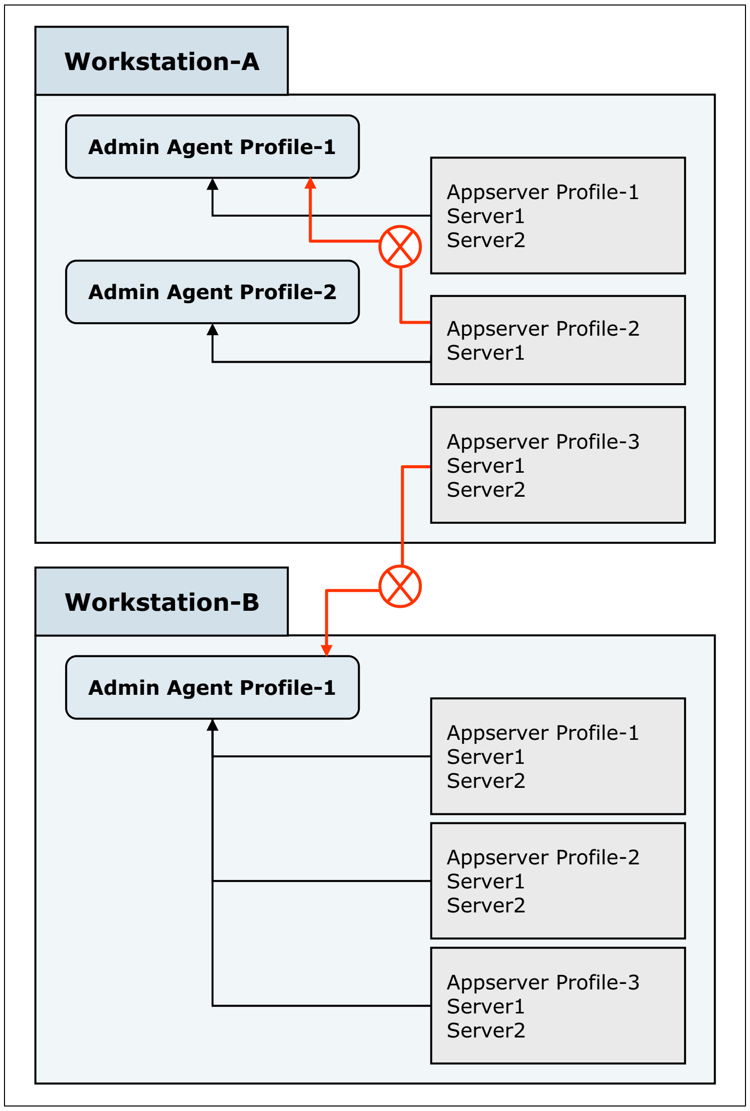


*Figure 7 Administrative agent topology in a distributed operating system*

When using a distributed environment, keep in mind the following considerations:

- Multiple administrative agent profiles can exist in a single workstation.
- You cannot register one WebSphere Application Server profile to multiple administrative agents.
- Registering a WebSphere Application Server profile to a remote administrative agent is not possible and is not supported.
- The WebSphere Application Server profile and the corresponding administrative agent must be in the same workstation.

## Administering WebSphere Application Server nodes from the administrative agent

An administrator can connect to registered WebSphere Application Server profiles from the administrative agent's console for managing the servers. The administration services in the administrative agent modify the configuration of the various registered WebSphere Application Server profiles directly, which means that the administrative agent can manage only WebSphere Application Server profiles that are running on the same computer. The node for each WebSphere Application Server profile is managed separately by the same administrative agent but using different connector ports for each node and different administrative runtime instances for each node.

Management functions performed by the administrative agent remain isolated between registered WebSphere Application Server profiles, enabling the user to direct operations to specific WebSphere Application Server profiles within the set of profiles that is being managed.

Figure 8 illustrates the configuration of workstation A when using the administrative agent to administer profiles.

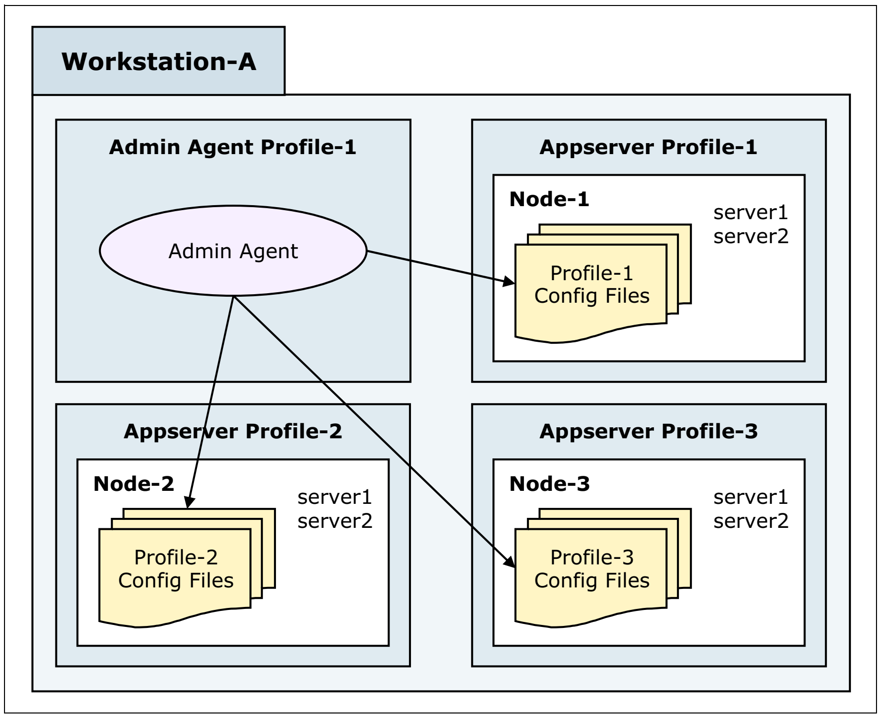


*Figure 8 Administering base profiles*

## Security state considerations in an administrative agent topology

You can register WebSphere Application Server profiles to the administrative agent with almost any security configuration. When registering profiles, keep in mind the following security considerations:

- The WebSphere Application Server profile that you register must have the same admin security state as the administrative agent, which is either *enabled* or *disabled*.
- Using a mixed state, for example, where the administrative agent is security enabled and WebSphere Application Server is security disabled or vice versa, is not possible. If you attempt to use a mixed state, an exception error will occur during registration.
- After a node is registered, you cannot enable or disable the administrative security for that node (or any other registered node) until the node has been deregistered.

Figure 9 illustrates the configuration of workstation A using multiple user registries with an administrative agent topology.

**LDAP note:** Figure 9 uses a complex case where each profile is configured with a different LDAP. Note, however, that all the profiles can share the same LDAP.

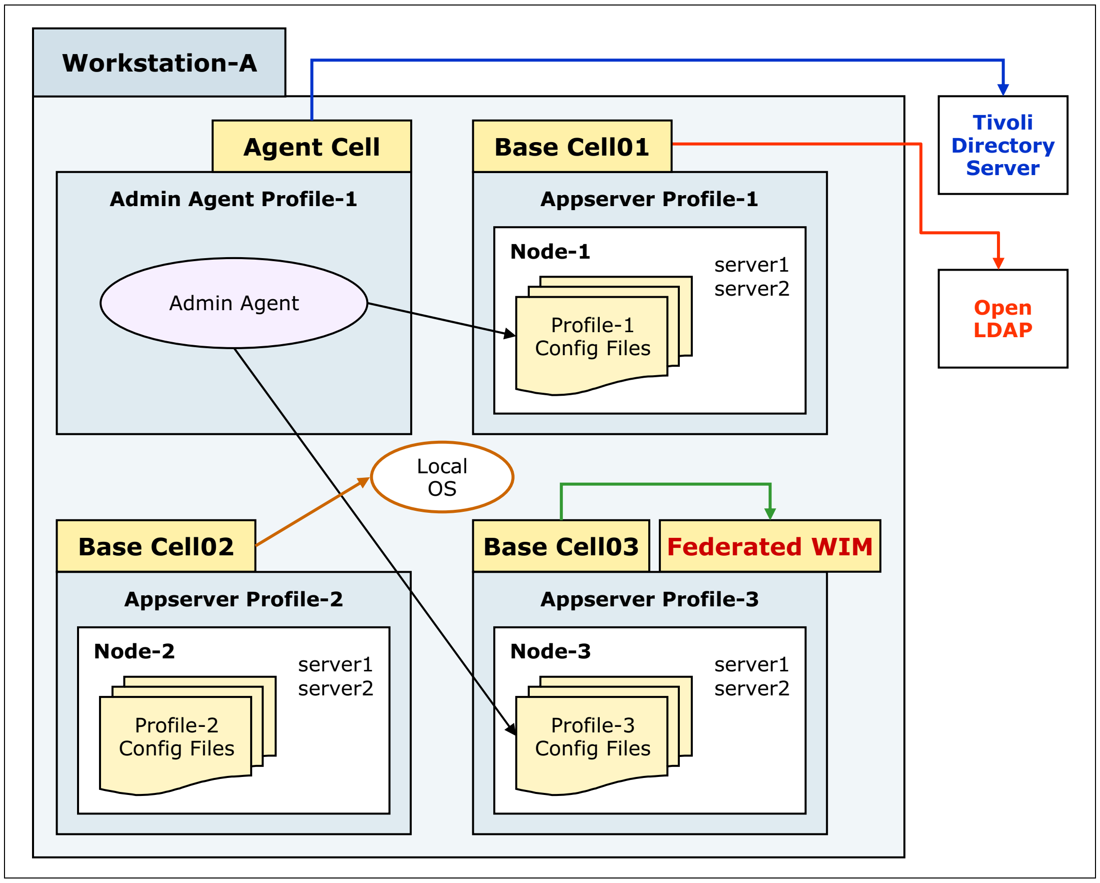


*Figure 9 Multiple user registries within the administrative agent topology*

For information about security considerations when registering a WebSphere Application Server node with the administrative agent, see this website:

http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/index.jsp?topic=/com.ibm.websphere.nd.doc/info/ae/ae/rsec_7register_admin_agent.html

## Registering a node with security enabled state

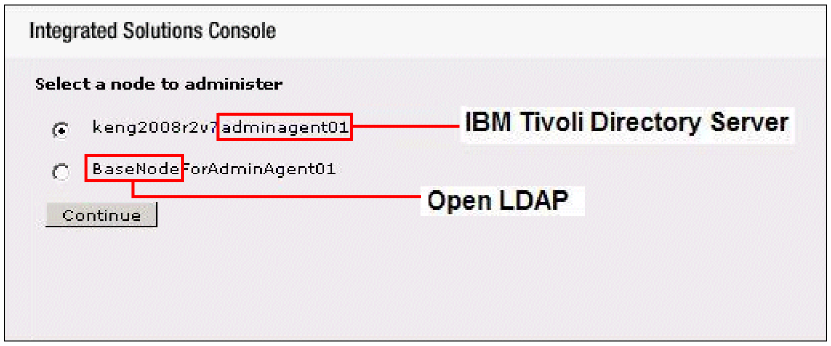
To register a WebSphere Application Server profile to an administrative agent, use the **registerNode** command. This command is available in the profile for the administrative agent, and you can execute it from the administrative agent's `bin` directory irrespective of the security state, as follows:

```
C:\IBM\WebSpherev7\AppServer\profiles\adminagent01\bin>registerNode -conntype
SOAP -port  <SOAP Port of Admin Agent>  -profilePath
C:\IBM\WebSpherev7\AppServer\profiles\AppServer01 -username adminagentuser
-password  agentsecret -nodeusername  appserveradmin -nodepassword
appserversecret
```

## Using the administrative console after registration

After you register the WebSphere Application Server node with the administrative agent, the console application (the administrative console) is disabled and all the administrative tasks are performed from the administrative agent's administrative console. Users always log in through the administrative agent's administrative console, where the profile's context is set even before the login. Console users between the administrative agent and the registered WebSphere Application Server profiles do not need to be exchanged.

Select options to set the login context as shown in Figure 10.


Integrated Solutions Console

Select a node to administer

(●) keng2008r2v7adminagent01

( ) BaseNodeForAdminAgent01

*Figure 10 Selecting the login context*

Based on the selected context, the corresponding `security.xml` file and related security configuration is loaded. The user needs to enter the user credential that is specific to the selected profile. When logging in to the administrative agent's console itself, use the administrative agent's user name and password. For any registered application server profiles, use the corresponding login credentials for the WebSphere Application Server profiles, which are the same credentials as the login to the stand-alone WebSphere Application Server profile console.

Although you can register multiple WebSphere Application Server profiles with the same administrative agent, user credentials or LTPA tokens are not shared between cells in the administrative agent topology.

The profile context is set before the login, and users are directed to the corresponding registry for authentication and authorization purposes. After logging in to the profile, the user will have the same experience as when logging in to the application server administrative console.

**RSA note:** The user login or application security processes do not use the RSA administrative authentication mechanism. The RSA administrative authentication mechanism is purely for server-to-server administrative trust and authentication.

## Advantages of using an administrative agent topology

The administrative agent process is a WebSphere Application Server product offering and does not require the WebSphere Application Server Network Deployment solution. Even after registering the WebSphere Application Server profile to the administrative agent, the process preserves the existing management experience. WebSphere Application Server profiles retain their configuration, and there is no master configuration repository in the administrative agent or node synchronization.

When using an administrative topology, consider the following advantages:

- Reducing memory footprint and resources

  You can register multiple stand-alone application servers with a single administrative agent. The administrative services of the registered WebSphere Application Server profiles are consolidated into the administrative agent, thus reducing the memory footprint from the previously duplicated services.

- Registering with job manager

  You can register the WebSphere Application Server profile to the job manager only using its administrative agent process. You cannot register stand-alone WebSphere Application Server profiles directly to the job manager.

- Remote administration of WebSphere Application Server

  After registering the WebSphere Application Server profiles to the administrative agent, you can start and stop the servers (for example, *server1*) in the profile remotely using either a browser or a remote wsadmin connected to the administrative agent's process.

- Administering multiple application servers using the console

  With the current WebSphere Application Server architecture, you can create multiple application servers (for example, server1, server2, and so forth) in the same profile, but you cannot administer these servers using the WebSphere Application Server administration console. In this configuration, console administration is limited only to server1. Other servers in that profile, such as server2 and server3, can be managed only using wsadmin.

  By registering the WebSphere Application Server profile with the administrative agent, you can manage all the application servers in that profile using the administrative agent's administrative console.

## Removing a WebSphere Application Server profile from the administrative agent

At any time, you can deregister a registered WebSphere Application Server profile from the administrative agent without data loss. Configuration changes made from the administrative agent process are directly updated in the application server profile's configuration files, and there is no data loss.

You can use the following commands:

- **deregisterNode** (security state disabled)

  ```
  C:\IBM\WebSpherev7\AppServer\profiles\adminagent01\bin>deregisterNode -conntype
  SOAP -port  <SOAP Port of Admin Agent>  -profilePath
  C:\IBM\WebSpherev7\AppServer\profiles\AppServer01
  ```

- **deregisterNode** (security state enabled)

  ```
  C:\IBM\WebSpherev7\AppServer\profiles\adminagent01\bin>deregisterNode -conntype
  SOAP -port  <SOAP Port of Admin Agent>  -profilePath
  C:\IBM\WebSpherev7\AppServer\profiles\AppServer01 -username adminagentuser
  -password  agentsecret -nodeusername  appserveradmin -nodepassword
  appserversecret
  ```

All the features of the administrative agent (for example, remote administration of WebSphere Application Server) are not available until you register the server with the administrative agent again.

## Job manager

In a flexible management environment, a *job manager* allows you to submit administrative jobs asynchronously for application servers that are registered to administrative agents and for deployment managers. The job manager is designed to complement both the Network Deployment and WebSphere Application Server and Network Deployment topologies.

You create a job manager profile using the **manageprofiles** command or using the Profile Management Tool. Similar to the deployment manager profile, the job manager profile is available only with WebSphere Application Server Network Deployment. The job manager does not take over the node's configuration. Registered nodes (such as the deployment manager and WebSphere Application Server) retain their autonomy and isolation. Even after registering with job manager, you can continue to manage these nodes separately using the administrative agent and the deployment manager as before.

To enable a job manager topology, you do not need to reconfigure any existing WebSphere Application Server profile node (node agent), administrative agent, or deployment manager nodes. The topologies that are managed by a job manager maintain their autonomy, including their security configuration, and thus you can manage these topologies directly using existing administrative processes, such as scripts or the administrative console.

The primary purpose of the job manager is to submit jobs to the application server nodes. Each application server node or the deployment manager that is registered with the job manager is known as a *managed node* to the job manager. After you register application server nodes and deployment managers with the job manager, you can queue administrative jobs that are directed at the application server nodes or deployment managers through the job manager. When you queue administrative jobs, keep in mind the following considerations:

- Jobs are queued in the job manager and are not pushed to the respective nodes.
- Job manager does not broadcast or notify the jobs to the nodes.
- The administrative agent and deployment manager processes check (poll) the job manager to determine whether the job manager has queued jobs that require action.

This polling model enables a single job manager to manage multiple application server nodes. Otherwise, broadcasting jobs to hundreds of nodes would overkill the job manager. Because the managed node is not owned by the deployment manager, a Network Deployment topology node can be managed by multiple job managers. The asynchronous job submission model facilitates the management of nodes that are geographically dispersed and reachable only through low-bandwidth, high-latency networks.

For more information about the job manager, see this website:

http://www14.software.ibm.com/webapp/wsbroker/redirect?version=compass&product=was-nd-dist&topic=cagt_jobmanager

## Registering the deployment manager with the job manager

You can register deployment manager profiles to the job manager with almost any security configuration. You can register the job manager using the wsadmin **AdminTask** command or using the job manager's administrative console.

When registering the deployment manager, keep in mind the following security state considerations:

- The deployment manager profile that you register must have the same admin security state as the job manager (which is either *enabled* or *disabled*).
- You cannot use a mixed state (for example, where the job manager is security enabled and the deployment manager is security disabled or vice versa). If you use this configuration, errors will occur during registration.
- When registered, you cannot enable or disable the administrative security for the deployment manager node.
- The job manager and the registered nodes can have different user registries.

**RSA preferred practice:** The deployment manager profiles are not RSA enabled by default. As a preferred practice, before you add the deployment manager profile to the job managers, enable RSA as an administrative authentication mechanism. Then save and synchronize the changes, and restart the deployment manager.

To register the deployment manager, execute **AdminTask.registerWithJobManager** from the wsadmin client by connecting to the job manager's port as follows:

- Register the deployment manager to the job manager with security state disabled:

```
<dmgr_profile_root>/bin>wsadmin –lang jython

wsadmin>AdminTask.registerWithJobManager ('[-host jobmanagerhost -port  <
unsecure port HTTP Transport> -managedNodeName  ActualDMGRNodeName –alias
OptionalAliasNameForDMGRNode]')
```

- Register the deployment manager to the job manager with security state enabled:

```
<dmgr_profile_root>/bin>wsadmin –lang jython -user dmgruser -password
dmgrsecret

wsadmin>AdminTask.registerWithJobManager('[-host jobmanagerhost -port  < secure
port HTTPS Transport> -user jobmanageruser -password jobmanagersecret
-managedNodeName  <ActualDMGRNodeName> –alias <Optional
AliasNameForDMGRNode>]')
```

You do not need to register the application server nodes (application server profiles) separately that are already federated to the deployment manager cell. After you register the deployment manager with the job manager, the job manager can submit jobs to all the application servers in that deployment manager cell. No separate registration is needed for application server nodes in the same deployment manager cell.

Use **`AdminTask.unregisterWithJobManager`** to deregister the deployment manager from the job manager.

## Registering a WebSphere Application Server profile with the job manager in an administrative agent topology

You do not register the administrative agent with the job manager. You register only the WebSphere Application Server profiles to the job manager using the administrative agent. The administrative agent is a *pure agent process* that facilitates the job manager registration process for WebSphere Application Server nodes that are registered within.

When registering a WebSphere Application Server project with the job manager in an administrative agent topology, keep in mind the following security considerations:

- The WebSphere Application Server profile that you register and its administrative agent profile must have the same admin security state as the job manager (which is either *enabled* or *disabled*).
- You cannot use mixed security states (for example, where the job manager is security enabled and the deployment manager is security disabled or vice versa).
- Before registering to job manager, you must enable or disable administrative security for the WebSphere Application Server profile and its administrative agent profile. When registered, you cannot enable or disable the administrative security for that deployment manager node.
- The job manager and the WebSphere Application Server profile and its administrative agent profile nodes can have different user registries.

To register a WebSphere Application Server profile, execute **AdminTask.registerWithJobManager** from the wsadmin client by connecting to the administrative agent port, as follows:

- Register the application server to the job manager with security state disabled:

```
<admin_agent_profile_root>/bin>wsadmin –lang jython –port <SOAP Port of Admin
Agent process>

wsadmin>AdminTask.registerWithJobManager('[-host jobmanagerhost -port  <
unsecure port HTTP Transport> -managedNodeName  <Registering AppServer
ProfileNodeName> –alias <Optional MyAppServProfileNodeName]')
```

- Register the application server to the job manager with security state enabled:

```
<admin_agent_profile_root>/bin> wsadmin –lang jython –port <SOAP Port of Admin
Agent process> -user adminagentuser -password adminagentsecret

wsadmin>AdminTask.registerWithJobManager('[-host jobmanagerhost -port  < secure
port HTTPS Transport> -user jobmanageruser -password jobmanagersecret
-managedNodeName  <Registering AppServer ProfileNodeName> –alias
<OptionalMyAppServProfileNodeName>]')
```

You do not need to register all the WebSphere Application Server nodes from the administrative agent to the job manager. If multiple stand-alone application profiles are registered with a single administrative agent, only selected application server profiles are registered with the job manager.

Use **AdminTask.unregisterWithJobManager** to deregister WebSphere Application Server from the job manager.

## Flexible management topology

Figure 11 shows the flexible management topology with the job manager, deployment manager, administrative agent, and application server.

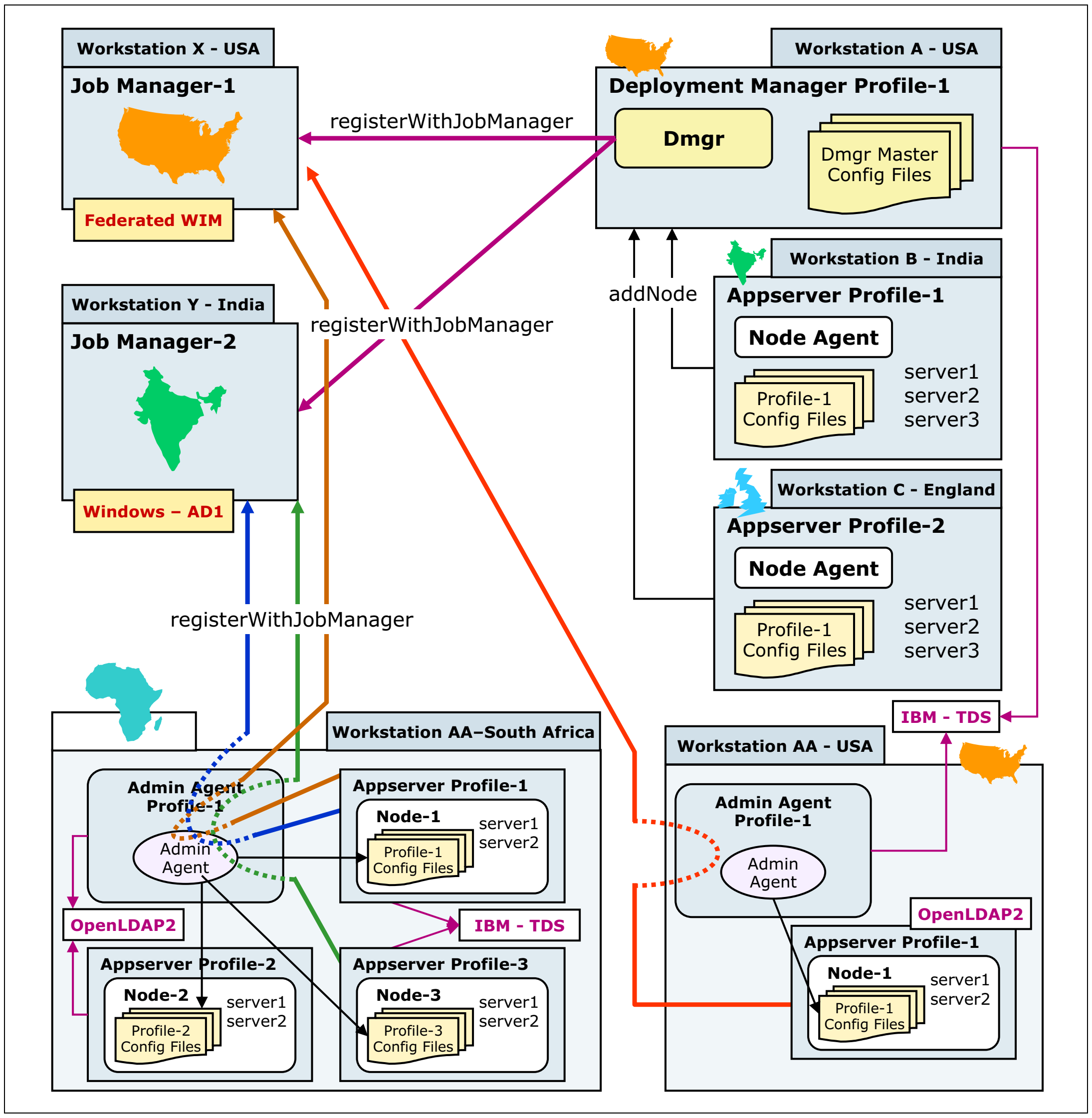


*Figure 11 Flexible management topology*

In this example, three WebSphere Application Server profiles (Profile-1, Profile-2, and Profile-3) are registered with an administrative agent on workstation AA-South Africa. Another application server Profile-1 on workstation AA- USA is registered with an administrative agent on workstation AA- USA.

In this configuration, the following job managers are available:

- The job manager at workstation X—USA
- The job manager at workstation Y—INDIA

The deployment manager and application servers (through the administrative agent) are registered to one or more job managers. The application server and deployment manager periodically poll the job manager to determine whether the job manager has queued jobs that require action.

## Points to remember with flexible management topology

Keep in mind the following points with a flexible management topology:

- All the profiles (the administrative agent, the stand-alone WebSphere Application Server that is registered with the administrative agent, and the deployment manager) that participate in the flexible management topology must have the same admin security state (which is either *enabled* or *disabled*).
- There is no user registry restriction. Participating profiles (the administrative agent, the stand-alone WebSphere Application Server that is registered with the administrative agent, and the deployment manager) can use almost any security configuration independently.
- The RSA authentication mechanism is used only for server-to-server administrative authentication, such as administrative connector and file transfer requests. The RSA authentication mechanism does not replace LTPA or Kerberos for use by applications.

### Deployment manager topology

With a deployment manager topology, keep in mind the following points:

- The deployment manager is a single point of contact for all the application server nodes that are managed by that deployment manager. Only the deployment manager node is registered with the job manager.
- The deployment manager node polls the job manager on behalf of all the federated nodes that are within it. Execute **`AdminTask.registerWithJobManager`** from the wsadmin client by connecting to the registering deployment manager server port. Provide the actual deployment manger node name for **`-managedNodeName`**.
- One deployment manager can be registered with multiple job managers.

### Administrative agent topology

With an administrative agent topology, keep in mind the following points:

- There is no requirement of registering an administrative agent to the job manager. In an administrative agent topology, jobs are submitted only to registered WebSphere Application Server profiles.
- An administrative agent is required for a stand-alone WebSphere Application Server profile to register with a job manager. The administrative agent is a pure agent process that facilitates the registration process for the application server nodes that are registered within.
- Execute **`AdminTask.registerWithJobManager`** from the wsadmin client by connecting to the administrative agent port.
- Provide the actual WebSphere Application Server profile node for **`-managedNodeName`**.
- One application server profile node can be registered with multiple job managers, but it cannot be registered with more than one administrative agent.

## Defining and using jobs in a flexible management topology

In a flexible management topology, you can define a *job* as an executable task at the target nodes. You can predefine a job as a single administrative task, or you can use a custom wsadmin script that contains multiple tasks. You can submit jobs to any registered nodes, but for successful execution at target nodes, the respective administrative agent or deployment manager and the node agent must be running.

**Jobs with target nodes:** If a target node is a stand-alone application server and if the target node is registered with the administrative agent, jobs are executed at target nodes only if the administrative agent is running. In addition, if a target node is a federated application server with the deployment manager, jobs are executed at target nodes only if the deployment manager and the corresponding node agent are running.

The types of jobs that you can submit from the job manager vary based on the nodes that you have registered with the job manager. You can have job types that manage applications, modify the product configuration on remote computers, or do general-purpose tasks, such as run a script. The following job types exist:

- Collect file
- Configure properties
- Create application server
- Create cluster
- Create cluster member
- Create proxy server
- Delete application server
- Delete cluster
- Delete cluster member
- Delete proxy server
- Distribute file
- Install application
- Inventory
- Remove file
- Run wsadmin script
- Start application
- Start cluster
- Start server
- Status
- Stop application
- Stop cluster
- Stop server
- Uninstall application
- Update application

For the latest job types and more information about job types, refer to the following topics in the WebSphere Application Server Information Center:

- Submitting jobs using the job manager console

  `http://www14.software.ibm.com/webapp/wsbroker/redirect?version=compass&product=was-nd-dist&topic=tagt_jobsub`

- Submitting jobs using the wsadmin **`submitJob`** command

  `http://www14.software.ibm.com/webapp/wsbroker/redirect?version=compass&product=was-nd-mp&topic=rxml_7jobtypes`

- Submitting jobs to run wsadmin scripts on managed nodes

  `http://www14.software.ibm.com/webapp/wsbroker/redirect?version=compass&product=was-nd-dist&topic=tagt_jobmgr_run_wsadmin`

## Security considerations when submitting and executing jobs

In a flexible management environment, you can submit jobs to remote nodes. In this case, the job submitter ID must be authorized for the administrator role or the operator role to submit jobs. When you submit a job, you can specify a user name and password for authentication and authorization at the target node or nodes. If you do not provide a user name and password in the job parameters, the credentials for the job submitter at the job manager are used for this purpose. When you submit a job to multiple target nodes, the user name and password or the credentials for the submitter must apply to all the job targets.

Successful job submission (either from the job manager administrative console or using a wsadmin client that is connected to the job manager process) from the job manager to the actual job execution at the target nodes requires careful consideration and configurations of security aspects.

## Identity propagation

*Identity propagation* provides a mechanism to allow a user identity from a security realm to be preserved, regardless of where the identity information was created. The user identity originates from a remote system and is passed to one or more systems over a network for authorization and auditing purposes.

In a job manager topology, where all the participating nodes are using the same user registry and the job submitter is an authenticable user in all the target nodes, identity propagation is used to execute the jobs in the downstream target nodes. Otherwise, the user credentials need to be sent along with the jobs.

## User role assignment

The job submission source is always the job manager. The job submitter's user ID at the job manager must be authorized for the administrator role or the operator role to submit jobs from the job manager.

**Operator role required:** At a minimum, the operator role is required for job submission from the job manager.

Target nodes vary based on the topology. In a federated deployment manager topology, the job executer's user ID must be authorized for the administrator or the operator role (or the required role based on the job type) in the deployment manager.

In an administrative agent topology, where a WebSphere Application Server profile is registered with the administrative agent, the job executer's user ID must be authorized for the administrator or the operator role (or the required role based on the job type) in the managed WebSphere Application Server node. There is no requirement to add the job executer's ID in the administrative agent's authorization table (`admin-authz.xml`) itself.

**Target node note:** The target node's administrative role requirement depends on the job type. Before submitting the job from the job manager, use the same user to perform the same job at the target node by directly logging in to the target node to confirm the role requirement.

When considering the user registry, user identify propagation, and user authorization, the process of submitting and executing jobs can be divided into the following major categories:

- All the participating nodes use the same user registry.
- Participating nodes use different user registries.

The sections that follow discuss these categories.

## All the participating nodes use the same user registry

Even if all the participating profiles (the job manager, administrative agent, application server registered with the administrative agent, and deployment manager) use the same user registry, you need to configure the downstream nodes for successful job execution.

The sample topology shown in Figure 12 illustrates a federated node (workstation B-INDIA) that is managed by a deployment manager (workstation A—USA). The WebSphere Application Server profile-2 at workstation AA—USA is registered with an administrative agent. The job manager is available at workstation X—USA.

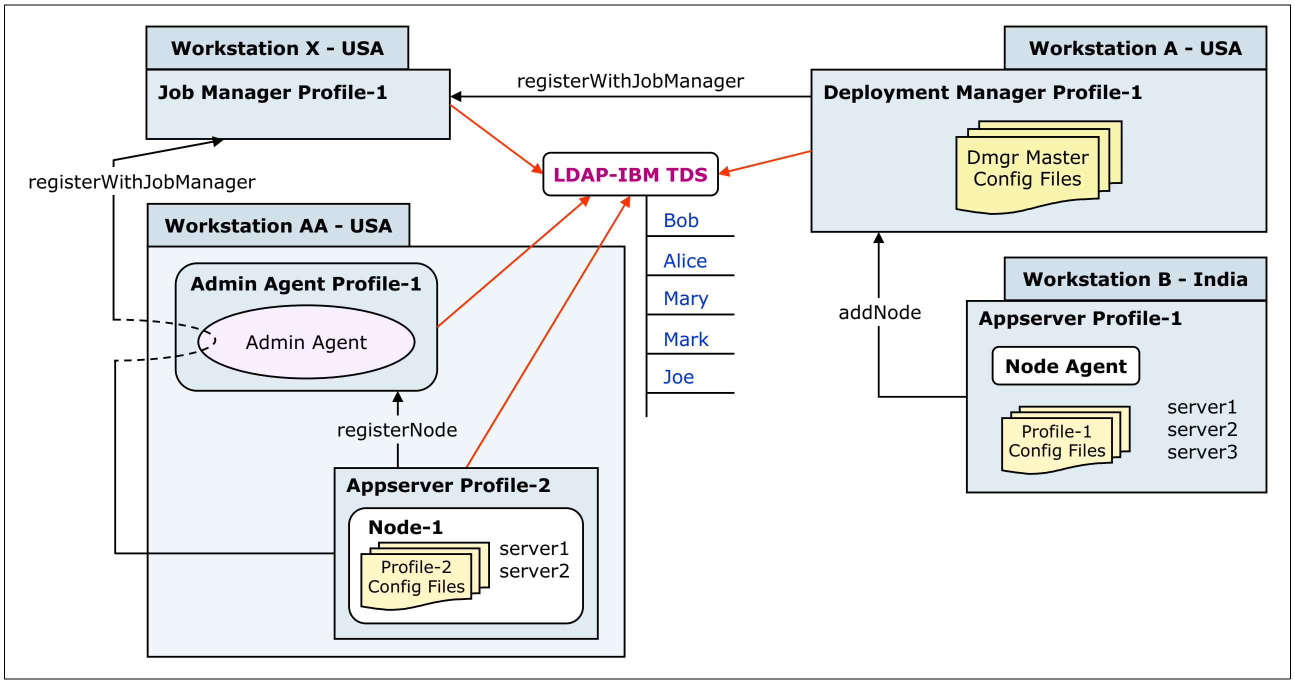


*Figure 12 All the profiles use the same user registry*

All the profiles in this topology use the same LDAP user registry, IBM Tivoli® Directory Server. Users Bob, Alice, Mary, Mark, and Joe are available in the LDAP user registry. Although all the profiles share the same registry (administrative agent, stand-alone WebSphere Application Server, which is registered with the administrative agent and deployment manager) each profile can define its authenticable users individually in their respective administrative authorization tables (`admin-authz.xml`).

When considering the authenticable user for the job execution at the target nodes, the following options are available for this topology:

- Supplying a user name and password when the job is submitted
- Adding the user name for job submitter to the target node's authorization table for identity propagation

The following sections provide examples of these options.

## Supplying a user name and password when the job is submitted

Figure 13 shows that the job submitter *Alice* is not in the target nodes' authorization tables.

**Authorization tables:** Though all the profiles use the same user registry, users needs to be added in each profile's authorization tables explicitly. The profiles are separate and independent cells.

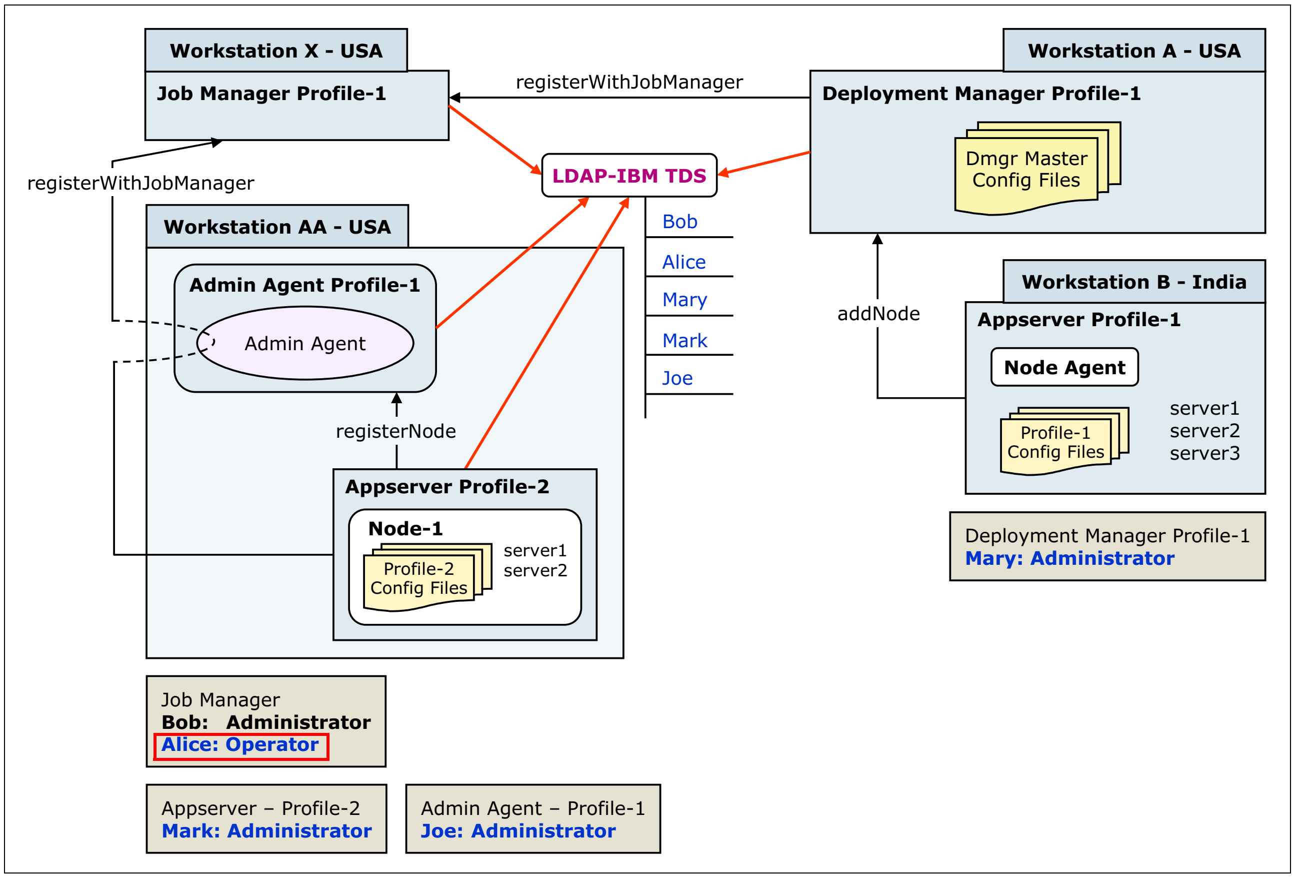
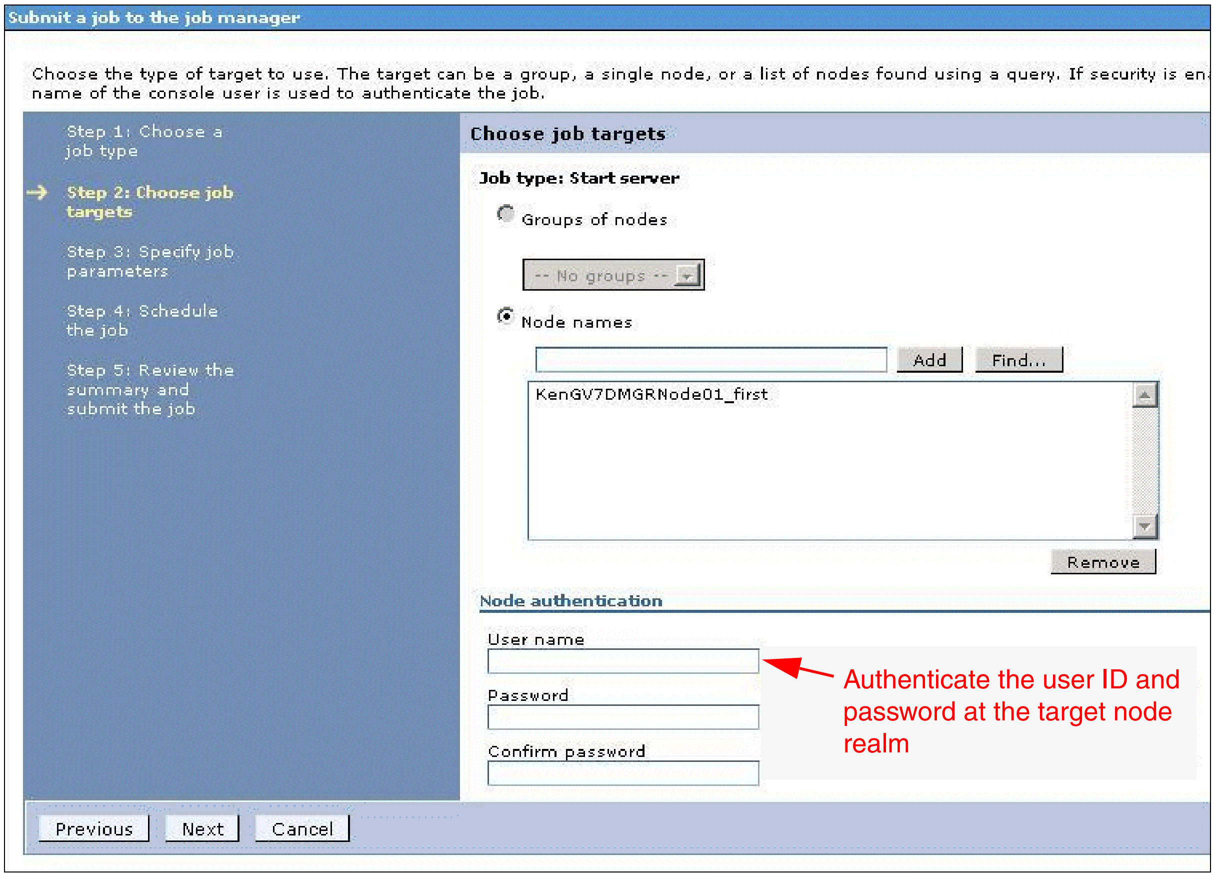


*Figure 13 Alice defined only at the job manger's admin-authz.xml as an operator role*

After a logging in to the job manager, Alice can submit the job. However, for the job to be executed successfully at the target nodes, Alice needs to enter the authenticable user name and password in the Node authentication pane (Figure 14) during the job submission itself.



Submit a job to the job manager

Choose the type of target to use. The target can be a group, a single node, or a list of nodes found using a query. If security is en
name of the console user is used to authenticate the job.

Choose job targets

Job type: Start server

Groups of nodes [✓ Node names]

KenGV7DMGRNode01_first

Node authentication

User name

Password

Confirm password

Step 1: Choose a job type
Step 2: Choose job targets
Step 3: Specify job parameters
Step 4: Schedule the job
Step 5: Review the summary and submit the job
-- No groups --
Add
Find...
Remove
Previous
Next
Cancel
Authenticate the user ID and password at the target node realm

*Figure 14 Node authentication pane*

Alice needs to know the authenticable user name and password from the target nodes:

- To submit the jobs to the deployment manager profile-1, Alice must know Mary's user name and password and must enter that information in the Node authentication pane.
- To submit the jobs to the WebSphere Application Server Profile-2, Alice must know Mark's user name and password and must enter that information in the Node authentication pane.

**User name and password note:** The user name and password are passed from the job manager to the target nodes and are used to log in. This user name and password must be valid at the target node's realm. In addition, when you submit a job to multiple target nodes, the user name and password credentials that are submitted with the job must be applicable to all of the target nodes.

**Challenge of sharing user credentials:** This approach presents a challenge in a real-world environment where people are reluctant to share user credentials. Mary or Mark can say no to Alice's request for credentials sharing.

## Adding the user name for job submitter to the target node's authorization table for identity propagation

For Alice to submit and execute the jobs without using others' credentials, she needs to be an authenticable user with the required administrative roles defined in the source and target nodes. Alice's user ID can be added easily to the needed target nodes using the respective nodes' administrative console.

Figure 15 illustrates adding the user ID for Alice to the target node's `admin-authz.xml` file.

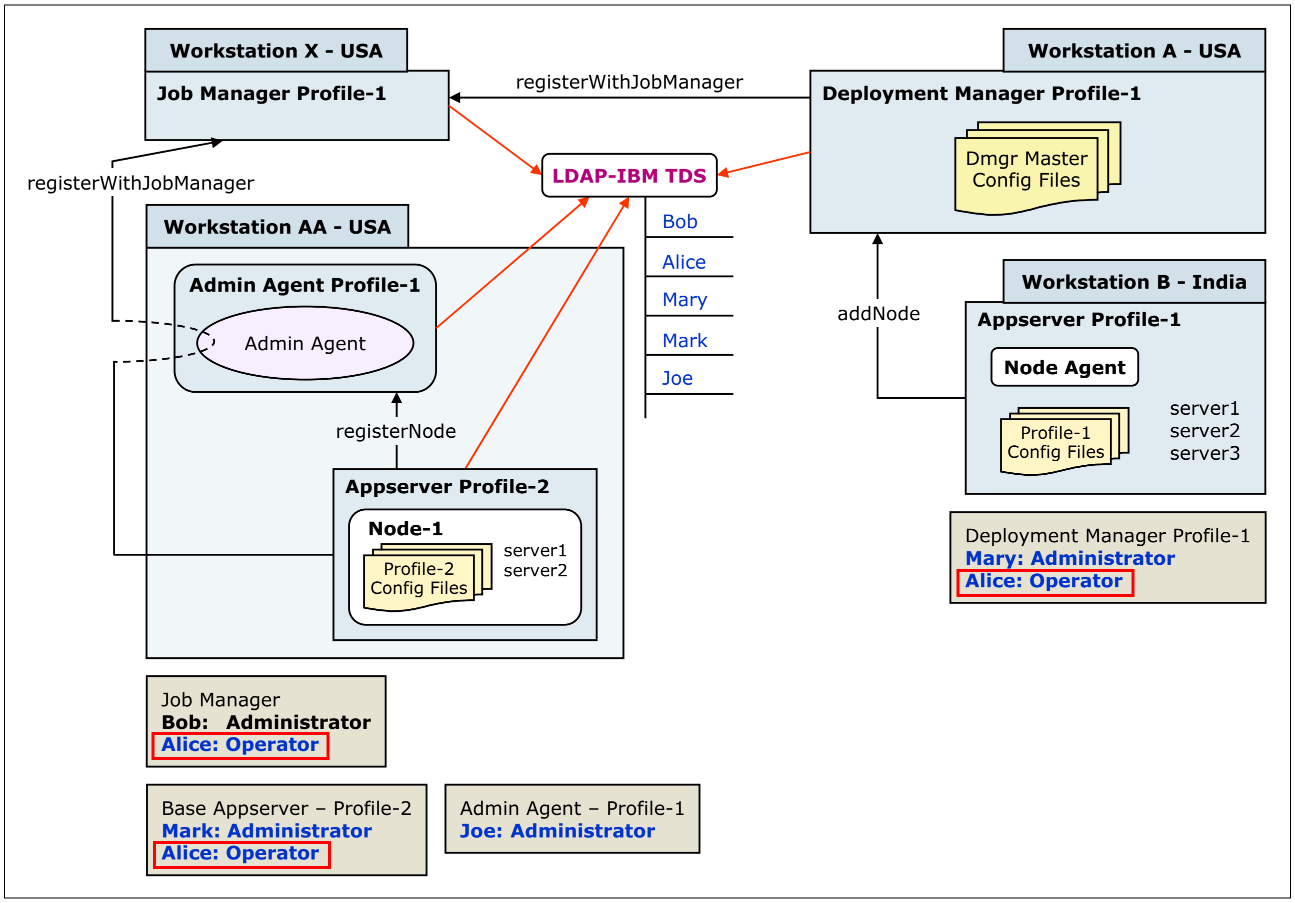
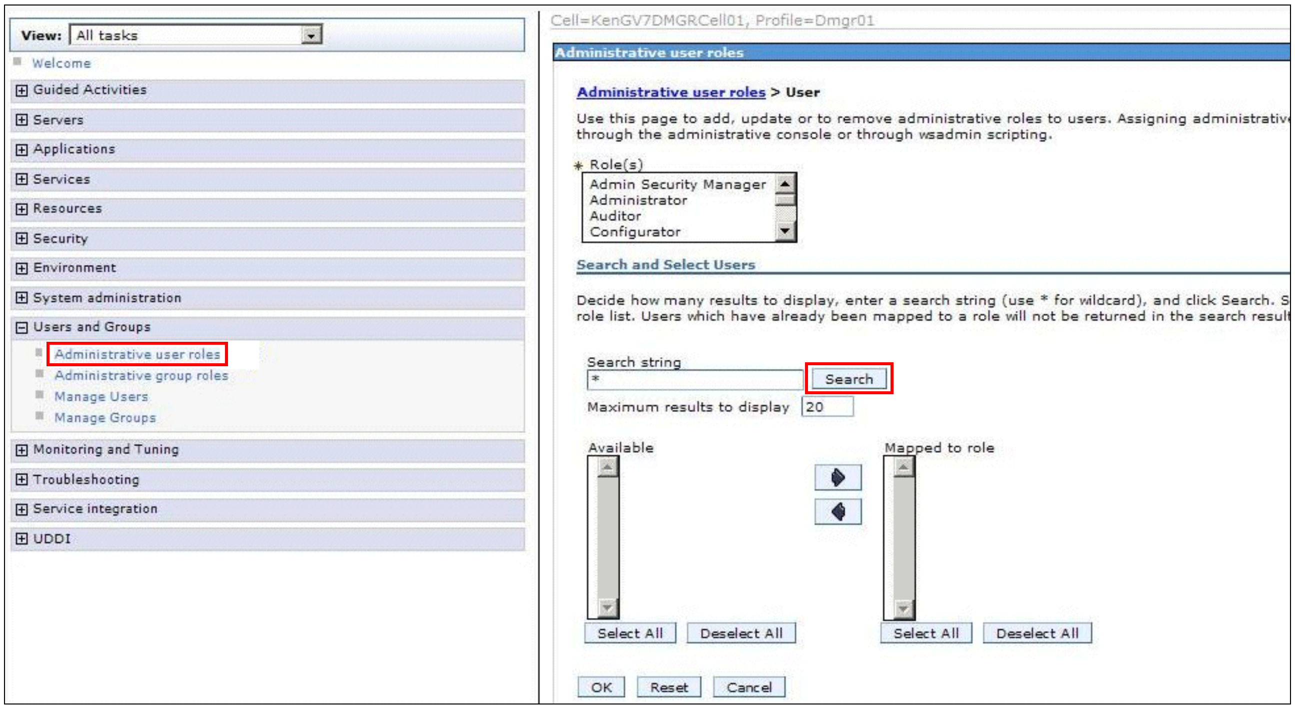


*Figure 15 Alice's user ID added to the target nodes in the admin-authz.xml file*

If Alice's name is not defined as an authenticable user with the required roles, you can add her name using the administrative user roles mapping. To do this, select **Console** → **Users and Groups** → **Administrative user roles** (Figure 16).



Cell=KenGV7DMGRCell01, Profile=Dmgr01

Administrative user roles

Administrative user roles > User

Use this page to add, update or to remove administrative roles to users. Assigning administrativ
through the administrative console or through wsadmin scripting.

* Role(s)
Admin Security Manager
Administrator
Auditor
Configurator

Search and Select Users

Decide how many results to display, enter a search string (use * for wildcard), and click Search. S
role list. Users which have already been mapped to a role will not be returned in the search resul

Search string: * [Search]

Maximum results to display: 20

*Figure 16   Assigning administrative roles*

After logging in to the job manager, Alice can then submit the job. Because the jobs are submitted by Alice, her user credentials are passed with the job to the downstream target nodes, and she does not need to re-enter her credentials in the Node authentication pane. Leave that pane empty (Figure 14 on page 27).

**Identity token:** Alice's identity is propagated automatically from the job manager to target nodes along with the job. The identity token is passed from the job manager to the node and the deployment manager.

The job manager queues the jobs in the job manager queue with all the required credential information. When the target nodes query for the jobs, this information is passed with the jobs.

**Submitting a job to multiple target nodes:** When you submit a job to multiple target nodes, the user name and password or the credentials that are submitted with the job must be applicable to all of the target nodes.

**Node authentication:** Specifying the user name and password in the Node authentication pane overwrites the identity token that is passed from the job manager and acts as an actual user in the target node realm. If that user is not in that target node realm, the job fails.

## Executing jobs at target nodes

All the submitted jobs contain enough information, including user information, to execute the job at the target nodes. These user credentials are validated at target nodes before the job is executed. If the user does not have authorization to execute the job at the target nodes, the job fails.

**Administrative role:** If a valid administrative role is defined in the source (job manager) and the target nodes (a federated deployment manager and WebSphere Application Server node in an administrative agent topology), Alice can submit and execute the job in the job manager topology successfully.

## Participating nodes use different user registries

In this example, the participating profiles (that is, job manager, administrative agent, and profiles registered with administrative agent and deployment manager) use different user registries (Figure 17).

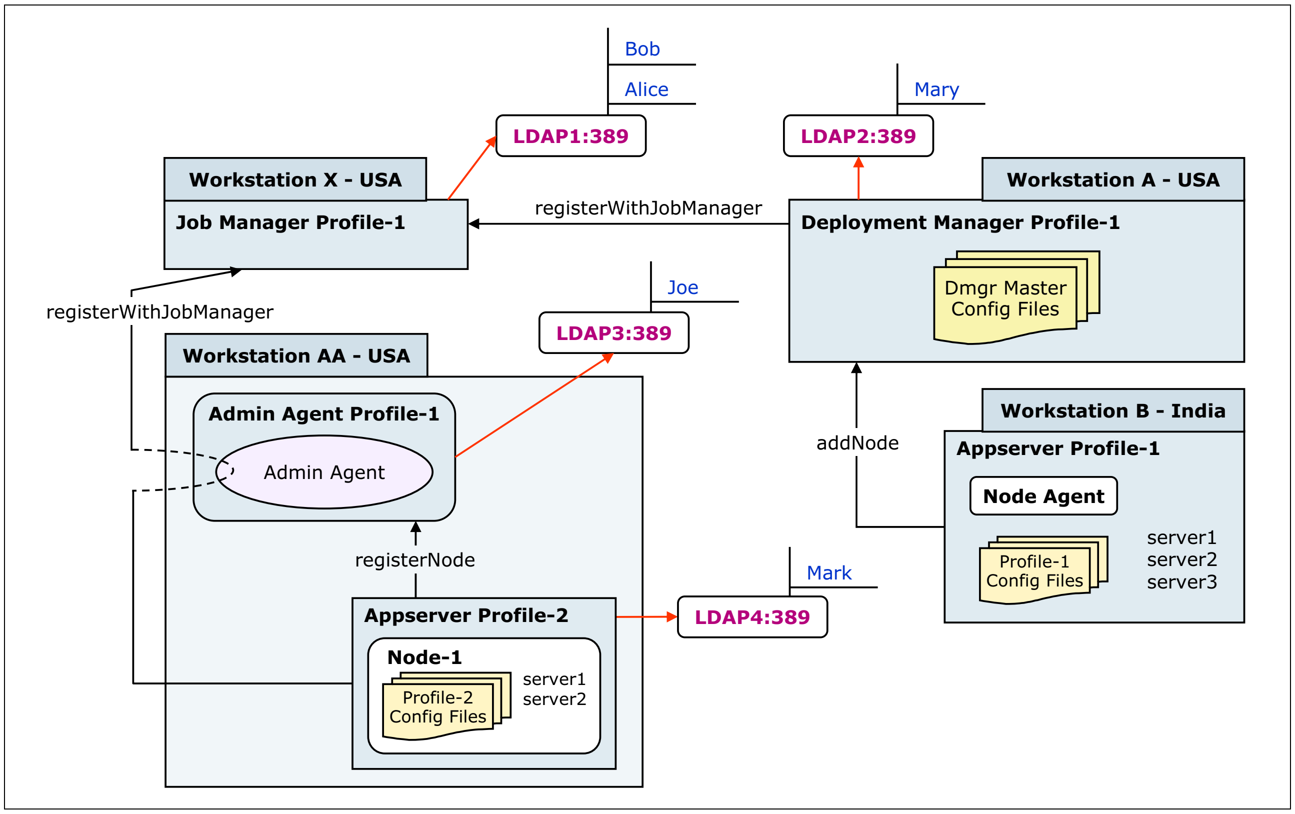


*Figure 17 Profiles configured with different user registries*

This topology shows a federated node (workstation B—INDIA) that is managed by a deployment manager (workstation A—USA), an application server named profile-2 that is registered with an administrative agent (Workstation AA—USA), and a job manager (workstation X—USA).

When all the nodes are using different LDAP user registries, additional configuration is required (Figure 18).

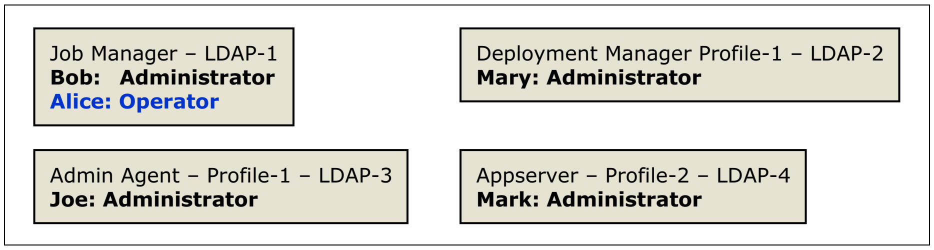


*Figure 18 LDAP user registries configuration*

## Establishing trust between the job manager and the target node realms

In the RSA authentication mechanism, the RSA token is passed between the nodes that are involved in the job manager topology. When the RSA token is received by the downstream server, it attempts to decrypt the token. If the realms are not trusted between nodes, the token validation fails. If you want the cross-realm communication to succeed with RSA tokens, you must first establish trust between the realms involved, both for inbound and outbound communications.

The server that is originating the request must have the trusted realms configured before sending the token to that target realm, which is referred to as *outboundTrustedRealms*. The server receiving the request must be configured to trust the realms from which it receives LTPA tokens, which is referred to as *inboundTrustedRealms*.

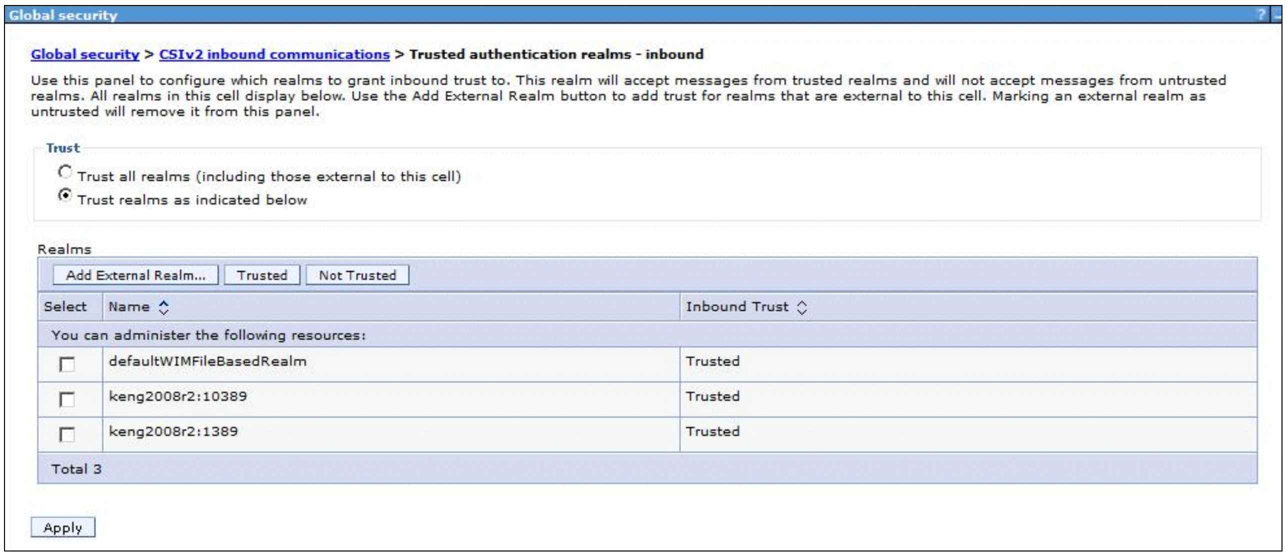
In a single user registry topology, there is no question of trusting realms between the participating nodes. However, when the user registries are different in the job manager topology, cross-realm trust is important for successful RSA token validation.

This section provides an overview of how to establish cross-realm trust between the job manager and the target node.

### Configuring the inbound trust realm for the target nodes

In a cross-realm environment, the target node must trust the incoming RSA token from the job manager. Thus, you must add the job manager's realm in the target node's inbound trust. Target nodes are either the deployment manager or application servers that are registered with the administrative agent, which can be done from the target node's administrative console.

Go to **Console** → **Global security** → **CSIv2 inbound communications** → **Trusted authentication realms – inbound** (Figure 19).



Global security > CSIv2 inbound communications > **Trusted authentication realms - inbound**

Use this panel to configure which realms to grant inbound trust to. This realm will accept messages from trusted realms and will not accept messages from untrusted realms. All realms in this cell display below. Use the Add External Realm button to add trust for realms that are external to this cell. Marking an external realm as untrusted will remove it from this panel.

Trust

Trust all realms (including those external to this cell) [✓ Trust realms as indicated below]

Realms

| Select | Name | Inbound Trust |
|---|---|---|
| You can administer the following resources: | | |
| ☐ | defaultWIMFileBasedRealm | Trusted |
| ☐ | keng2008r2:10389 | Trusted |
| ☐ | keng2008r2:1389 | Trusted |
| Total 3 | | |

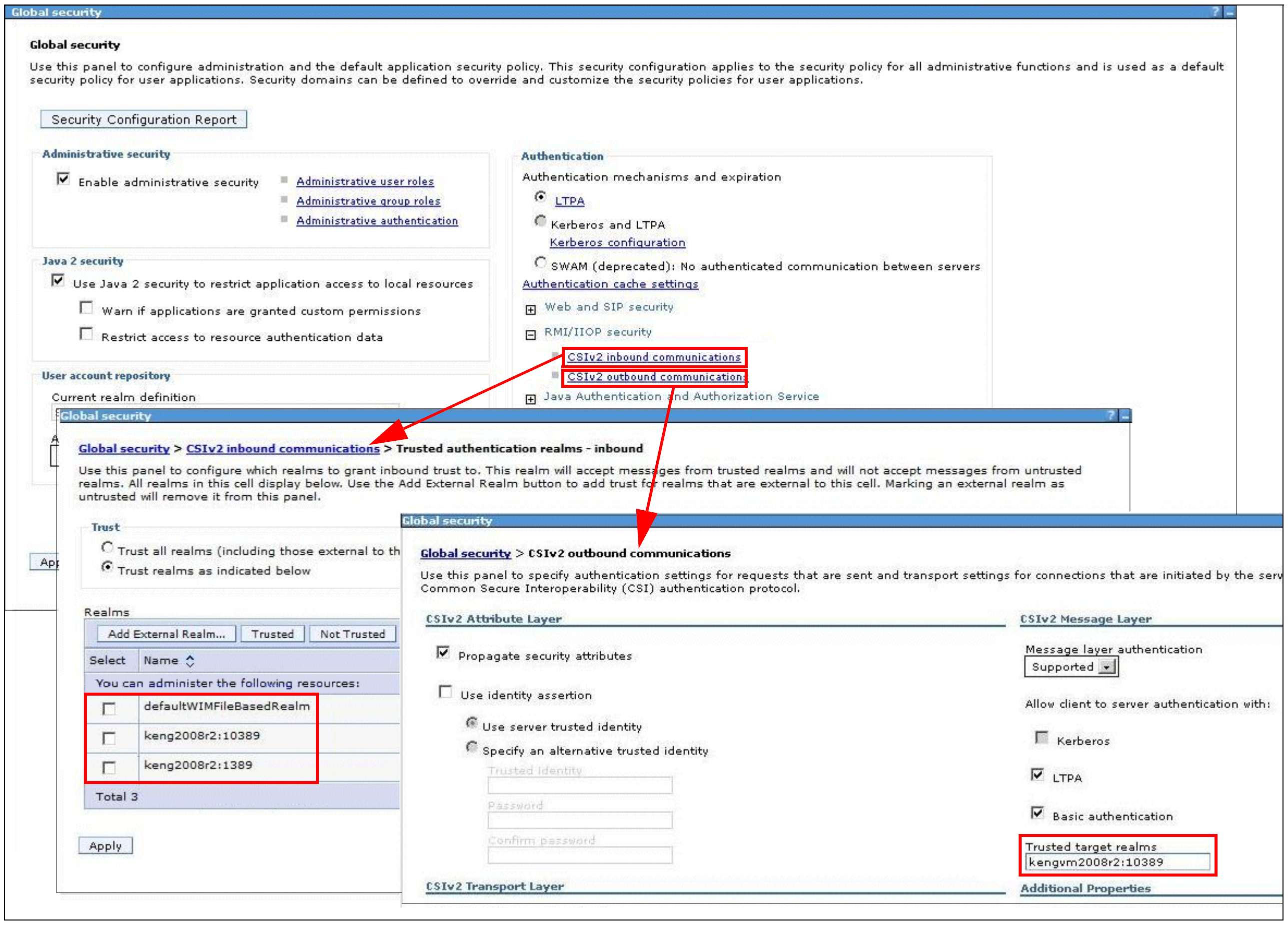
*Figure 19 Inbound trust configuration*

Add the job manager realm in the target nodes. Then save the changes and restart the nodes. In a deployment manager topology, you need to restart the deployment manager and all the nodes.

## Configuring inbound and outbound trust realm for the job managers

In a cross-realm environment where the job polling is done by the target nodes, the job manager needs to trust the inbound requests. Thus, you need to add the target node's security realm in the job manager's inbound and outbound trust, and then restart the job manager.

To configure the inbound trust, go to **Console** → **Global security** → **CSIv2 inbound communications** → **Trusted authentication realms—inbound**. To configure the outbound trust, go to **Console** → **Global security** → **CSIv2 inbound communications** → **Trusted authentication realms—outbound** (Figure 20).



Global security

**Global security**

Use this panel to configure administration and the default application security policy. This security configuration applies to the security policy for all administrative functions and is used as a default security policy for user applications. Security domains can be defined to override and customize the security policies for user applications.

Security Configuration Report

**Administrative security**

☑ Enable administrative security

- Administrative user roles
- Administrative group roles
- Administrative authentication

**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

**Authentication**

Authentication mechanisms and expiration

● LTPA

○ Kerberos and LTPA

Kerberos configuration

○ SWAM (deprecated): No authenticated communication between servers

Authentication cache settings

⊞ Web and SIP security

⊟ RMI/IIOP security

- CSIv2 inbound communications
- CSIv2 outbound communications

⊞ Java Authentication and Authorization Service

Global security

Global security > CSIv2 inbound communications > **Trusted authentication realms - inbound**

Use this panel to configure which realms to grant inbound trust to. This realm will accept messages from trusted realms and will not accept messages from untrusted realms. All realms in this cell display below. Use the Add External Realm button to add trust for realms that are external to this cell. Marking an external realm as untrusted will remove it from this panel.

**Trust**

○ Trust all realms (including those external to th

● Trust realms as indicated below

Realms

Add External Realm... | Trusted | Not Trusted

| Select | Name |
|---|---|
| You can administer the following resources: | |
| ☐ | defaultWIMFileBasedRealm |
| ☐ | keng2008r2:10389 |
| ☐ | keng2008r2:1389 |
| Total 3 | |

Apply

Global security

Global security > **CSIv2 outbound communications**

Use this panel to specify authentication settings for requests that are sent and transport settings for connections that are initiated by the serv Common Secure Interoperability (CSI) authentication protocol.

**CSIv2 Attribute Layer**

☑ Propagate security attributes

☐ Use identity assertion

● Use server trusted identity

○ Specify an alternative trusted identity

Trusted identity

Password

Confirm password

**CSIv2 Transport Layer**

**CSIv2 Message Layer**

Message layer authentication: Supported

Allow client to server authentication with:

☐ Kerberos

☑ LTPA

☑ Basic authentication

Trusted target realms: kengvm2008r2:10389

**Additional Properties**

*Figure 20 Configure the inbound and outbound trust configuration*

**Outbound trust configuration:** The outbound trust configuration is not required for the job manager, but to avoid problems (SECJ5008W) when trying to make a downstream request, it is a preferred practice to add the outbound trust also.

## Administrative authorization consideration

After the cross-realm trust is established, the user submitting the job needs to have the required authorization in the target nodes. In this example, for Alice to submit and execute the job, she needs to be an authenticable user with the required administrative roles defined in the source and the target nodes. In this topology, she is a valid user only in the job manager realm (LDAP1:389). All the other target nodes are configured with different user registries.

At the source, Alice is an authorized operator (minimum level) in the job manager.

At the target, in the federated deployment manager topology, Mary is the administrator. In the administrative agent topology, where the WebSphere Application Server profile is registered with the administrative agent, Mark is the administrator for the managed WebSphere Application Server node.

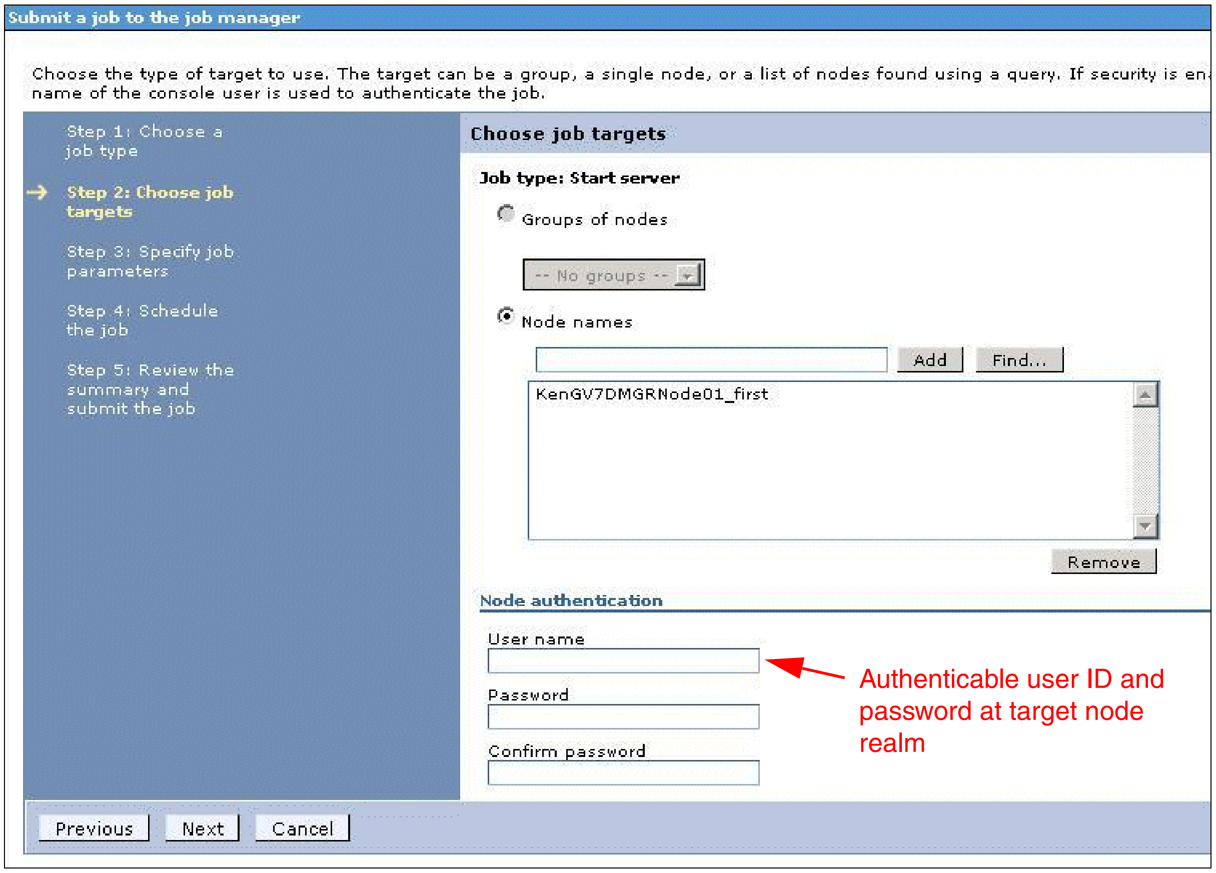
## Submitting and executing a job

In this topology, the following options are available when considering an authenticable user for the job submission and execution:

- Supplying a user name and password during job submission
- Adding the external realm's user in the authorization table for identity propagation

### Supplying a user name and password during job submission

After successfully logging in to the job manager, Alice can submit the job. However, for the job to be executed successfully at the target nodes, Alice needs to enter the authenticable user ID and password in the Node authentication pane during job submission itself (Figure 21).


Submit a job to the job manager

Choose the type of target to use. The target can be a group, a single node, or a list of nodes found using a query. If security is en
name of the console user is used to authenticate the job.

Step 1: Choose a job type
Step 2: Choose job targets
Step 3: Specify job parameters
Step 4: Schedule the job
Step 5: Review the summary and submit the job

Choose job targets

Job type: Start server

Groups of nodes

-- No groups --

[✓ Node names]

KenGV7DMGRNode01_first

Node authentication

User name

Password

Confirm password

*Figure 21 Node authentication pane*

Alice needs to know the authenticable user name and password for the target nodes, as follows:

- To submit the job to the deployment manager profile (LDAP2:389), Alice must know Mary's user name and password and must enter that information in the Node authentication pane.
- To submit the job to the WebSphere Application Server profile2 (LDAP4:389), Alice must know Mark's user name and password and must enter that information in the Node authentication pane.

**User name and password note:** The user name and password are passed from the job manager to the target nodes and are used to log in. This user name and password must be valid at the target nodes' realm. In addition, when you submit a job to multiple target nodes, the user ID and password credentials that are submitted with the job must be applicable to all of the job targets.

**Challenge of sharing user credentials:** This approach presents a challenge in a real-world environment where people are reluctant to share user credentials. Mary or Mark can say no to Alice's request for credential sharing.

### Adding the external realm's user in the authorization table for identity propagation

Identity propagation is not possible in an environment that uses different user registries, because the user Alice is not an authenticable user in the target nodes. To execute the job (that Alice submitted from the job manager) at the target nodes, Alice must use one of the following methods:

- Borrow the user name and password from someone who is an authenticable user in the target nodes, and enter it in the Node authentication pane during job submission itself.
- To avoid using someone else's credentials, add Alice to all the other user registries (for example, LDAP2:389 and LDAP4:389) and the target node's administrative authorization table (`admin-authz.xml`). Note, however, that adding the same user in multiple registries just for job execution results in duplicate entries across the organization. In addition, having one user in multiple realms with the same or a different password is not a preferred practice.

### Adding Alice's accessId in the target nodes' administrative authorization table

Instead of the methods described in the previous section, which have their limitations, you can add the external realm user (for example, Alice) into the target nodes' authorization table directly. This method allows you to overcome the problems of using someone else's credentials or duplicating Alice's user ID in all the user registries. WebSphere provides an option to achieve this configuration with the use of *accessId* (which is the realm-qualified uniqueId of the user). By adding the accessId in the target nodes' administrative authorization table directly, identity propagation can be achieved.

To add the accessId in the target nodes' administrative authorization table, you need to complete the following steps:

1. Get Alice's accessId from the source's realm (for example, job manager—LDAP1:389).
2. Add that accessId to the target nodes' administrative authorization tables.
3. Review and confirm the results.
4. Synchronize the node agents, and then restart the deployment manager, nodes, and all the application servers.

Currently, you can use this method only with wsadmin and not with the administrative console.

## Getting Alice’s accessId from the source realm

The job submission source is always the job manager.

Start wsadmin from the job manager profile’s `bin` directory, and get Alice’s accessId from the job manager security realm. Use the following commands:

```
<jobmanager_profile_root>/bin>wsadmin
wsadmin>$AdminTask listRegistryUsers {-securityRealmName LDAP1:389
-displayAccessIds true -userFilter Alice}
```

This command results in the following output:

```
{accessId user:LDAP1:389/CN=Alice,CN=Users,DC=ibm,DC=com} {name CN=
alice,CN=Users,DC=ibm,DC=com@LDAP1:389}
```

## Adding Alice’s accessId to the target nodes’ administrative authorization table

The target node varies based on the topology.

In a federated deployment manager topology, you need to add Alice’s accessId in the deployment manager’s administrative authorization table.

In an administrative agent topology, where a WebSphere Application Server profile is registered with the administrative agent, you need to add Alice’s accessId in the managed WebSphere Application Server nodes’ administrative authorization table. There is no requirement to add Alice’s accessId in the administrative agent authorization table itself.

Use the following commands:

```
<dmgr/node_profile_root>/bin>wsadmin
wsadmin>$AdminTask mapUsersToAdminRole {-roleName operator -userids { CN=
alice,CN=Users,DC=ibm,DC=com@LDAP1:389} -accessids {
user:LDAP1:389/CN=Alice,CN=Users,DC=ibm,DC=com }}
```

Save the changes using the following command:

```
wsadmin>$AdminConfig save
```

## Reviewing and confirming the results

Review the `admin-authz.xml` file (the administrative authorization table) to confirm that it includes the following entry:

```
<dmgr/node_profile_root>/config/cells/<cell_name>/admin-authz.xml
```

Example 1 shows an example of this file.

*Example 1 The admin-authz.xml file*

```
<?xml version="1.0" encoding="UTF-8"?>
<rolebasedauthz:AuthorizationTableExt xmi:version="2.0"
xmlns:xmi="http://www.omg.org/XMI"
xmlns:rolebasedauthz="http://www.ibm.com/websphere/appserver/schemas/5.0/rolebased
authz.xmi" xmi:id="AuthorizationTableExt_1" context="domain">
<authorizations xmi:id="RoleAssignmentExt_1" role="SecurityRoleExt_1">
    .....
<users xmi:id="UserExt_1302026833218" name=" CN=
alice,CN=Users,DC=ibm,DC=com@LDAP1:389" accessId="
user:LDAP1:389/CN=Alice,CN=Users,DC=ibm,DC=com "/>
```

```
<specialSubjects xmi:type="rolebasedauthz:ServerExt" xmi:id="ServerExt_1"/>
<specialSubjects xmi:type="rolebasedauthz:PrimaryAdminExt"
xmi:id="PrimaryAdminExt_1"/>
  </authorizations>
```

### Synchronizing

In a deployment manager federated environment, you need to synchronize the node agents and then restart the deployment manager, nodes, and all the application servers.

This completes the task of adding the remote user Alice from the job manager LDAP1:389 realm into the target nodes' authorization table (`admin-authz.xml`).

**accessId note:** Although Alice is in the target nodes' authorization table, she does not qualify for direct console access or any direct administrative activities in the target nodes. The accessId is used only for identity propagation from the job manager.

Now Alice can submit the job without specifying the user name and password in the Node authentication pane (Figure 14 on page 27) similar to a single registry environment.

**Identity token:** The identity token is propagated from the job manager to the target nodes. The user assumes the role that is defined at the target nodes even though that user is not in the user target nodes' realm.

If the accessId is added for a foreign realm in the target nodes, do not use the user name and password in the Node authentication pane. The identity propagation takes care of this information automatically.

**User name and password:** Specifying the user name and password in this pane overwrites the identity token that is passed from the job manager and acts as an actual user in the target node realm. If that user is not in that target node realm, the job fails.

For information about assigning users from a foreign realm to the `admin-authz.xml` file, see this URL:

`http://www14.software.ibm.com/webapp/wsbroker/redirect?version=compass&product=was-nd-mp&topic=tsec_7assign_users_adminauthz`

## References

For more information about the topics presented in this paper, consult these resources:

- WebSphere Application Server Version 7.0 Information Center

  `http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/index.jsp`

- IBM developerWorks® article, *System administration for WebSphere Application Server V7: Part 1--An overview of administrative enhancements*

  `http://www.ibm.com/developerworks/websphere/techjournal/0811_apte/0811_apte.html`

- IBM developerWorks article, *System administration for WebSphere Application Server V7: Part 2: New administrative topologies*

  http://www.ibm.com/developerworks/websphere/techjournal/0901_cundiff/0901_cundiff.html

- IBM developerWorks article, *System administration for WebSphere Application Server V7: Part 3: Administering a flexible management topology*

  http://www.ibm.com/developerworks/websphere/techjournal/0903_khalil/

## The team who wrote this paper

**Ganesan Karuppaiah** is a subject matter expert (SME) and IBM Certified IT Specialist with the WebSphere Application Server Technical Support Organization. His areas of expertise include WebSphere Scripting Administration (wsadmin), system management, application management, cryptography, and security. He is currently an Open Group Master Certified IT Specialist. Ganesan has been with the IBM Corporation for 10 years and supports multiple versions of WebSphere Application Server, from V3.5 to V8.

Thanks to the following people for their contributions to this project:

Carla Sadtler, Margaret Ticknor, Debbie Willmschen
International Technical Support Organization, Rochester Center

Michael Cheng
IBM US

Bill Holtzhauser
IBM US

## Now you can become a published author, too!

## Stay connected to IBM Redbooks publications

- Look for us on LinkedIn:

  `http://www.linkedin.com/groups?home=&gid=2130806`

- Explore new Redbooks® publications, residencies, and workshops with the IBM Redbooks weekly newsletter:

  `https://www.redbooks.ibm.com/Redbooks.nsf/subscribe?OpenForm`

- Stay current on recent Redbooks publications with RSS Feeds:

  `http://www.redbooks.ibm.com/rss.html`

# Notices

This document REDP-4839-00 was created or updated on March 29, 2012.

Redpaper™

## Trademarks