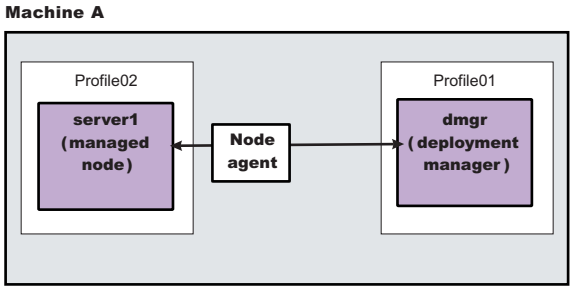
server, called server1, and default applications. An Application Server node becomes a managed node after federating the node into the deployment manager cell.

The deployment manager provides the administration for all managed nodes that are in its cell. Periodically the configuration and application files on a managed node refresh from the master copy of the files hosted on the deployment manager during *synchronization.*

In certain secure environments, the Profile creation wizard cannot federate a custom profile into a cell. Such cases require you to use the **addNode** command instead. If you have configured the deployment manager to use a JMX connector type other than the default SOAP connector, use the **addNode** command to add the node to the cell.

The deployment manager provides the administration for all managed nodes that are in its cell. Periodically the deployment manager refreshes the configuration files and application files on the managed node. Copying the master version of the files hosted on the deployment manager to the managed nodes is a process called *synchronization*.

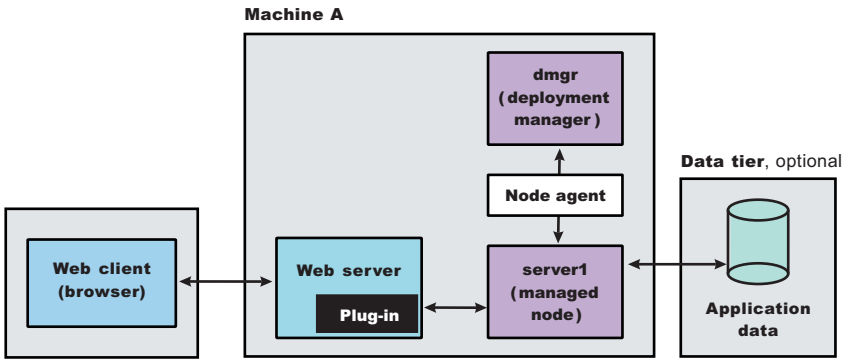
In a cell environment, only the managed nodes serve applications, not the deployment manager. The managed node in this scenario uses its internal HTTP transport chain for communication, which is suitable for an application with a relatively low request work load. For example, this type of installation can support a simple test environment or a departmental intranet environment.



1. Install WebSphere Application Server Network Deployment.
2. Create a deployment manager profile using the Profile creation wizard.
3. Start the deployment manager using the First steps console or the **startManager** command.
4. Create an Application Server profile using the Profile creation wizard.
5. Start the application server using the First steps console or the **startServer server1** command.
6. Add the application server node to the cell using the administrative console of the deployment manager. Click **System Administration > Nodes** to add the node.

- **Scenario 6:** Install a cell of managed application server nodes and a Web server on one machine.

  Installing a Web server, such as IBM HTTP Server, on the same machine as the application server provides a richer set of configuration options. Installing a Web server plug-in is required for the Web server to communicate with the server in the managed node. This type of installation can support either rigorous testing in a cell environment or production environments that do not require a firewall.



1. Install WebSphere Application Server Network Deployment.

Table 13. Web server plug-in configuration properties (continued)

In the administrative console, click **Servers > Web Servers >** *Web_server_name* **> Plug-in properties > Request and Response**	Priority used by the IIS Web server when loading the plug-in configuration file	IISPluginPriority
In the administrative console, click **Servers > Web Servers >** *Web_server_name* **> Plug-in properties > Caching**	Enable Edge Side Include (ESI) processing to cache the responses	ESIEnable
In the administrative console, click **Servers > Web Servers >** *Web_server_name* **> Plug-in properties > Caching**	Maximum cache size	ESIMaxCacheSize
In the administrative console, click **Servers > Web Servers >** *Web_server_name* **> Plug-in properties > Caching**	Enable invalidation monitor to receive notifications	ESIInvalidationMonitor

Web server plug-in connections

The WebSphere Application Server Web server plug-ins are used to establish and maintain persistent HTTP and HTTPS connections to Application Servers .

When the plug-in is ready to send a request to the application server, it first checks its connection pool for existing connections. If an existing connection is available the plug-in checks its connection status. If the status is still good, the plug-in uses that connection to send the request. If a connection does not exist, the plug-in creates one. If a connection exists but has been closed by the application server, the plug-in closes that connection and opens a new one.

After a connection is established between a plug-in and an application server, it will not be closed unless the application server closes it for one of the following reasons:

- If the **Use Keep-Alive** property is selected and the time limit specified on the **Read timeout** or **Write timeout** property for the HTTP inbound channel has expired.
- The maximum number of persistent requests which can be processed on an HTTP inbound channel has been exceeded. (This number is set using the HTTP inbound channel's **Maximum persistent requests** property.)
- The Application Server is shutting down.

Even if the application server closes a connection, the plug-in will not know that it has been closed until it tries to use it again. The connection will be closed if one of the following events occur:

- The plug-in receives a new HTTP request and tries to reuse the existing connection.
- The number of **httpd** processes drop because the Web server is not receiving any new HTTP requests. (For the IHS Web server, the number of **httpd** processes that are kept alive depends on the value specified on the Web server's MinSpareServers directive.)
- The Web server is stopped and all **httpd** processes are terminated, and their corresponding sockets are closed.

Note: Sometimes, if a heavy request load is stopped or decreased abruptly on a particular application server, a lot of the plug-in's connections to that application server will be in CLOSE_WAIT state. Because these connections will be closed the first time the plug-in tries to reuse them, having a large number of connections in CLOSE-WAIT state should not affect performance

Specifies the name of the node for the deployment manager. The default is the name of the host computer on which the deployment manager is installed with `CellManager##` appended, where *##* is a two-digit number.

Data type	String
Default	*host_name*CellManager01

State:

Indicates the state of the deployment manager. The state is *Started* when the deployment manager is running and *Stopped* when it is not running.

Data type	String
Default	Started

Node

A node is a logical grouping of managed servers.

A node usually corresponds to a logical or physical computer system with a distinct IP host address. Nodes cannot span multiple computers. Node names usually are identical to the host name for the computer.

Nodes in the network deployment topology can be managed or unmanaged. Two types of managed nodes exist while one type of unmanaged node exists.

One type of managed node has a node agent which manages all servers on a node, whether the servers are WebSphere Application Servers, Java Message Service (JMS) servers (on Version 5 nodes only), Web servers, or generic servers. The node agent represents the node in the management cell. A deployment manager manages this type of managed node. The other type of managed node has no node agent. This type of managed node is defined on a standalone Application Server. The deployment manager cannot manage this standalone Application Server. An standalone Application Server can be federated. When it is federated, a node agent is automatically created. The node becomes a managed node in the cell. The deployment manager manages this node.

An unmanaged node does not have a node agent to manage its servers. Unmanaged nodes can have server definitions such as Web servers in the WebSphere Application Server topology. Unmanaged nodes can never be federated. That is, a node agent can never be added to an unmanaged node.

A supported Web server can be on a managed node or an unmanaged node. You can define only one Web server to a standalone WebSphere Application Server node. This Web server is defined on an unmanaged node. You can define Web servers to the deployment manager. These Web servers can be defined on managed or unmanaged nodes.

WebSphere Application Server supports basic administrative functions for all supported Web servers. For example, generation of a plug-in configuration can be performed for all Web servers. However, propagation of a plug-in configuration to remote Web servers is supported only for IBM HTTP Servers that are defined on an unmanaged node. If the Web server is defined on a managed node, propagation of the plug-in configuration is done for all the Web servers by using node synchronization. The Web server plug-in configuration file is created according to the Web server definition and is created based on the list of applications that are deployed on the Web server. You can also map all supported Web servers as potential targets for the modules during application deployment.

```
 grant codeBase "file:${application}" {
// The following are required by Java mail
permission java.io.FilePermission "${was.install.root}${/}java${/}jre${/}lib${/}ext${/}mail.jar", "read";
permission java.io.FilePermission "${was.install.root}${/}java${/}jre${/}lib${/}ext${/}activation.jar", "read";
permission java.io.FilePermission "${user.home}${/}.mailcap", "read";
permission java.io.FilePermission "${java.home}${/}lib${/}mailcap", "read";
};
```

Mail: Resources for learning

Use the following links to find relevant supplemental information about the JavaMail API. The information resides on IBM and non-IBM Internet sites, whose sponsors control the technical accuracy of the information.

These links are provided for convenience. Often, the information is not specific to the IBM WebSphere Application Server product, but is useful all or in part for understanding the product. When possible, links are provided to technical papers and Redbooks that supplement the broad coverage of the release documentation with in-depth examinations of particular product areas.

Programming model and decisions

- JavaMail documentation

Programming specifications

- JavaMail 1.3 API documentation (Sun Java specifications)

JavaMail support for IPv6

WebSphere Application Server Version 6.0 and its JavaMail component support Internet Protocol Version 6.0 (IPv6), meaning that:

- Both can run on a pure IPv4 network, a pure IPv6 network, *or* a mixed IPv4 and IPv6 network.
- On either the pure IPv6 network or the mixed network, the JavaMail component works with mail servers (such as the SMTP mail transfer agent, and the IMAP and POP3 mail stores) that are also IPv6 compatible. Additionally, a JavaMail component that is run on the mixed IPv4 and IPv6 network can communicate with mail servers using IPv4.

Use of brackets with IPv6 addresses

When you configure a mail session, you can specify the mail server hosts (also known as mail transport and mail store hosts) with domain-qualified host names or numerical IP addresses. Using host names is generally the preferred method. If you use IP addresses, however, consider enclosing IPv6 addresses in square brackets to prevent parsing inaccuracies. See the following example:

```
[fe80::202:57ff:fec4:2334]
```

The JavaMail API requires a combination of many host names or IP addresses with a port number, using the `host:port number` syntax . This extra colon can cause the port number to be read as part of an IPv6 address. Using brackets prevents your JavaMail implementation from processing the extra characters erroneously.

Using URL resources within an application

Java 2 Enterprise Edition (J2EE) applications can use Uniform Resource Locators (URLs) by looking up references to logically named URL connection factories through the `java:comp/env/url` subcontext, which is declared in the application deployment descriptor and mapped to installation specific URL resources. As in the case of other J2EE resources, this can be done in order to eliminate the need for the application to hard code references to external resources. The process is the same used with other J2EE resources, such as JDBC objects and JavaMail sessions.

1. Develop an application that relies on naming features.

the enterprise bean layer. For backward compatibility, WebSphere Application Server supports the Secure Authentication Service (SAS) security protocol, which was used in prior releases of WebSphere Application Server and other IBM products.
- **J2EE Security** - The security collaborator enforces Java 2 Platform, Enterprise Edition (J2EE)-based security policies and supports J2EE security APIs.
- **WebSphere Security** - WebSphere Application Server security enforces security policies and services in a unified manner on access to Web resources, enterprise beans, and JMX administrative resources. It consists of WebSphere Application Server security technologies and features to support the needs of a secure enterprise environment.

WebSphere Application Server Network Deployment installation: The following figure shows a typical multiple-tier business computing environment for a WebSphere Application Server Network Deployment installation.

Important: There is a node agent instance on every computer node.

Each product application server consists of a Web container, an EJB container, and the administrative subsystem. The WebSphere Application Server deployment manager contains only WebSphere administrative code and the administrative console. The administrative console is a special J2EE Web application that provides the interface for performing administrative functions. WebSphere Application Server configuration data is stored in XML descriptor files, which must be protected by operating system security. Passwords and other sensitive configuration data can be modified using the administrative console. However, you must protect these passwords and sensitive data. For more information, see “Protecting plain text passwords” on page 1614.

The administrative console Web application has a setup data constraint that requires the administrative console servlets and JSP files to be accessed only through an SSL connection when global security is enabled.

After installation, the administrative console HTTPS port is configured to use **DummyServerKeyFile.jks** and **DummyServerTrustFile.jks** with the default self-signed certificate. Using the dummy key and trust file certificate is not safe and you need to generate your own certificate to replace dummy ones immediately. It is more secure if you first enable global security and complete other configuration tasks after global security is enforced.

Security considerations specific to a multi-node or process Network Deployment environment

WebSphere Application Server Network Deployment allows for centralized management of distributed nodes and application servers. This inherently brings complexity, especially when security is included into the mix. Because everything is distributed, security plays an even larger role in ensuring that communications are appropriately secure between applications servers and node agents, and between node agents (a node specific configuration manager) and the Deployment Manager (a domain-wide, centralized configuration manager). The following issues should be considered when operating in this environment, but preferably prior to going to this environment.

Because the processes are distributed, the authentication mechanism that must be used is LTPA. The LTPA tokens are encrypted and signed and therefore, forwardable to remote processes. However, the tokens have expirations. The SOAP connector (the default connector) used for administrative security does not have retry logic for expired tokens, however, the protocol is stateless so a new token is created for each request (if there is not sufficient time to execute the request with the given time left in the token). An alternative connector is the RMI connector, which is stateful and has some retry logic to correct expired tokens by resubmitting the requests after the error is detected. Also, because tokens have time-specific expiration, the synchronization of the system clocks are crucial to the proper operation of token-based validation. If the clocks are off by too much (approximately 10-15 minutes), you can encounter unrecoverable validation failures that can be avoided by having them in sync. Verify that the clock time, date, and time zones are all the same between systems. It is acceptable for nodes to be across time zones, provided that the times are correct within the time zones (for example, 5 PM CST = 6 PM EST, and so on).

The following are issues to consider when using or planning for a Network Deployment environment.

1. When attempting run system management commands such as stopNode, you must explicitly specify administrative credentials to perform the operation. Most commands accept `-username` and `-password` parameters to specify the user ID and password, respectively. The user ID and password that are specified should be an administrative user; for example, a user who is a member of the console users with **Operator** or **Administrator** privileges or the administrative user ID configured in the user registry. An example for stopNode, `stopNode -username user -password pass`.
2. Verify that the configuration at the node agents are always synchronized with the Deployment Manager prior to starting or restarting a node. To manually get the configuration synchronized, issue the syncNode command from each node that is not synchronized. To synchronize the configuration for node agents that are started, click **System Administration** > **Nodes** and select all started nodes. Click **Synchronize**.
3. Verify that the clocks on all systems are in sync including the time zone, time and date. If they are out of sync, the tokens expire immediately when they reach the target server due to the time differences.
4. Verify that the LTPA token expiration period is long enough to complete your longest downstream request. Some credentials are cached and therefore the timeout does not always count in the length of the request.
5. The administrative connector used by default for system management is SOAP. SOAP is a stateless HTTP protocol. For most situations, this connector is sufficient. When running into a problem using the SOAP connector it might be desirable to change the default connector on all servers from SOAP to RMI. The RMI connector uses CSIv2, a stateful or interoperable protocol, and can be configured to use identity assertion (downstream delegation), message layer authentication (BasicAuth or Token), and client certificate authentication (for server trust isolation). To change the default connector on a given server, go to **Administration Services** in Additional Properties for that server.
6. The following error message might occur within the administrative subsystem security. This indicates that the sending process did not supply a credential to the receiving process. Typically the causes for this problem are:
   - The sending process has security disabled while the receiving process has security enabled. This typically indicates one of the two processes are not in sync with the cell.