The standalone/deployments directory in the JBoss Application Server

distribution is the location end users can place their deployment content

(e.g. war, ear, jar, sar files) to have it automatically deployed into the server

runtime.

Users, particularly those running production systems, are encouraged to use the

JBoss AS management APIs to upload and deploy deployment content instead of

relying on the deployment scanner subsystem that periodically scans this

directory. See the JBoss AS documentation for details.

DEPLOYMENT MODES

The filesystem deployment scanner in AS 7 and later works differently from

previous JBoss AS releases. The scanner can operate in one of two different

modes, depending on whether it will directly monitor the deployment content

in order to decide to deploy (or redeploy) it.

1) Auto-deploy mode: The scanner will directly monitor the deployment content,

automatically deploying new content and redeploying content whose timestamp

has changed. This is similar to the behavior of previous AS releases, although

there are differences:

a) A change in any file in an exploded deployment triggers redeploy. Because

EE 6 applications do not require deployment descriptors, there is no attempt

to monitor deployment descriptors and only redeploy when a deployment

descriptor changes.

b) The scanner will place marker files in this directory as an indication of

the status of its attempts to deploy or undeploy content. These are detailed

below.

2) Manual deploy mode: The scanner will not attempt to directly monitor the

deployment content and decide if or when the end user wishes the content to

be deployed or undeployed. Instead, the scanner relies on a system of marker

files, with the user's addition or removal of a marker file serving as a sort

of command telling the scanner to deploy, undeploy or redeploy content.

Auto-deploy mode and manual deploy mode can be independently configured for

zipped deployment content and exploded deployment content. This is done

via the "auto-deploy" attributes on the deployment-scanner element in the

standalone.xml configuration file:

<deployment-scanner scan-interval="5000" relative-to="jboss.server.base.dir"

path="deployments" auto-deploy-zipped="true" auto-deploy-exploded="false"/>

By default, auto-deploy of zipped content is enabled, and auto-deploy of

exploded content is disabled. Manual deploy mode is strongly recommended for

exploded content, as exploded content is inherently vulnerable to the scanner

trying to auto-deploy partially copied content. Manual deploy mode also allows

deployment resources (e.g. html and css files) to be replaced without

triggering a redeploy of the application.

MARKER FILES

The marker files always have the same name as the deployment content to which

they relate, but with an additional file suffix appended. For example, the

marker file to indicate the example.war file should be deployed is named

example.war.dodeploy. Different marker file suffixes have different meanings.

The relevant marker file types are:

.dodeploy -- Placed by the user to indicate that the given content should

be deployed into the runtime (or redeployed if already

deployed in the runtime.)

.skipdeploy -- Disables auto-deploy of the content for as long as the file

is present. Most useful for allowing updates to exploded

content without having the scanner initiate redeploy in the

middle of the update. Can be used with zipped content as

well, although the scanner will detect in-progress changes

to zipped content and wait until changes are complete.

.isdeploying -- Placed by the deployment scanner service to indicate that it

has noticed a .dodeploy file or new or updated auto-deploy

mode content and is in the process of deploying the content.

This marker file will be deleted when the deployment process

completes.

.deployed -- Placed by the deployment scanner service to indicate that the

given content has been deployed into the runtime. If an end

user deletes this file, the content will be undeployed.

.failed -- Placed by the deployment scanner service to indicate that the

given content failed to deploy into the runtime. The content

of the file will include some information about the cause of

the failure. Note that with auto-deploy mode, removing this

file will make the deployment eligible for deployment again.

.isundeploying -- Placed by the deployment scanner service to indicate that it

has noticed a .deployed file has been deleted and the

content is being undeployed. This marker file will be deleted

when the undeployment process completes.

.undeployed -- Placed by the deployment scanner service to indicate that the

given content has been undeployed from the runtime. If an end

user deletes this file, it has no impact.

.pending -- Placed by the deployment scanner service to indicate that it

has noticed the need to deploy content but has not yet

instructed the server to deploy it. This file is created if

the scanner detects that some auto-deploy content is still in

the process of being copied or if there is some problem that

prevents auto-deployment. The scanner will not instruct the

server to deploy or undeploy any content (not just the

directly affected content) as long as this condition holds.

Basic workflows:

All examples assume variable $AS points to the root of the JBoss AS distribution.

Windows users: the examples below use Unix shell commands; see the "Windows

Notes" below.

A) Add new zipped content and deploy it:

1. cp target/example.war $AS/standalone/deployments

2. (Manual mode only) touch $AS/standalone/deployments/example.war.dodeploy

B) Add new unzipped content and deploy it:

1. cp -r target/example.war/ $AS/standalone/deployments

2. (Manual mode only) touch $AS/standalone/deployments/example.war.dodeploy

C) Undeploy currently deployed content:

1. rm $AS/standalone/deployments/example.war.deployed

D) Auto-deploy mode only: Undeploy currently deployed content:

1. rm $AS/standalone/deployments/example.war

Note that this approach is not recommended with unzipped content as the server

maintains no other copy of unzipped content and deleting it without first

triggering an undeploy temporarily results in a live application with

potentially critical resources no longer available. For unzipped content use

the 'rm $AS/standalone/deployments/example.war.deployed' approach.

E) Replace currently deployed zipped content with a new version and deploy it:

1. cp target/example.war/ $AS/standalone/deployments

2. (Manual mode only) touch $AS/standalone/deployments/example.war.dodeploy

F) Manual mode only: Replace currently deployed unzipped content with a new

version and deploy it:

1. rm $AS/standalone/deployments/example.war.deployed

2. wait for $AS/standalone/deployments/example.war.undeployed file to appear

3. cp -r target/example.war/ $AS/standalone/deployments

4. touch $AS/standalone/deployments/example.war.dodeploy

G) Auto-deploy mode only: Replace currently deployed unzipped content with a

new version and deploy it:

1. touch $AS/standalone/deployments/example.war.skipdeploy

2. cp -r target/example.war/ $AS/standalone/deployments

3. rm $AS/standalone/deployments/example.war.skipdeploy

H) Manual mode only: Live replace portions of currently deployed unzipped

content without redeploying:

1. cp -r target/example.war/foo.html $AS/standalone/deployments/example.war

I) Auto-deploy mode only: Live replace portions of currently deployed unzipped

content without redeploying:

1. touch $AS/standalone/deployments/example.war.skipdeploy

2. cp -r target/example.war/foo.html $AS/standalone/deployments/example.war

J) Manual or auto-deploy mode: Redeploy currently deployed content (i.e. bounce

it with no content change):

1. touch $AS/standalone/deployments/example.war.dodeploy

K) Auto-deploy mode only: Redeploy currently deployed content (i.e. bounce

it with no content change):

1. touch $AS/standalone/deployments/example.war

Windows Notes:

The above examples use Unix shell commands. Windows equivalents are:

cp src dest --> xcopy /y src dest

cp -r src dest --> xcopy /e /s /y src dest

rm afile --> del afile

touch afile --> echo>> afile

Note that the behavior of 'touch' and 'echo' are different but the

differences are not relevant to the usages in the examples above.