**JBoss Enterprise Application Platform 6.4**

C:\Users\ADMIN\Downloads\1\Administration and Configuration Guide_files\title_logo.png

**Jboss Domain Mode**

**JBOSS Domain Mode**

Domain mode feature of JBoss allows to manage multiple server instances from a single control point.

preconfigured profiles:

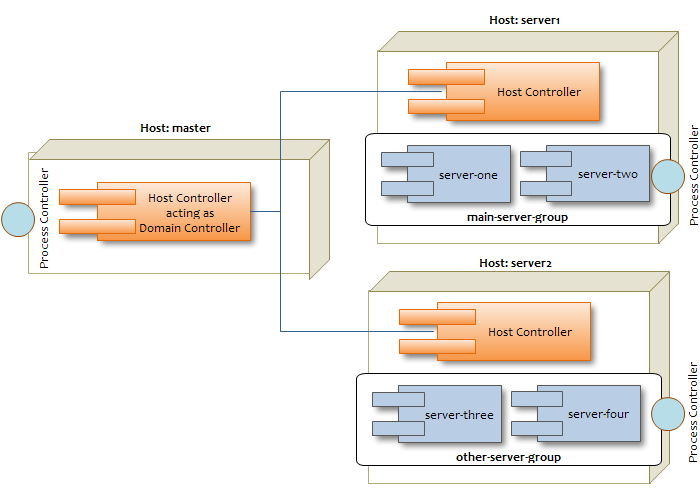
* **default** - Support of Java EE Web-Profile plus some extensions like RESTFul Web Services or support for EJB3 remote invocations
* **full** - Support of Java EE Full-Profile and all server capabilities without clustering
* **ha** - default profile with clustering capabilities
* **full-ha** - full profile with clustering capabilities

**Managed Domain**

One of the primary new features of JBoss is the ability to manage multiple JBoss instances from a single control point. A collection of such servers is referred to as the members of a "domain" with a single Domain Controller process acting as the central management control point. All of the JBoss Application Server 6.4 instances in the domain share a common management policy, with the Domain Controller acting to ensure that each server is configured according to that policy. Domains can span multiple physical (or virtual) machines, with all JBoss Application Server instances on a given host under the control of a special **Host Controller process**. One Host Controller instance is configured to act as the central Domain Controller. The Host Controller on each host interacts with the Domain Controller to control the lifecycle of the application server instances running on its host and to assist the Domain Controller in managing them.

When you launch a JBoss Application Server 6.4 managed domain on a host (via the domain.sh or domain.bat launch scripts) your intent is to launch a Host Controller and usually at least one JBoss Application Server 6.4 instance. On one of the hosts the Host Controller should be configured to act as the Domain Controller.

The following is an example managed domain topology:



**Host**

Each "Host" box in the above diagram represents a physical or virtual host. A physical host can contain zero, one or more server instances.

**Host Controller**

When the domain.sh or domain.bat script is run on a host, a process known as a Host Controller is launched. The Host Controller is solely concerned with server management; it does not itself handle application server workloads. The Host Controller is responsible for starting and stopping the individual application server processes that run on its host, and interacts with the Domain Controller to help manage them.

Each Host Controller by default reads its configuration from the domain/configuration/host.xml file located in the unzipped JBoss Application Server 6.4 installation on its host's filesystem. The host.xml file contains configuration information that is specific to the particular host. Primarily:

* the listing of the names of the actual JBoss Application Server 6.4 instances that are meant to run off of this installation.
* configuration of how the Host Controller is to contact the Domain Controller to register itself and access the domain configuration. This may either be a configuration of how to find and contact a remote Domain Controller, or a configuration telling the Host Controller to itself act as the Domain Controller.
* configuration of items that are specific to the local physical installation. For example, named interface definitions declared in domain.xml (see below) can be mapped to an actual machine-specific IP address in host.xml. Abstract path names in domain.xml can be mapped to actual filesystem paths in host.xml.

**Domain Controller**

One Host Controller instance is configured to act as the central management point for the entire domain, i.e. to be the Domain Controller. The primary responsibility of the Domain Controller is to maintain the domain's central management policy, to ensure all Host Controllers are aware of its current contents, and to assist the Host Controllers in ensuring any running application server instances are configured in accordance with this policy. This central management policy is stored by default in the domain/configuration/domain.xml file in the unzipped JBoss Application Server 6.4 installation on Domain Controller's host's filesystem.

A domain.xml file must be located in the domain/configuration directory of an installation that's meant to run the Domain Controller. It does not need to be present in installations that are not meant to run a Domain Controller; i.e. those whose Host Controller is configured to contact a remote Domain Controller. The presence of a domain.xml file on such a server does no harm.

The domain.xml file includes, among other things, the configuration of the various "profiles" that JBoss Application Server 6.4 instances in the domain can be configured to run. A profile configuration includes the detailed configuration of the various subsystems that comprise that profile (e.g. an embedded JBoss Web instance is a subsystem; a JBoss TS transaction manager is a subsystem, etc). The domain configuration also includes the definition of groups of sockets that those subsystems may open. The domain configuration also includes the definition of "server groups":

We have two server groups: **main-server-group** and **other-server-group**. You can in turn associate each server group with a different profile.   
The default configuration includes four preconfigured profiles:

* **default** - Support of Java EE Web-Profile plus some extensions like RESTFul Web Services or support for EJB3 remote invocations
* **full** - Support of Java EE Full-Profile and all server capabilities without clustering
* **ha** - default profile with clustering capabilities
* **full-ha** - full profile with clustering capabilities

**Server Groups**  
Server groups are exactly what they sound like; logical groupings of server instances which allow you to apply all the settings in a certain profile to all the server instances within that group as shown in the example below, which applies the settings contained in the profile named "full" to all servers in the group "main-server-group":

<server-groups>

<server-group name="main-server-group" profile="full">

<jvm name="default">

<heap size="64m" max-size="512m"/>

<permgen size="256m" max-size="256m"/>

</jvm>

<socket-binding-group ref="full-sockets"/>

</server-group>

Within a server group configuration, JVM settings can be configured (as above). These JVM settings will be applied as a default to all the server instances within "main-server-group".  
  
  
**Server Instances**  
At the bottom of the host.xml file is where all the servers which will run on the physical machine are configured. Every server listed will run on the machine where that XML file is located, so it would not be unusual for each physical machine to have a different host.xml file, even if the only difference is the logical name of the server (e.g. "server-one").  
  
Also at the end of the host.xml file, just above the <servers> tag is another place to further configure the JVM:

<jvms>

<jvm name="default">

<heap size="64m" max-size="256m"/>

<jvm-options>

<option value="-server"/>

</jvm-options>

</jvm>

</jvms>

In this example, the name of the JVM is "default", the same as referenced in the server group "main-server-group" detailed above. Because there are two named JVM configurations with the same name, the order of precedence decides which values are applied. The JVM configuration in the <server-group> of the domain.xml is applied first. This is then overridden by the values set in <jvms> in the host.xml file, so that all server instances on a host can be affected by certain values.