Configuring a host controller

Goals

After completing this section, students should be able to do the following:

- **Describe the configuration options for a host controller and make configuration changes to a host controller.**

Host Controller General Settings

A host controller is configured in a JBOSS_HOME/domain/configuration/host.xml file. This book has already introduced parts of this archive, but now it's time to start looking at its structure.

A host.xml file has the following general structure, as defined by the XML schema in JBOSS_HOME/docs/schema/wildfly-config_4_1.xsd:

```
<host name="my_hostname" xmlns="urn:jboss:domain:4.1">
    <extensions>
     ... host controller extensions
    </extensions>
    <system-properties> ...
     for defining system properties </system-
    properties> <paths> ... for
    defining
     filesystem paths </paths> <vault>


     ... for storing encrypted passwords </vault>

    <management> ...
     the management interfaces and their security settings appear here </management> <domain-
    controller> ... the
    settings for how to connect
     to the Domain Controller </domain-controller>

    <interfaces>
     ... interfaces are defined here </interfaces>
    <jvms> ... JVMs
    are
     defined here </jvms> <servers>


     ... servers are defined here
    </servers>
    <profile> ...
     subsystems configurations for host controller extensions </profile>  </host>
```

Most items are optional and may not be seen in the factory host controller configuration files. The only required elements are: <management>, <domain-controller>, and the <host> of the top-level element.

The following are very high level explanations about each item. Its usage and syntax are explained in more detail later in this course.

<extension>

Declares EAP extensions that should be loaded by the host controller itself, rather than by server instances.

<system-properties>

Defines system properties for this particular host. A system property defined here overrides the same system property defined in domain.xml (if the property is defined in both files), but NOT by a system property defined on the command line.

<paths>

Defines the actual file system paths used by the EAP subsystem configuration. This allows each host controller to map those routes to different physical locations.

<vault>

Defines a security store, a place for storing encrypted passwords.

<management>

Defines the management interfaces available to administrators, as well as their security and audit settings.

<interfaces>

It defines the actual IP addresses to which the different services and subsystems will bind.

<domain-controller>

Specifies whether this host controller can locate the domain controller or is the domain controller.

<jvms>

Defines multiple Java virtual machine configurations that can be referenced by different groups of servers, as well as different individual server instances.

<servers>

Defines the server instances on this host controller. Each server definition refers to a group of servers defined in the domain.xml configuration file.

<profile>

Specifies settings for subsystems provided by the <extension> element mentioned above.

Elements found in standalone.xml for standalone server operating mode generally have the same meaning and syntax in host.xml for managed domain operating mode. Some elements, such as <system properties> and <jvms> can be set in different places in host.xml and domain.xml. Sometimes this can happen more than once in the same file, in a way that may seem redundant. This is allowed because it is used to configure domain-wide defaults, which can be overridden at the host, pool, or server instance level.

A host configuration typically defines multiple servers within the <server> element. In this case, each server instance probably requires a different port offset, so there are no network port conflicts.

Explore the configuration of a host controller

Server and JVM configurations are the same for master and slave host controllers, even if it is common for a master to have neither.

The following is a sample, partial host.xml configuration file. The following text explains the server instance and JVM settings:

```
<host name="myhost"        (1)  xmlns="urn:jboss:domain:4.1">
        ...
        <jvms>  (2)
            <jvm name="default">
                <heap size="64m" max-size="128m"/> </jvm>
            <jvm
            name="myhost-jvm"> <heap
                size="512m" max-size="1024m"/> </jvm> </
            jvms>

        <servers>  (3)

            <server name="server1" group="group1">                  (4)

                <jvm name="default"/>  </  (5)
            server>

            <server name="server2" group="group2">                  (6)

                <jvm name="myhost-jvm"/>  (7)

                <socket-bindings port-offset="200"/>  </server>  (8)

        </servers>
</host>
```

1. The name of this host controller is myhost. If the name attribute is not specified, the host name of the computer is used instead.
2. Two JVMs are defined, named default and myhost-jvm. These JVM names must also be defined by domain.xml, since they can be referenced through server group definitions, rather than individual server instances, as in this sample.
3. Two server instances are defined, named server1 and server2.
4. The server instance server1 is a member of the server group group1 defined in domain.xml.
5. The server instance server1 uses the default JVM definition, so it gets very little heap, with a maximum size of only 128 MB.
6. The server instance server2 is a member of the server group group2 defined in domain.xml.
7. The server instance server2 uses the myhost-jvm JVM definition, so it gets a larger heap, with a maximum size of 1024 MB (1 GB).
8. The server instance server2 is configured with a port offset of 200, so its HTTP port will be 8280. The server instance server1 has no port offset, so it will get the default HTTP port 8080.

The sample host controller configuration above defines two server instances, so the host will run four OS processes, each with its own JVM: the process controller, the host controller, and the two server instances.

Configuring a slave host controller

The fundamental parts of configuring a slave host controller have already been introduced in this book, although a complete guide to a slave configuration has not yet been made. The following guide highlights the differences between the master and slave host controller settings:

As we already know, a host controller instance carries the host name of its computer, but this can be overridden by using the name attribute on the top-level <host> element at the beginning of the host.xml configuration file. For example:

```
<?xml version="1.0" ?> <host
xmlns="urn:jboss:domain:4.1" name="mydomainslave1">
 ...
```

To denote that a host controller is a slave, specify where the master can be found:

```
 ...
<domain-controller>
        <remote security-realm="ManagementRealm">
                <discovery-options> <static-
                        discovery name="primary" protocol="$
                                {jboss.domain.master.protocol:remote}" host="$
                                {jboss.domain.master.address}" port="$
                                {jboss.domain.master.port:9999}"/>
                </discovery-options> </
        remote>
</domain-controller>
 ...
```

Details about previous values have already been explained; review the previous section for more information.

A slave host controller must declare a native management interface in the same way that a master must. For example:

```
...
        </extensions>

        <management>
                ...
                <management-interfaces>
                        <native-interface security-realm="ManagementRealm"> <socket
                                interface="management" port="$
                                        {jboss.management.native.port:9999}"/> </native-interface> </
                        management-interfaces>
                </management>


        <domain-controller>
...
```

Remember that all administrative actions are performed on the domain controller. Therefore, there is no purpose in defining an HTTP management interface for a slave host controller.

The native management interface for a slave host controller references the management (network) interface, again the same way the master does:

```
...
        </domain-controller>

        <interfaces>
            <interface name="management">
                <inet-address value="${jboss.bind.address.management:127.0.0.1}"/>
            </interface>
            <interface name="public"> <inet-
                address value="${jboss.bind.address:127.0.0.1}"/>
            </interface> </
        interfaces>

        <jvm>
...
```

For a slave host controller, the public IS network interface is required, since it is supposed to have server instances.

Only the domain controller should communicate with a slave native management interface. It is recommended to enforce this restriction using firewall rules, and that is the main reason why the factory EAP slave host controller configuration defines the network interface management.

Start a slave host controller

At least two system properties must be defined to start a slave host controller:

- **jboss.domain.master.address – Provides the IP address of the domain controller.**
- **jboss.bind.address.management – Offer the slave with an IP address without loopback. This is required because the domain controller also connects to the slave.**

These variables are defined on the command line of the domain.sh script or their values are inserted directly into the slave host.xml configuration file.

Assuming that the master management interface IP address is 192.168.0.14 and the slave management interface IP address is 192.168.1.25, the following command starts the host controller as slave:

```
$ ./domain.sh -Djboss.bind.address.management=192.168.1.25 \
-Djboss.domain.master.address=192.168.0.14
```

In the example above, it is assumed that the native management interface TCP port was left as the default value of 9999 for both master and slave.

Configuring multiple master controllers

The ability to define multiple master controllers in the same slave host controller configuration is a new feature in EAP 7. For example:

```
<domain-controller>
    <remote security-realm="ManagementRealm">
        <discovery-options>
            <static-discovery name="primary"                    (1)
                protocol="${jboss.domain.master.protocol:remote}"
                host="172.16.81.100" port="${jboss.domain.master.port:9999}"
            <static-discovery name="backup"  protocol="${       (2)
                {jboss.domain.master.protocol:remote}" host="172.16.81.101" port="$
                {jboss.domain.master.port:9999}"/> </discovery-options> </remote>

</domain-controller>
```

1. **This points to a regular domain controller IP address.**
2. **This is a slave host controller that is supposed to be started with the --backup option.**

The slave host controller using the above example configuration will attempt to connect to each domain controller in the order they appear, registering with the first to respond. The names primary and backup have no special meaning.

If the primary domain controller is not available, the backup domain controller must be manually stopped and reconfigured to be a domain controller. If you started with the --backup option last time, you should already have a current and complete domain.xml configuration file, and can take over the domain controller tasks.

Promoting a slave host controller to the master role is still a manual process for EAP 7.0.0. The above sample configuration just shows a way to NOT require changing the configuration of the slave host controller between each other to point to the new master.

Authentication with a domain controller

A host controller connected to a domain controller has access to all settings used by all profiles. However, an external host controller can connect to a managed domain and obtain these settings from the domain controller if authentication is not required. To avoid this scenario, EAP requires a host controller to authenticate the domain controller to be part of the managed domain. This authentication requires:

- ***An administration user in the managed domain:*** **created during the process of installation or later, by running the JBOSS_HOME/bin/add-user.sh script on the domain controller.**
- ***An XML element with an encrypted password in the host controller configuration file –*** **Provided by the JBOSS_HOME/bin/add-user.sh script, it will be updated in each host controller configuration file (host-slave.xml) . The XML element is named secret and the value attribute is an encrypted password obtained from the add user.sh script.**
- ***A username in the host controller's configuration file –*** **Added using the JBOSS_HOME/bin/add-user.sh in the host controller, it must be added to the remote element in the host controller's configuration file (host -slave.xml).**

In the host-slave.xml file of the host controller, the following XML contains the encrypted password:

```
<secret value="c2xhdmVfdXNlcl9wYXNzd29yZA==" />
```

Also, in the host-slave.xml file, you must add the username to the domain controller by changing the following XML extract:

```
<remote username="jbossadm" security-realm="ManagementRealm">
```