Advanced Red Hat JBoss Enterprise Application Platform 7

RED HAT® DEVELOPER PROGRAM

Table of Contents

Red Hat JBoss Enterprise Application Platform 7 (JBoss EAP) is a Java EE 7-certified application platform. It is built on a collection of open source technologies, including an embeddable web server, messaging, clustering and high availability, and caching. It can be a single standalone server or have multiple defined domains, with a master server for centralized management, profiles to define configuration, and hosted servers for scale. Using the command-line tool

Basic operations syntax

The basic structure of an operation is:
[resourceType=name]:operation(parameter=value)

- The resource type is the domain location (in a domain) and the subsystem, meaning the profile, server, or host; this is not used in standalone
  - Set explicitly to run on any resource
  - Skip it to run it on the current resource
  - Use a slash to run on the root resource
  - Use a period-slash (./) to run on a child resource
  - **NOTE: Most of the examples here are for a standalone server, unless noted otherwise**
- Operation (command) name
- Any parameters to update or to be included to run the operation

Basic Management

`./bin/jboss-cli.sh --connect`	Connect to the CLI.
`./bin/standalone.sh`	Start a standalone EAP server.
`./bin/domain.sh`	Start an EAP domain controller.
`Ctrl + C in the terminal`	Stop an EAP standalone or domain controller interactively.

`shutdown`	Stop an EAP standalone or domain controller running in the background.
`/host=master:start`	Start a server in a domain.
`/host=master:stop`	Stop a server in a domain.
`:suspend(timeout=60)`	Suspend a standalone server.
`:suspend-servers(timeout=60)`	Suspend all servers in a domain.
`/host=master/server-config=server-one:suspend(timeout=60)`	Suspend a single server in a managed domain.
`/server-group=main-server-group:suspend-servers(timeout=60)`	Suspend all servers in a server group.
`:resume`	Resume a server.
`./bin/add-user.sh`	Add a user interactively.
`./bin/add-user.sh -u 'mgmtuser1' -p \\ 'password1!' -g 'guest,mgmtgroup'`	Add a user with parameters for the username (-u), password (-p), and user groups (-g).
`patch apply /path/to/downloaded-patch.zip` `shutdown --restart=true`	Patch a server.

`patch apply /path/to/downloaded-patch.zip --host=Host`
 	
`shutdown --restart=true`
	Patch a host in a managed domain.

`patch rollback --patch-id=PATCH_ID --reset-configuration=TRUE`
 	
`shutdown --restart=true`
	Roll back a patch.

Deploying and managing applications

`deploy /path/to/app/application.war`	Deploy an application to a standalone server.
`deploy --server-groups=main-server-group /path/to/app/application.war`	Deploy an application to a domain host.

Socket bindings

A socket binding associates an EAP interface with a defined port. Since there are a variety of different ports used by a single EAP server, that collection of socket bindings is a socket binding group. For managed servers -- where multiple servers may be on the same host or using the same configuration -- port offsets are used to ensure unique port assignments for each server while still using the same socket binding group configuration; the offset adds a defined number to each port number in the socket binding group and uses the new port numbers for the managed server.

`/socket-binding-group=new-sockets:add(default-interface=public)`	Create a socket binding group.
`/socket-binding-group=new-sockets/socket-binding=new-socket-binding:add(port=1234)`	Add a socket binding (port) to the group.
`/socket-binding-group=new-sockets/socket-binding=new-socket-binding:write-attribute(name=interface,value=unsecure)`	Change the interface for the socket binding to something other than the default for the socket binding group.
`/host=master/server-config=server-two/:write-attribute(name=socket-binding-port-offset,value=250)`	Set the port offset for a server within a host. The offset allows the server to inherit the socket binding group settings and then to offset the port values so there are no conflicts with other servers on the host. **This is only set for a managed server.**

Logging

`GET http://localhost:9990/management/subsystem/logging/log-file/server.log?operation=attribute&name=stream&useStreamAsResponse`	Read the server.log using REST.
`/subsystem=logging/log-file=server.log:read-log-file`	Read the server.log using the CLI.
`/subsystem=logging/root-logger=ROOT:write-attribute(name=level,value=LEVEL)`	Set the log level.

`/subsystem=logging/periodic-rotating-file-handler=PERIODIC_HANDLER_NAME:add(file={path=FILE_PATH,relative-to=RELATIVE_TO_PATH},suffix=SUFFIX)`
 	
`/subsystem=logging/periodic-rotating-file-handler=PERIODIC_HANDLER_NAME:write-attribute(name=level,value=LEVEL)`
 	
`/subsystem=logging/logger=CATEGORY:add-handler(name=PERIODIC_HANDLER_NAME)`
	Add a log handler, set its attributes, and assign it to a logger. There are several different kinds of log handlers; this example uses a rotating file handler.

`/subsystem=logging/pattern-formatter=PATTERN_FORMATTER_NAME:add(pattern=PATTERN_STRING)`	Enable the pattern formatting for the log file.
`/subsystem=logging/pattern-formatter=PATTERN_FORMATTER_NAME:write-attribute(name=color-map,value="LEVEL:COLOR,LEVEL:COLOR")`	Change the pattern format.
`/core-service=management/access=audit/logger=audit-log:write-attribute(name=enabled,value=true)`	Configure audit logging for a standalone server.
`/host=HOST_NAME/core-service=management/access=audit/logger=audit-log:write-attribute(name=enabled,value=true)`	Configure audit logging for a domain controller.

`/subsystem=logging/logger=com.arjuna:write-attribute(name=level,value=VALUE)`	Enable transaction logging.

JVM settings

The default JVM settings are defined in the standalone.conf or domain.conf file for the app server. These settings can be configured directly in the configuration file or using the command-line tools.

`JAVA_OPTS="-Xms64m -Xmx512m -XX:MaxMetaspaceSize=256m -Djava.net.preferIPv4Stack=true"JAVA_OPTS="$JAVA_OPTS -Djboss.modules.system.pkgs=$JBOSS_MODULES_SYSTEM_PKGS -Djava.awt.headless=true"`	Set the default heap settings in the configuration file.
`/host=HOST_NAME/jvm=production_jvm:add( heap-size=2048m, max-heap-size=2048m, max-permgen-size=512m, stack-size=1024k, jvm-options=["-XX:-UseParallelGC"] )`	Set the default heap settings for a domain host.
`/host=HOST_NAME/jvm=default:write-attribute(name=heap-size,value="1024m")`	Set the heap size for a host and all associated servers.
`/server-group=groupName/jvm=default:write-attribute(name=heap-size,value="1024m")`	Set the heap size for a server group.
`/host=HOST_NAME/server-config=server-one/jvm=default:write-attribute(name=heap-size,value="1024m")`	Set the heap size for a single server.
`/host=HOST_NAME/server-config=server-one/jvm=production_jvm:add()`	Set a server to use the default JVM settings for the domain.
`/host=HOST_NAME/jvm=default:add-jvm-option(jvm-option="-Djava.net.preferIPv4Stack=false") /host=HOST_NAME/jvm=default:add-jvm-option(jvm-option="-Djava.net.preferIPv6Addresses=true") /host=HOST_NAME/interface=management:write-attribute(name=inet-address,value="${jboss.bind.address.management:[::1]}")`	Enable IPv6 and update the interfaces to use IPv6. This changes the default settings for a domain host.
`/host=HOST_NAME/jvm=default:add-jvm-option(jvm-option="-d64")`	Enable 64-bit settings. This changes the default settings for a domain host.

Security and role-based access control

`<jboss-cli xmlns="urn:jboss:cli:2.0">`
`  <default-protocol use-legacy-override="true">remote+https</default-protocol>`
`  <!-- The default controller to connect to when 'connect' command is executed w/o arguments -->`
`  <default-controller>`
`  <protocol>remote+https</protocol>`
`  <host>localhost</host>`
`  <port>9993</port>`
`  </default-controller>`
	Configure the JBoss CLI to use SSL. This is set in the jboss-cli.xml configuration file. Set these properties:
	• default-protocol to remote+https
	• protocol to remote+https
	• port to 9993

`/core-service=management/security-realm=ManagementRealmHTTPS:add`	Create a security realm.
`/core-service=management/security-realm=ManagementRealmHTTPS/authentication=properties:add(path=https-mgmt-users.properties,relative-to=jboss.server.config.dir)`	Set authentication properties for a security realm.
`/core-service=management/management-interface=http-interface:write-attribute(name=security-realm,value=ManagementRealmHTTPS)`	Set which security realm to use.
`/core-service=management/security-realm=ManagementRealmHTTPS/server-identity=ssl:add(keystore-path=identity.jks,keystore-relative-to=jboss.server.config.dir,keystore-password=password1,alias=appserver)`	Add a keystore to the realm configuration.
`/core-service=management/access=authorization:write-attribute(name=provider, value=rbac)`	Enable role-based access control.
`/core-service=management/access=authorization:write-attribute(name=permission-combination-policy, value=POLICYNAME)`	Set the permission policy for authorization.

`/core-service=management/access=authorization:read-children-names(child-type=role-mapping)`
`{`
`  "outcome" => "success",`
`  "result" => [`
`  "Administrator",`
`  "Deployer",`
`  "Maintainer",`
`  "Monitor",`
`  "Operator",`
`  "SuperUser"`
`  ]`
`}`
	List configured roles.

`/core-service=management/access=authorization/role-mapping=ROLENAME:add`	Create a role.
`/core-service=management/access=authorization/role-mapping=ROLENAME/include=ALIAS:add(name=GROUPNAME,type=GROUP)`	Add a group to a role.

`/core-service=management/access=authorization/role-mapping=ROLENAME:read-resource(recursive=true)`
`{`
`  "outcome" => "success",`
`  "result" => {`
`  "include-all" => false,`
`  "exclude" => undefined,`
`  "include" => {`
`  "user-theboss" => {`
`  "name" => "theboss",`
`  "realm" => undefined,`
`  "type" => "USER"`
`  },`
`  "user-harold" => {`
`  "name" => "harold",`
`  "realm" => undefined,`
`  "type" => "USER"`
`  },`
`  "group-SysOps" => {`
`  "name" => "SysOps",`
`  "realm" => undefined,`
`  "type" => "GROUP"`
`  }`
`  }`
`  }`
`}`
	Show the role mappings for a given role.

Class loading

Classes are defined through modules. The classes available to deployments are loaded by explicitly defining what modules and dependencies are required. There are two types of modules: static and dynamic. Static modules are in the modules/ directory in the app server. All APIs, including the Java EE APIs, are included as static modules. Dynamic modules are loaded by the app server for each JAR or WAR deployment or EAR subdeployment. These dynamic modules can define dependencies on other modules and can themselves be defined as dependencies.

`module add --name=MODULE_NAME --resources=PATH_TO_RESOURCE --dependencies=DEPENDENCIES`	Create a custom class module.
`/subsystem=ee:write-attribute(name=global-modules,value=[{name=MODULE_NAME_1},{name=MODULE_NAME_2}]`	Define a list of global modules to include as dependencies for all deployments.

Transactions

`/subsystem=transactions/:write-attribute(name=statistics-enabled,value=true)`	Enable statistics.
`/subsystem=transactions/:write-attribute(name=default-timeout,value=300)`	Set default timeout.

`/subsystem=logging/logger=com.arjuna:write-attribute(name=level,value=VALUE)`	Set the transaction log level for a standalone server.
`/profile=default/subsystem=logging/logger=com.arjuna:write-attribute(name=level,value=VALUE)`	Set the transaction log level for a managed domain.
`/profile=default/subsystem=transactions/log-store=log-store/:probe`	Refresh the transactions log. For a standalone server, remove the /profile=default argument.
`/subsystem=transactions/log-store=log-store:read-children-names(child-type=transactions)`	View the transactions log for a standalone server.
`/host=master/server=server-one/subsystem=transactions/log-store=log-store:read-children-names(child-type=transactions)`	View the transactions log for a server within a managed domain.
`/profile=default/subsystem=transactions/log-store=log-store/transactions=0\:ffff7f000001\:-b66efc2\:4f9e6f8f\:9:read-resource`	View the attributes for a specific transaction.
`/host=master/server=server-one/subsystem=transactions/log-store=log-store:read-children-names(child-type=transactions)`	View all prepared transactions.
`/profile=default/subsystem=transactions/log-store=log-store/transactions=0\:ffff7f000001\:-b66efc2\:4f9e6f8f\:9:delete`	Delete a transaction.
`/profile=default/subsystem=transactions/log-store=log-store/transactions=0\:ffff7f000001\:-b66efc2\:4f9e6f8f\:9/participants=2:recover`	Recover a transaction.

`/subsystem=datasources/data-source=DATASOURCE_NAME:write-attribute(name=jta,value=true)`
 	
`reload`
	Set a non-XA datasource to use JTA.

Datasources

`module add --name=MODULE_NAME --resources=PATH_TO_JDBC_JAR --dependencies=DEPENDENCIES`
 	
`connect`
 	
`/profile=PROFILE_NAME /subsystem=datasources/jdbc-driver=DRIVER_NAME:add(driver-name=DRIVER_NAME,driver-module-name=MODULE_NAME,driver-xa-datasource-class-name=XA_DATASOURCE_CLASS_NAME, driver-class-name=DRIVER_CLASS_NAME)`
	Add and register a new module. For a managed domain, also specify the profile to which to register the module.

`data-source add --name=DATASOURCE_NAME --jndi-name=JNDI_NAME --driver-name=DRIVER_NAME --connection-url=CONNECTION_URL --profile=PROFILE_NAME`	Add a new non-XA datasource. For a managed domain, also specify the profile with which to use the datasource.
`xa-data-source add --name=XA_DATASOURCE_NAME --jndi-name=JNDI_NAME --driver-name=DRIVER_NAME --xa-datasource-class=XA_DATASOURCE_CLASS --xa-datasource-properties={"ServerNam e"=>"HOSTNAME","DatabaseName"=>"DATABA SE_NAME"}`	Add a new XA datasource.
`/subsystem=datasources/xa-data-source=XA_DATASOURCE_NAME/xa-datasource-properties=ServerName:add(value=HOSTNA ME)`	Change a datasource property. This can be done for both XA and non-XA datasources; this example is for setting the server name for an XA datasource.
`/host=HOSTNAME/server=SERVER_NAME/ subsystem=datasources/data-source=DATASOURCE_NAME:test-connection-in-pool`	Test the connection to a datasource. For a managed domain, specify the host and server name (as shown).
`/subsystem=datasources/data-source=ExampleDS/statistics=pool:write-attribute(name=statistics-enabled,value=true)`	Enable statistics for pools. This is shown for a standalone server; for a managed domain, specify the host and server.
`/subsystem=datasources/data-source=ExampleDS/statistics=jdbc:write-attribute(name=statistics-enabled,value=true)`	Enable statistics for JDBC runtime. This is shown for a standalone server; for a managed domain, specify the host and server.
`/subsystem=datasources/data-source=ExampleDS/statistics=pool:read-resource(include-runtime=true)`	View the pool statistics. This is shown for a standalone server; for a managed domain, specify the host and server.

```
{
  "outcome" => "success",
  "result" => {
    "ActiveCount" => 1,
    "AvailableCount" => 20,
    "AverageBlockingTime" => 0L,
    "AverageCreationTime" => 122L,
    "AverageGetTime" => 128L,
    "AveragePoolTime" => 0L,
    "AverageUsageTime" => 0L,
    "BlockingFailureCount" => 0,
    "CreatedCount" => 1,
    "DestroyedCount" => 0,
    "IdleCount" => 1,
    ...
}
```

About the author

DEON BALLARD is a middleware marketing writer for Red Hat and writes for and edits the Middleware Blog. She has previously worked as a journalist, designer, and as a technical writer for identity management and security, cloud and virtualization, system management, and middleware software

@JBoss
http://middlewareblog.redhat.com