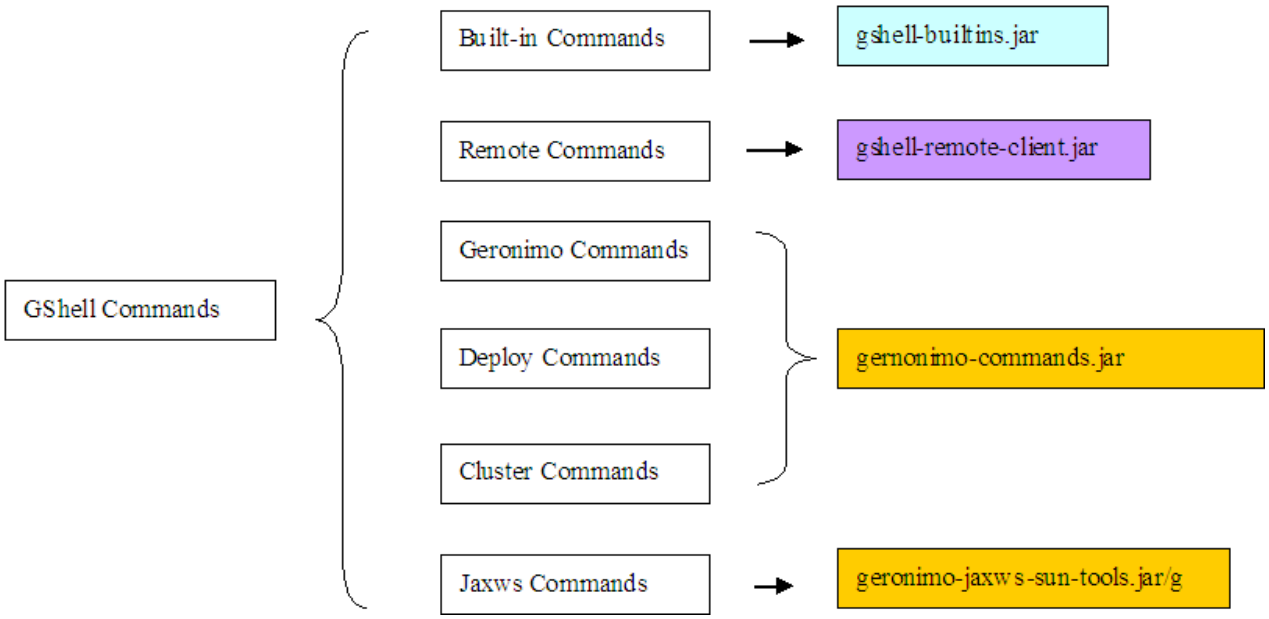
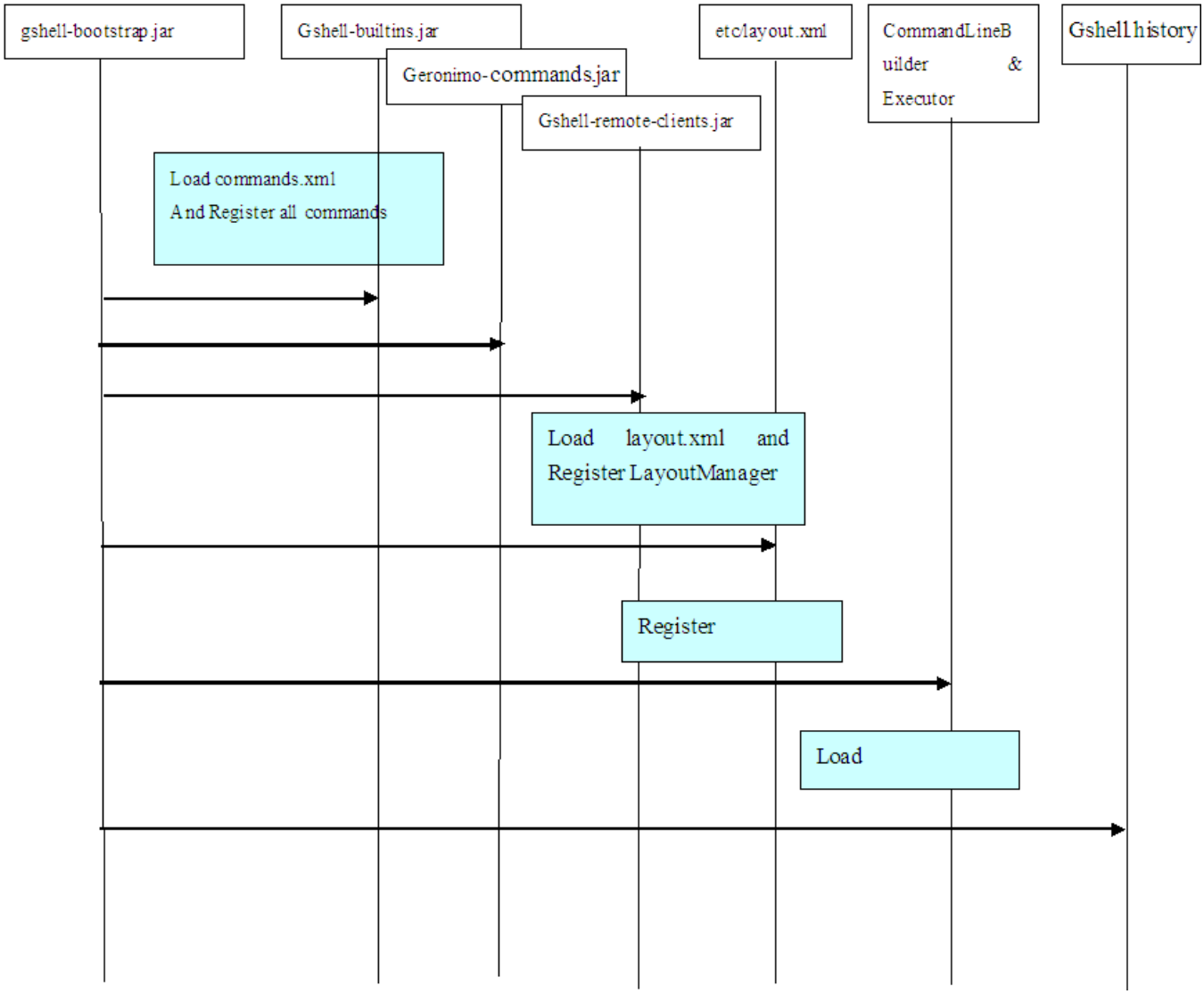
How GShell works with Geronimo kernel

GShell Commands Structure



Build-in commands like set, clean, help, quit and so on are defined in gshell-builtin project
Geronimo commands like start-server,stop-server adn deploy commands like start-module are defined in geronimo-commands project.
Jaxws commands are defined in geronimo-jaxws-builder project.
Remote commands are defined in gshell-remote-client project.

GShell Startup progress



1 org.codehaus.plexus.ClassLoader load every jar defined in etc/gsh-classworlds.conf
2 org.apache.geronimo.gshell.plugin.CommandDiscoverer load the commands.xml in gshell-builtin.jar/META-INF/gshell
3 All the commands register in org.apache.geronimo.gshell.registry..CommandRegistry.
4 The same process for geronimo-commands and jaws-commands
5 org.apache.geronimo.gshell.layout.loader.XMLLayoutLoader load the layout.xml in etc/layout.xml
6 GShell start up DefaultLayoutManager.
7 GShell start up org.apache.geronimo.gshell.DefaultCommandLineBuilder and DefaultCommandExecutor
8 Gshell read argument from console and read history from Document and Setting/[username]/.gshell/gshell.history
9 Finish initialization and waiting

Note:

Command.xml shows reflection between command id and command implementation

Layout.xml shows reflection between command line string and command id

GShell process command line progress

1 org.apache.geronimo.gshell.console.JLineConsole get a command line from console
2 org.apache.geronimo.gshell.DefaultCommandLineBuilder parse the command line and check if it has a syntax error
3 org.apache.geronimo.gshell.layout.DefaultLayoutManager find corresponding command id and give to org.apache.geronimo.gshell.DefaultCommandExecutor .
4org.apache.geronimo.gshell.DefaultCommandExecutor call A certain command execute the doExecute() method.
5 org.apache.geronimo.gshell.clp.CommandLineProcessor process the arguements.

Define your own command

1 Define command class

In geronimo/framework/modules/geronimo-commands projects, you can define your own command class in src folder.
Note all the command classes are defined in groovy.
E.g.

```
package org.apache.geronimo.commands
import org.apache.geronimo.gshell.command.annotation.CommandComponent

/**
  * Test
  */
@CommandComponent(id='geronimo-commands:test', description="Test")
class TestCommand extends ConnectCommand {
    @Option(name='-a', aliases=['--argument'], description='Arguement test')
    int argu = 1;
    protected Object doExecute() throws Exception {
        print "Hello World!"
    }
}
```

Compile source codes and put the target jar file in the same directory in binary server file.

2 Modify configuration file

Modify file layout.xml in [server dir]/etc
E.g

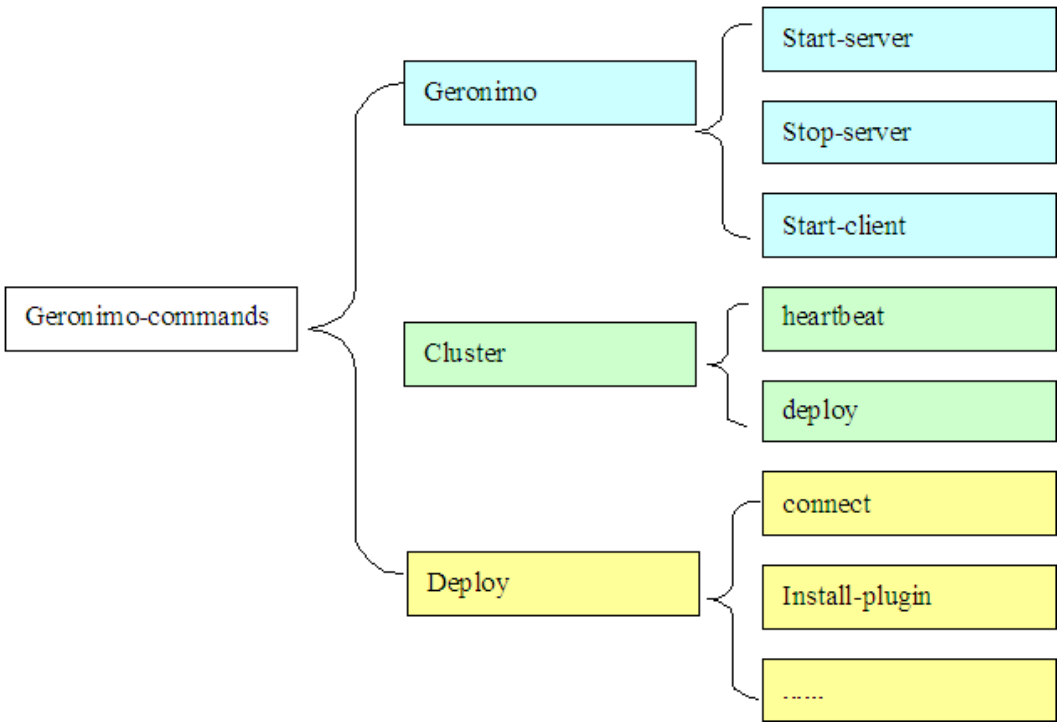
```
.......
<!-- Geronimo -->
        <group>
              <name>geronimo</name>
                  <nodes>
                      <command>
                          <name>stop-server</name>
                          <id>geronimo-commands:stop-server</id>
                      </command>
                      <command>
                          <name>test</name>
                          <id>geronimo-commands:test</id>
                      </command>
                  </nodes>
        </group>
.....
```

3 restart gshell

You can find test command in help list and use test to see print line.

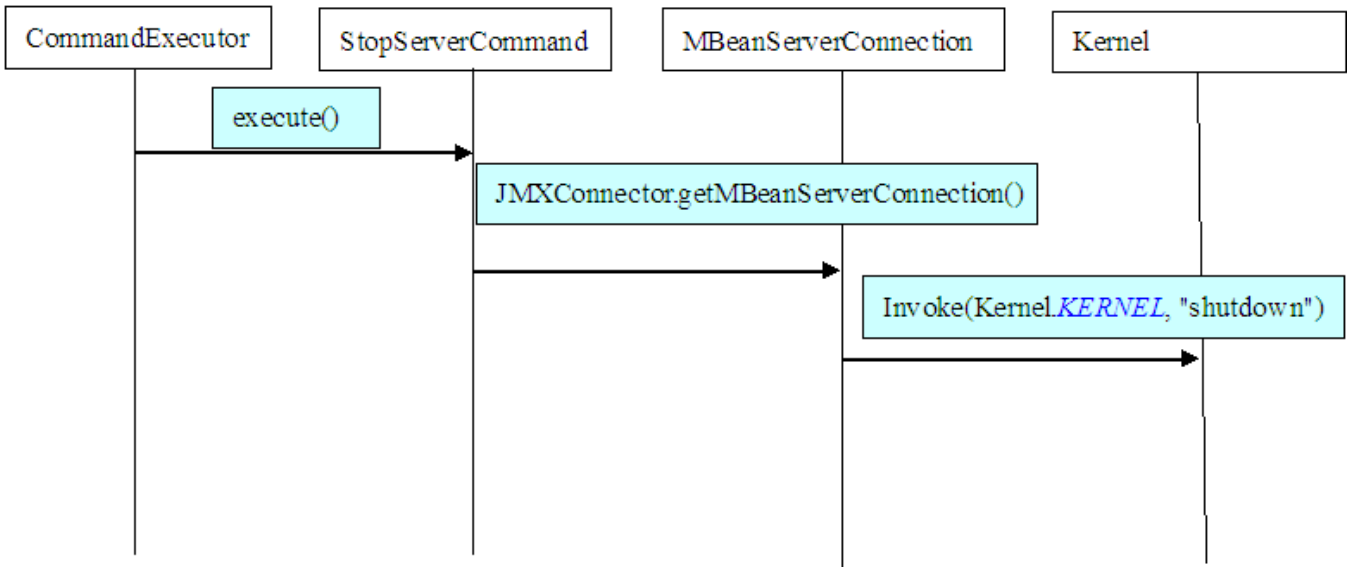
```
deploy

stop                    Stop a module
list-modules            List modules
list-plugins            Install plugins into a server
disconnect              Disconnect from a server
start                   Start a module
assemble                Create a custom server from the current one
deploy                  Deploy a module
list-targets            List targets
redeploy                Redeploy a module
install-plugin          Install a plugin
install-library         Install library
restart                 Restart a module
distribute              Distribute a module
undeploy                Undeploy a module
login                   Saves the username and password for this connection
connect                 Connect to a server


geronimo

wait-for-server         Wait for a server to start
start-server            Start a Geronimo server
start-client            Start an application client
test                    Test
stop-server             Stop a server


Harmony@ibm-eb710a950ca:/> geronimo/test
Hello World!Harmony@ibm-eb710a950ca:/>
```

Details in Geronimo-commands(Client)



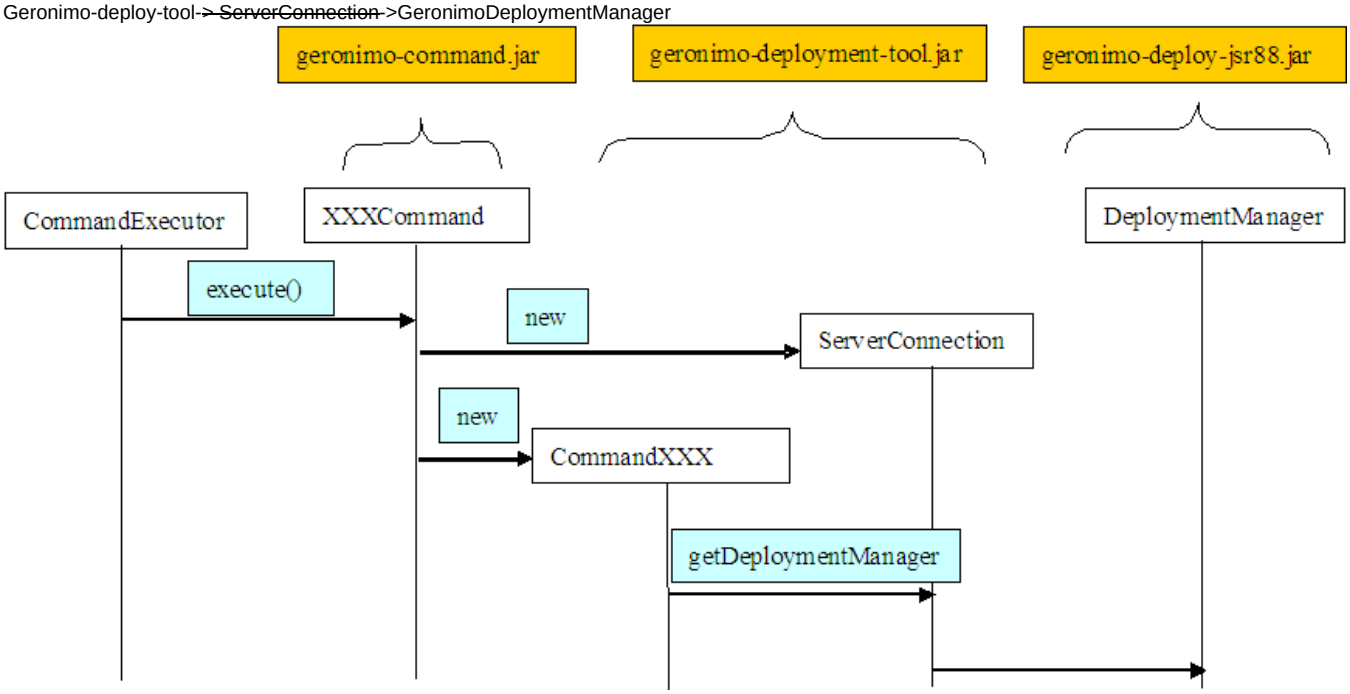
Geronimo command part:

Jmx-~~>kenel~~ >invoke(shutdown/getAttribute)
Basically, Geronimo stop and wait for server commands connect to remote/local server via jmx connections and get a MBeanServerConnection.Through MBeanServerConnection command get Kernel as Mbean and call invoke() method to shutdown or getAttribute.



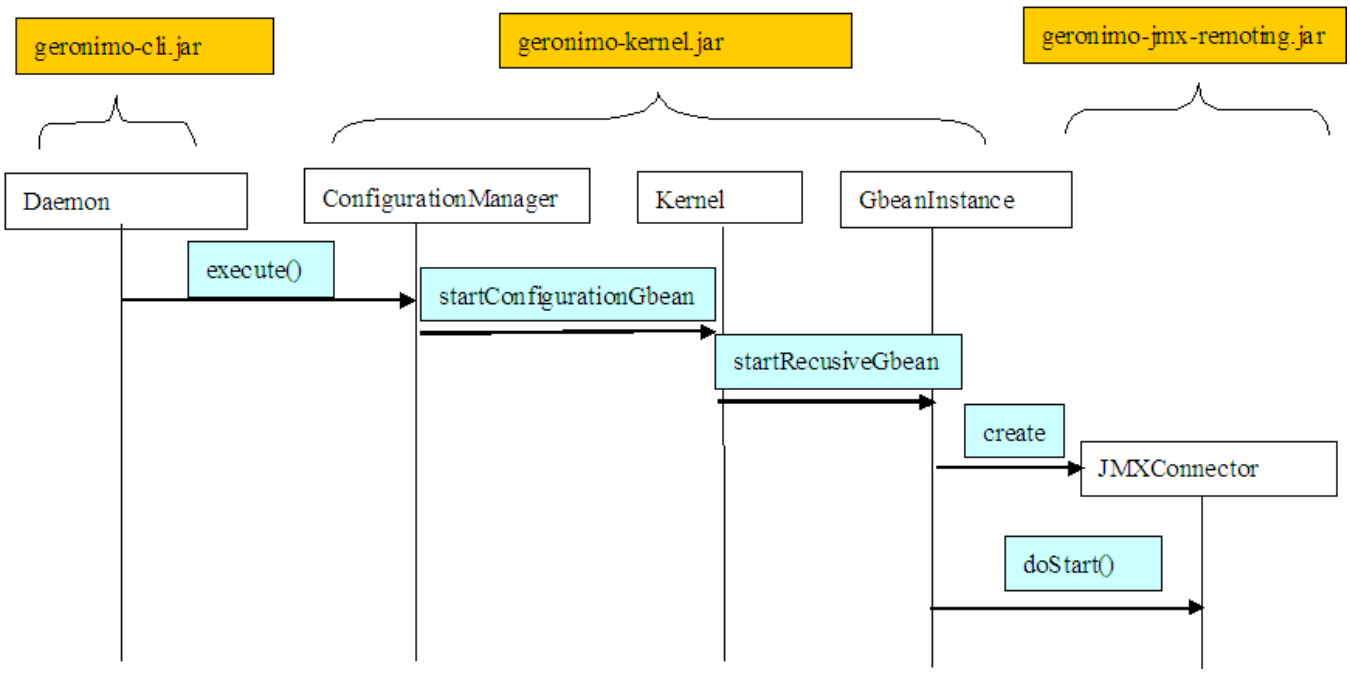
Deploy command part

Geronimo-deploy-tool-~~> ServerConnection~~ >GeronimoDeploymentManager



Details in geronimo-jmx-remoting(Server)

Build JMX Server progress



Key codes

in JMXConnector in geronimo-jmx-remoting project

```
public void doStart() throws Exception {
        jmxServiceURL = new JMXServiceURL(protocol, host, port, urlPath);
        Authenticator authenticator = null;
        Map env = new HashMap();
        if (applicationConfigName != null) {
            authenticator = new Authenticator(applicationConfigName, classLoader);
            env.put(JMXConnectorServer.AUTHENTICATOR, authenticator);
        } else {
            log.warn("Starting unauthenticating JMXConnector for " + jmxServiceURL);
        }        RMIServerSocketFactory serverSocketFactory = new GeronimoRMIServerSocketFactory(host);
        env.put(RMIConnectorServer.RMI_SERVER_SOCKET_FACTORY_ATTRIBUTE, serverSocketFactory);
        server = JMXConnectorServerFactory.newJMXConnectorServer(jmxServiceURL, env, mbeanServer);
        NotificationFilterSupport filter = new NotificationFilterSupport();
        filter.enableType(JMXConnectionNotification.OPENED);
        filter.enableType(JMXConnectionNotification.CLOSED);
        filter.enableType(JMXConnectionNotification.FAILED);
        server.addNotificationListener(authenticator, filter, null);
        server.start();
        log.debug("Started JMXConnector " + server.getAddress());
```