Java Email Server – Web Entry Bean
(JES-WEB)

version 1.1.x

License Information

This version of the documentation processed with LibreOffice 4.1.x, last edited 1. Jan. 2014

Java Runtime

JES-WEB 1.1.x compiles under Java SE 6, 7.

Dependencies

Runtime

commons-codec-1.4
commons-logging 1.1.1
derbyclient-10.10.1.1.jar *
jsf-api-2.1.1-b04
jsf-impl-2.1.1-b04
jsp-api-2.1
jstl-1.2
primefaces-2.2.1
servlet-api-2.5
redmond-1.0.1
south-street-1.0.1

All the aforementioned libraries are downloaded when the maven project compiles. There are no custom built versions. JES-WEB 1.1.x is thus fully portable.

* derbyclient-10.10.1.1.jar requires special handling, since it is not part of the jes war archive. A copy of this file can be found in the binary release of JES. Alternatively, it can be downloaded by clicking on http://db.apache.org/derby/releases/release-10.10.1.1.cgi and selecting either db-derby-10.10.1.1-bin.zip or db-derby-10.10.1.1-bin.tar.gz. Copy or extract the jar into an appropriate folder of the application server of choice (e.g. $CATALINA_BASE/lib for Tomcat).

Table of Contents

1. Important considerations

a. The JES server instance to be administered **must** be running JES version 2.8+. The domain/user/realm backend **must** be a database.

b. The augmented CBC facility is used to administer the database backend.

2. Introduction

Administering security sensitive resources that relate to many aspects of a Mail Handling Service such as JES poses a significant challenge. This challenge is two fold. On one hand, the need to retain the security integrity of a mail server requires that administrator access employs a number of security measures (authentication, channel security). On the other hand, a proliferation of ever technologically advancing mobile devices (smartphones, tablets), has made it so that access to networked resources is fulfilled with virtually no restriction with respect to locality.

Attempting to address this challenge, Java E-mail Server – Web Entry Bean (JES-WEB) has been developed. As the name implies, a web browser is to be used to access JES remotely. The JES Connection Based Configurator (CBC) is the employed communication layer protocol. The CBC is used to administer the server settings and to submit/update user, domain and DIGEST-MD5 realms. Data relating to users, domains and DIGEST-MD5 realms is retrieved by directly accessing the database backend.

The issue of security is split into two segments. The first segment concerns the connection between JES (or the database) and the web browser. Currently there are two security levels available. Either is selected in JES and copied over to JES-WEB. The first is virtually unrestricted access provided the CBC is contacted from the same (local) address as the one it is listening on. The second applies TLS/SSL to the communication channel and allows access to the CBC from any address. No authentication is carried out in either case. The second segment has the web browser at one end and the administrator's hardware of choice in any given time at the other. The latter can be anything from a standard desktop PC to a 3.2' smartphone. Access to the JES-WEB instance running on the web server requires the contacting party to authenticate itself. The authentication mechanism employed is SCRAM-SHA1. The JES administrator carries the burden of applying a security layer to the http connection. It goes without saying that communicating data in this segment, without a security layer, is anything but recommended.

Accessing a web site from devices such as smartphones presents another challenge for a developer. The more when the work of a mail server administrator is to be carried out. Considerable effort has been made to offer a responsive and comfortable web experience for small form factor devices.

3. Installation

JES-WEB is a standard J2EE 5 web application packaged inside a typical war archive file. It can thus be made to run from any compliant Servlet/JSP container. There exist however custom resource dependencies to facilitate access to the JES backend database and the CBC. At initial release time an auxiliary library is available that allows using either Tomcat (6, 7) or Glassfish (3.1) as the hosting web server.

The installation process is outlined as follows: The auxiliary library is added to a appropriate web server folder. The war archive is dragged/copied to the autodeploy directory of the web server. The server is then taken off line and the deployment descriptor is edited. The web server is then restarted and JES-WEB is made available to administer JES.

3.1 JES prerequisites

As already stated, JES must use a db as the backend and the CBC must be enabled. Consult the JES documentation on setting up JES in this manner. There are a number of parameters specified (explicitly or implicitly) as part of the JES installation that also affect JES-WEB. They are required by the custom resource dependencies and their values are to be supplied to the host web server via the appropriate deployment descriptors (context.xml in Tomcat, glassfish-resources.xml / domain.xml in Glassfish). They are listed in the box to the right.

Although the process as earlier outlined is common to both Tomcat and Glassfish, there are considerable differences that warrant detailing the installation process individually for these two web servers.

guiDbUsername: the username of the entity that has limited (read-only) access to the database backend.

guiDbPassword: the password of the afore-mentioned entity.

derbySchema: this is the handle that corresponds to the user that has full access to the database backend. Always in uppercase.

backend.secure: true/false indicating whether access to the db backend is TLS/SSL secured or not. This setting mirrors the backend/secure attribute in mail.xml.

db.url: the hostname address and port that the backing db listens on for requests. Mirrors the backend/db/host and backend/db/port attributes in mail.xml.

cbc.url: the hostname address and port that CBC listens on for requests. Mirrors the cbc/listenAddress and cbc/port attributes in mail.xml.

Table 1: Important deployment descriptor entries

3.2 Tomcat installation

JES-WEB is compatible with versions of Tomcat 6 and 7. The first step in the process is to copy JES-WebResources.jar to the directory $CATALINA_BASE/lib[1]. After that, JES-WEB.jar is to be copied to the directory $CATALINA_BASE/webapps. Tomcat is allowed to carry out its automatic deployment. When it is finished it is taken off line so that the context.xml deployment descriptor can be edited. JES-WEB is deployed to $CATALINA_BASE/webapps/JES-WEB. The context.xml file is located in folder META-INF under the previous mentioned deployment folder. There is a discrepancy between the two Tomcat version when it comes to which deployment descriptor to edit. In version 6, after a successful deployment, the JES-WEB.xml located in $CATALINA_BASE/conf/[enginename]/[hostname] is to be edited. In version 7, context.xml located in $CATALINA_BASE/webapps/JES-WEB/META-INF is the file to edit.

In keeping with the security outline as defined in section 2, context.xml is depicted in the next two code blocks, first when no security layer is defined (Deployment descriptor 1), then with a security layer in effect (Deployment descriptor 2). In both cases, all the username entries correspond to the *guiDbUsername* entry in Table 1. As is the case with the usernames, all the password entries correspond to the *guiDbPassword* entry in Table 1. The *derbySchema* and *backend.secure* entries need no further clarification. The url attribute of the *cbc/jeswebCC* Resource entry refers to the *cbc.url* entry in Table 1. Finally, the url attribute of either the *jdbc/jeswebDS* or the *jdbc/jeswebDSSSL* Resource reflects the *db.url* entry in Table 1.

1 $CATALINA_BASE denotes the directory that tomcat is installed into.

```
<?xml version="1.0" encoding="UTF-8"?>
<Context antiJARLocking="true" path="/JES-WEB">
 <Environment name="guiDbUsername" override="false" type="java.lang.String" value="username"/>
 <Environment name="guiDbPassword" override="false" type="java.lang.String" value="password"/>
 <Environment name="derbySchema" override="false" type="java.lang.String" value="BACKEND"/>
 <Environment name="backend.secure" override="false" type="java.lang.String" value="true"/>
 <Resource factory="com.xlat4cast.jesweb.TomcatConnectorControlFactory"
        name="cbc/jeswebCC" type="javax.net.SocketFactory" url="localhost:41001"/>
 <Resource defaultTransactionIsolation="READ_UNCOMMITTED"
           driverClassName="org.apache.derby.jdbc.ClientDriver"
           initialSize="2" name="jdbc/jeswebDS" type="javax.sql.DataSource"
           url="jdbc:derby://xlat4cast.com:1527/JES" username="username" password="password"/>
</Context>
```

Deployment descriptor 1: Tomcat context.xml with no security layer

3.2.1 Entries that relate to the TLS/SSL keystore and truststore

There are a number of entries in the Resources named *cbc/jeswebCCSSL* and *jdbc/jeswebDSSSL* that relate to the JSSE store for the (web) server private key/certificate and the store for the trusted Certificate Authorities certificates. These entries should match in both Resources. It is left to the administrator's discretion to select whether to have a common certificate store pair with the Tomcat installation or to keep them separate.

Detailed instructions on supplying a key pair (public/private key) to Tomcat are available in the Tomcat docs (located locally at $CATALINA_BASE/webapps/docs/ssl-howto.html). The instructions apply to the case of serving secure content to incoming http connections, but can be educational on the issue of handling X509 certificates in general in java.

A thorough walk through of the entire process that leads to establishing secure connections in both segments (outside world → JES-WEB, JES-WEB → JES) is supplied in instructions.txt in the openSSL directory under the root JES install directory.

Next follows a complete example using exclusively the java keytool. It should be noted that in Java SE6, keytool offers only the option to self-sign certificates, unlike in SE7. All the generated key pairs are self-signed, JES-WEB and Tomcat share the same stores as well as the same key pair (JES-WEB to use it as the client when contacting JES, Tomcat when serving secure content) and the default Tomcat alias is used (tomcat). The localhost is presumed to host JES-WEB and the CBC. This example also covers JES itself (the JES keystore and truststore are consider not to exist at the start of the process). Using the default Tomcat password is to be avoided at all costs in a production run.

Switch to the JES security folder. Generate the JES server key pair:
keytool -genkeypair -alias JESServer -keyalg RSA -keysize 2048 -validity 60 -keystore keystore.jks

Use the word “changeit” (unquoted) as the password. Enter localhost as the first and last name, use sensible choices for the other entries.

Export the certificate:
keytool -exportcert -alias JESServer -rfc -file JESServer.cer -keystore keystore.jks
Copy JESServer.cer to the root Tomcat folder.

Switch to the root Tomcat folder. Import the certificate into the Tomcat truststore:
keytool -importcert -alias JESServer -file JESServer.cer -trustcacerts -keystore truststoreCBC.jks

Generate the CBC client / Tomcat server key pair:
keytool -genkeypair -alias tomcat -keyalg RSA -keysize 2048 -validity 60 -keystore keystoreCBC.jks

Use the word “changeit” (unquoted) as the password. Enter localhost as the first and last name, use sensible choices for the other entries.

Export the certificate:
keytool -exportcert -alias tomcat -rfc -file tomcat.cer -keystore keystoreCBC.jks
Copy tomcat.cer to the JES security folder.

Switch to the JES security folder. Import the certificate into the JES truststore:
keytool -importcert -alias tomcat -file tomcat.cer -trustcacerts -keystore truststore.jks

Edit the JES mail.xml entries that relate to the certificate stores. Restart JES.

context.xml will have the following entries:

```
<?xml version="1.0" encoding="UTF-8"?>
<Context antiJARLocking="true" path="/JES-WEB">
  <Environment name="guiDbUsername" override="false" type="java.lang.String" value="username"/>
  <Environment name="guiDbPassword" override="false" type="java.lang.String" value="password"/>
  <Environment name="derbySchema" override="false" type="java.lang.String" value="BACKEND"/>
  <Environment name="backend.secure" override="false" type="java.lang.String" value="true"/>
  <Resource auth="Application" factory="com.xlat4cast.jesweb.TomcatConnectorControlFactory"
        name="cbc/jeswebCC" type="javax.net.SocketFactory" url="localhost:41001"/>
  <Resource factory="com.xlat4cast.jesweb.TomcatConnectorControlFactory"
        confDir="${catalina.base}" keystorePassword="changeit"
        type="javax.net.ssl.SSLSocketFactory" name="cbc/jeswebCCSSL"/>
  <Resource connectionProperties="ssl=peerAuthentication;"
            defaultTransactionIsolation="READ_UNCOMMITTED"
            driverClassName="org.apache.derby.jdbc.ClientDriver"
            factory="com.xlat4cast.jesweb.TomcatDbControlFactory"
            initialSize="2" name="jdbc/jeswebDSSSL" type="javax.sql.DataSource"
            confDir="${catalina.base}" keystorePassword="changeit"
            url="jdbc:derby://localhost:1527/JES" username="username" password="password"/>
</Context>
```

Notice the absence of most of the keystore/trustostore related entries. Default Java values are used throughout. Furthermore, the confDir attribute points to the folder where both stores are to be found. The default store names, keystoreCBC.jks and truststoreCBC.jks, allows skipping setting the store locations altogether. No default keystore password exists for JES-WEB. The confDir, keystoreLocation and truststoreLocation attributes allow the use of ${catalina.base} as a shortcut to the actual Tomcat root directory.

Finally, server.xml will have the following Connector element defined:

```
...
<Connector port="8443" SSLEnabled="true"
           maxThreads="150" scheme="https" secure="true"
           clientAuth="false" sslProtocol="TLS" keystoreFile="${catalina.base}/keystoreCBC.jks"
             protocol="org.apache.coyote.http11.Http11Protocol"/>
...
```

Having used the default Tomcat alias and password entries renders the use of the relative attributes unnecessary. After restarting Tomcat, JES-WEB can securely connect to the CBC server and Tomcat can serve JES-WEB pages over a secure connection.

```
<?xml version="1.0" encoding="UTF-8"?>
<Context antiJARLocking="true" path="/JES-WEB">
 <Environment name="guiDbUsername" override="false" type="java.lang.String" value="username"/>
 <Environment name="guiDbPassword" override="false" type="java.lang.String" value="password"/>
 <Environment name="derbySchema" override="false" type="java.lang.String" value="BACKEND"/>
 <Environment name="backend.secure" override="false" type="java.lang.String" value="true"/>
 <Resource auth="Application" factory="com.xlat4cast.jesweb.TomcatConnectorControlFactory"
       name="cbc/jeswebCC" type="javax.net.SocketFactory" url="localhost:41001"/>
 <Resource factory="com.xlat4cast.jesweb.TomcatConnectorControlFactory"
       confDir=""
       keystoreLocation="${catalina.base}/keystoreCBC.jks" keystorePassword="password"
       keystoreProvider="SunJCE" keystoreType="JCEKS"
       truststoreLocation="${catalina.base}/truststoreCBC.jks" truststorePassword="password"
       truststoreProvider="SUN" truststoreType="JKS"
       type="javax.net.ssl.SSLSocketFactory" name="cbc/jeswebCCSSL"/>
 <Resource connectionProperties="ssl=peerAuthentication;"
           defaultTransactionIsolation="READ_UNCOMMITTED"
           driverClassName="org.apache.derby.jdbc.ClientDriver"
           factory="com.xlat4cast.jesweb.TomcatDbControlFactory"
           initialSize="2" name="jdbc/jeswebDSSSL" type="javax.sql.DataSource"
           confDir=""
           keystoreLocation="${catalina.base}/keystoreCBC.jks" keystorePassword="password"
           keystoreProvider="SunJCE" keystoreType="JCEKS"
           truststoreLocation="${catalina.base}/truststoreCBC.jks" truststorePassword="password"
           truststoreProvider="SUN" truststoreType="JKS"
           url="jdbc:derby://xlat4cast.com:1527/JES" username="username" password="password"/>
</Context>
```

Deployment descriptor 2: Tomcat context.xml with security layer

3.3 Glassfish installation

JES-WEB has been tested with Glassfish OSE version 3.1.2. It is believed that previous 3.x Glassfish releases will present no problem in hosting JES-WEB. The first step in the process is to copy JES-WebResources.jar to the directory $domainDir/lib[2]. After that, JES-WEB.jar is to be copied to the directory $domainDir/autodeploy.

2 $domainDir denotes the directory where the Glassfish domain to host JES-WEB is located. This could be for example /usr/local/glassfish3/glassfish/domains/domain1

Glassfish is allowed to carry out its automatic deployment. When it is finished it is taken off line so that domain.xml can be edited. JES-WEB is deployed to $domainDir/applications/JES-WEB. The glassfish-resources.xml file is located in folder WEB-INF under the previous mentioned deployment folder. The entries located in glassfish-resources.xml are copied during the deployment to domain.xml under $domainDir/config. All changes are affected in the domain.xml file.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE resources PUBLIC "-//GlassFish.org//DTD GlassFish Application Server 3.1 Resource Definitions//EN"
"http://glassfish.org/dtds/glassfish-resources_1_5.dtd">
<resources>
  <custom-resource res-type="java.util.Properties" jndi-name="jes-web/env-properties"
          factory-class="org.glassfish.resources.custom.factory.PropertiesFactory">
    <property name="guiDbUsername" value="username" />
    <property name="guiDbPassword" value="password" />
    <property name="derbySchema" value="BACKEND" />
    <property name="backend.secure" value="true" />
  </custom-resource>
  <jdbc-connection-pool res-type="javax.sql.DataSource"
              transaction-isolation-level="read-uncommitted"
              max-pool-size="8" steady-pool-size="2" name="jesPool"
              datasource-classname="org.apache.derby.jdbc.ClientConnectionPoolDataSource">
    <property name="DatabaseName" value="JES"></property>
    <property name="ServerName" value="xlat4cast"></property>
    <property name="PortNumber" value="1527"></property>
    <property name="User" value="username"/>
    <property name="Password" value="password"/>
  </jdbc-connection-pool>
  <jdbc-resource pool-name="jesPool" jndi-name="jdbc/jeswebDS"></jdbc-resource>
  <custom-resource res-type="javax.net.SocketFactory" jndi-name="cbc/jeswebCC"
          factory-class="com.xlat4cast.jesweb.GlassfishConnectorControlFactory">
    <property name="url" value="localhost:41001" />
  </custom-resource>
</resources>
```

Deployment descriptor 3: Glassfish glassfish-resources.xml with no security layer

Like in the case of Tomcat and in keeping with the security outline as defined in section 2, glassfish-resources.xml is depicted in two code blocks, first when no security layer is defined (Deployment descriptor 3, above), then with a security layer in effect (Deployment descriptor 4, next page). In both cases, all the username entries correspond to the *guiDbUsername* entry in Table 1. Similarly, all the password entries correspond to the *guiDbPassword* entry in Table 1. The *derbySchema* and *backend.secure* entries need no further clarification. The url property of the *cbc/jeswebCC* Resource entry refers to the *cbc.url* entry in Table 1. Finally, the ServerName and PortNumber properties of either the *jesPool* or the *jesPoolSSL jdbc-connection-pool* resource reflects the *db.url* entry in Table 1.

3.3.1 Entries that relate to the TLS/SSL keystore and truststore

Using a secure layer in Glassfish bears a notable difference compared to Tomcat. It arises from the fact that the connection pool used to retrieve connections to the JES db is not a custom resource as in Tomcat but a

standard Glassfish jdbc resource. The implication is that there is no option for JES-WEB to use distinct certificate stores but is bound to the ones specified explicitly for Glassfish. They are instantiated by Glassfish solely (at least to the developer's knowledge) using the “javax.net.ssl.*” system properties.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE resources PUBLIC "-//GlassFish.org//DTD GlassFish Application Server 3.1 Resource Definitions//EN"
"http://glassfish.org/dtds/glassfish-resources_1_5.dtd">
<resources>
  <custom-resource res-type="java.util.Properties" jndi-name="jes-web/env-properties"
            factory-class="org.glassfish.resources.custom.factory.PropertiesFactory">
    <property name="guiDbUsername" value="username" />
    <property name="guiDbPassword" value="password" />
    <property name="derbySchema" value="backend" />
    <property name="backend.secure" value="true" />
  </custom-resource>
  <jdbc-connection-pool res-type="javax.sql.DataSource"
                transaction-isolation-level="read-uncommitted"
                max-pool-size="8" steady-pool-size="2" name="jesPoolSSL"
                datasource-classname="org.apache.derby.jdbc.ClientConnectionPoolDataSource">
    <property name="DatabaseName" value="JES"></property>
    <property name="ServerName" value="xlat4cast"></property>
    <property name="PortNumber" value="1527"></property>
    <property name="User" value="username"/>
    <property name="Password" value="password"/>
    <property name="Ssl" value="peerAuthentication"/>
  </jdbc-connection-pool>
  <jdbc-resource pool-name="jesPoolSSL" jndi-name="jdbc/jeswebDSSSL"></jdbc-resource>
  <custom-resource res-type="javax.net.SocketFactory" jndi-name="cbc/jeswebCC"
            factory-class="com.xlat4cast.jesweb.GlassfishConnectorControlFactory">
    <property name="url" value="localhost:41001" />
  </custom-resource>
  <custom-resource res-type="javax.net.ssl.SSLSocketFactory" jndi-name="cbc/jeswebCCSSL"
            factory-class="com.xlat4cast.jesweb.GlassfishConnectorControlFactory">
    <property name="keystorePassword" value="changeit" />
  </custom-resource>
</resources>
```

Deployment descriptor 4: Glassfish glassfish-resources.xml with no security layer

The instructions supplied in instructions.txt located in the openSSL folder under the JES root directory also cover the case of using Glassfish. Furthermore, It is pretty straight forward to adapt the keytool example that covered the Tomcat case so as to apply to Glassfish as well. One difference is the fact that Glassfish uses s1as as the default alias instead of tomcat. One should also take note of the fact that Glassfish uses the master password that is defined at domain creation time to open its keystore. In order to make the example work, instead of creating a clientCBC/Tomcat key pair, one is to simply export the s1as certificate already present in the Glassfish keystore and import into into the JES truststore.

It is recommended to create a new Glassfish domain so as to define a master password other that “changeit”. Read the Glassfish asadmin help supplied at the command line on details on how to setup a new domain.

3.4 Secure port redirect

Although security is pretty high on the list of prerequisites for a successful JES-WEB deployment, access to its resources via a non-secured channel is not prohibited. To redirect unsecured access so as to enforce a strictly secure connection scheme, uncomment the following code block in web.xml. The file can be found in WEB-INF under the directory that JES-WEB was deflated to, during automatic deployment.

```
<security-constraint>
  <web-resource-collection>
    <url-pattern>/</url-pattern>
  </web-resource-collection>
  <user-data-constraint>
    <transport-guarantee>CONFIDENTIAL</transport-guarantee>
  </user-data-constraint>
</security-constraint>
```

3.5 Security policy files for Tomcat

Security policy entries for Tomcat's security policy file exist under /WEB-INF/classes/policy. Append the contents of the catalina.policy file (located in the folder that corresponds to the version of Tomcat of choice) to the catalina.policy file located in $CATALINA_BASE/conf. The codeBase value for all entries (but the first, which applies to JES-WEBResources.jar) implies that JES-WEB has been copied to the $CATALINA_BASE/webapps, where it has been auto-deployed. Change this value for each grant so as to correspond to the actual JES-WEB deployment folder. Furthermore, all entries that relate to the keystore, the truststore and the CBC listen address/port must be appropriately altered.

4. Additional Notes

4.1 Executing JES-WEB

An instance of JES-WEB is made available by starting the hosting web server. Glassfish is started by invoking asadmin -start-domain <domain_name>, where domain_name corresponds to the domain name supplied at domain creation time (or the default domain, domain1). Tomcat is started by executing $CATALINA_HOME\bin\startup.bat or $CATALINA_HOME\bin\startup.sh at the command line. Both servers can be setup as a window service or a unix deamon. You are referred to their respective documentation for further details and instrunction concerning available options.

4.2 Java Cryptographic Key Strength Policy

Unlimited key strength is required in order to use the 256 bit TLS cipher suites. The unrestricted jurisdiction java policy files should be installed towards this end.

4.3 CBC security levels

There are two security levels for the CBC. One allows unauthenticated access to the CBC and therefore the

backend. The other requires a TLS security layer to be established. Not only does the server certificate need to be verified at the JES-WEB end, but JES-WEB must also be supplied with a certificate to be validated at the CBC end. To accommodate this, the JSSE framework needs access to a keystore containing a valid client certificate and to a truststore containing a trusted CA certificate. The client certificate will have to be validated by its CA's issuing certificate at the CBC end. The JES server certificate will have to be validated by its CA certificate located in the JES-WEB truststore. The administrator has to supply these certificates. A process to generate certificates issued and signed by a custom CA is present in JES.

4.4 CBC commands

All the CBC commands listed in the JES documentation appendix, when a database backend is used, are supported.

4.5 Settings not available through the CBC

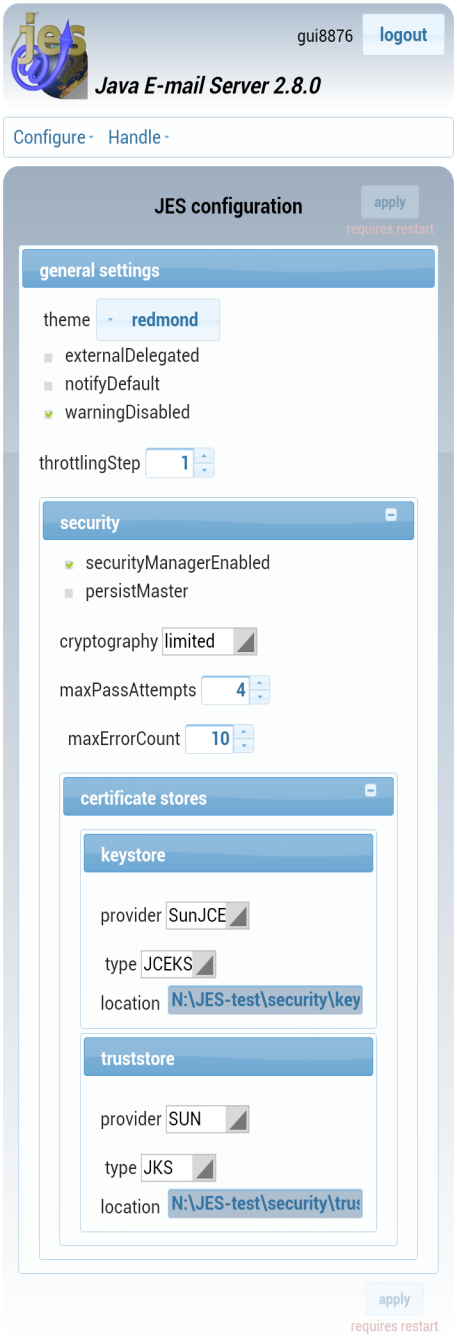
Not all settings specified in mail.xml can be retrieved or modified using the CBC facility, mostly on security grounds. These are the settings for the CBC itself, the javaWrapper section, as well as the allowRemoteRestart and legacyFileIOMode attributes in subsection security under the general section. Finally, no means of specifying the backend type exists, other than directly modifying mail.xml.

5. Accessing JES over the World Wide Web

The first step towards gaining access to JES-WEB is a mandatory login. The username and password corresponds to the guiDbUsername and guiDbPassword credentials listed on Table 1.

After successfully completing the login process, the main index page is presented. The menu bar constists of two drop down submenus, entitled “Configure” and “Handle”.

The configuration segments as declared in mail.xml are selected from the menu items of the "Configure" drop down submenu. If the "allowRemoteRestart[3]" is not set to true, a number of options can not have their values altered.

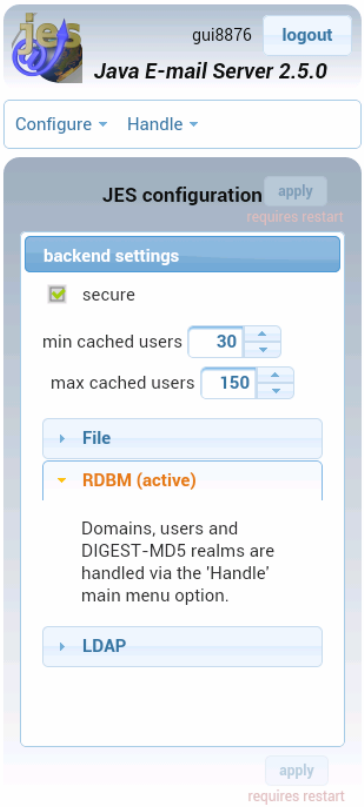


a. GENERAL SETTINGS

The maxPassAttempts values range between 3 and 10. The maxErrorCount values range between 10 and 50. Besides the options that are related to JES there is also a theme selector. There are two themes available to select, redmond and south-street.

b. BACKEND SETTINGS

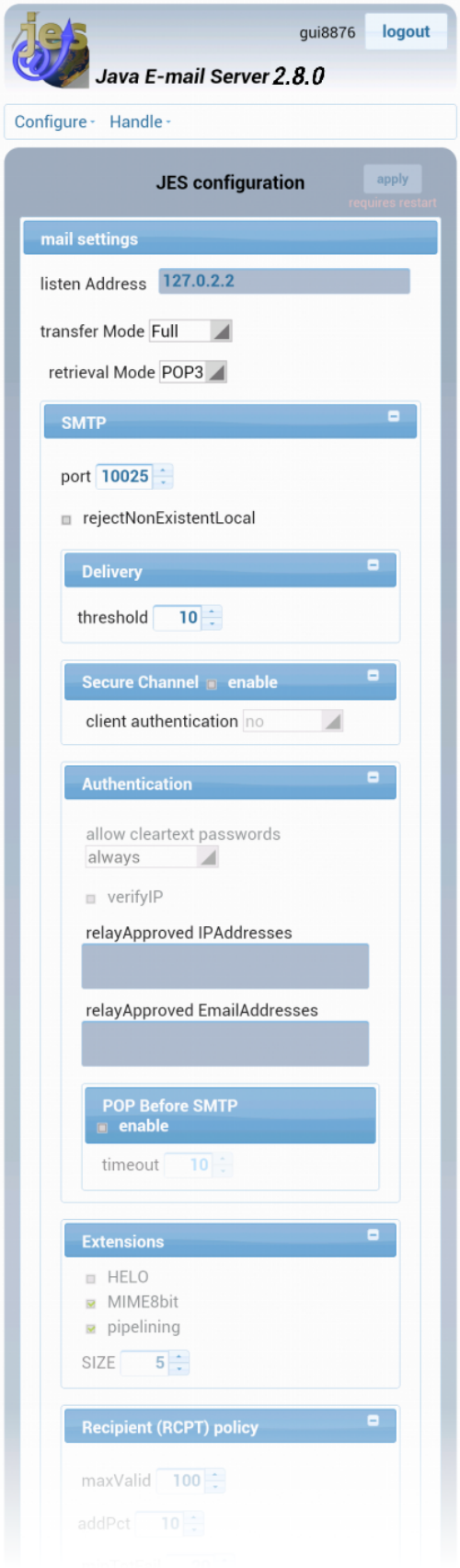
The value of the secure attribute is co-determined by the attribute that carries the same name in the cbc section of mail.xml. If either attribute is true, the other is also forcibly set to the same value. The number of minimum cached users ranges between 15 and 50. The maximum number of cached users ranges between 100 and 1000. These etnries only affect a db backend (using a db is a prerequisite of using JES-WEB). The backend type can not be selected via the CBC



c. MAIL SETTINGS

The deilvery threshold value ranges between 1 and 60. The interval value is specified in seconds. The POPBeforeSMTP timeout value ranges between 1 and 30 mins. The SMTP SIZE

3 AllowRemoteRestart is not available through the CBC.



extension value ranges between 1 and 1024 MiB. The Recipient (RCPT) policy entry values have a restriction only on the lowest allowed value. This lower bound is,

maxValid(Rcpt): 100,
addPct(Rcpt): 5 (a percentage),
minTotFail(Rcpt):20 and
minPctFail(Rcpt): 70 (a percentage).

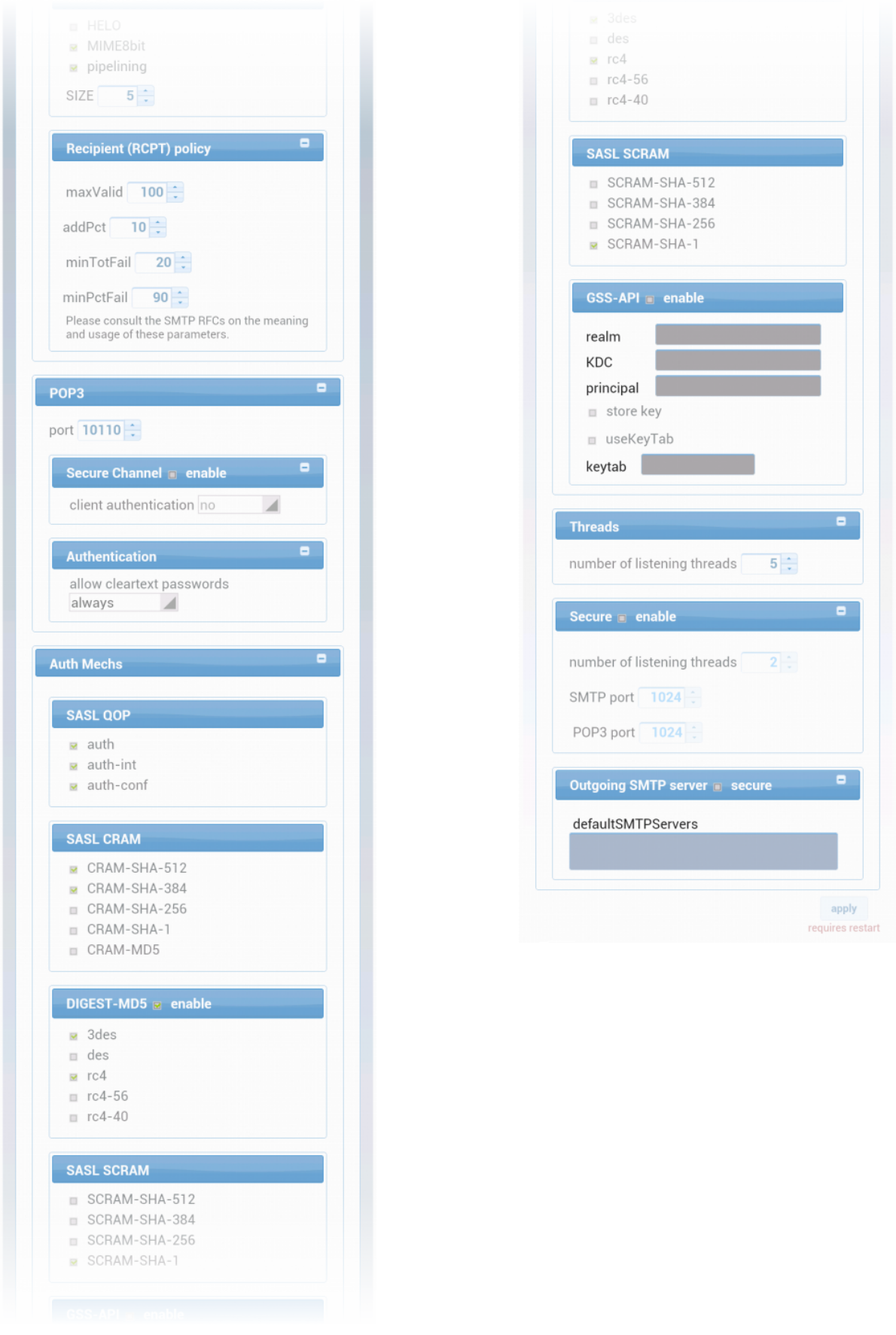
When selecting a SASL QOP value of either auth-int or auth-conf the auth value is also forcibly enabled. Deselecting the auth option also deselects the other QOP options. Finally, the number of standard or secure (SMTP/POP3) listening threads ranges between 2 (standard) or 1 (secure) and 50.

About the screen shots

All the screen shots are taken from emulated Android devices via the Android SDK emulator facility. Three different devices running Android 4(.1.2) have been used:

a. a generic mdpi QVGA 3.2'' device,
b. the Nexus S hdpi 4.00'' device and
c. the Nexus xhdpi 4.65'' device.

The screen shots depict the device displays at approximately their true width.


HELO
MIME8bit
pipelining
SIZE 5
Recipient (RCPT) policy
maxValid 100
addPct 10
minTotFail 20
minPctFail 90
Please consult the SMTP RFCs on the meaning and usage of these parameters.
POP3
port 10110
Secure Channel enable
client authentication no
Authentication
allow cleartext passwords
always
Auth Mechs
SASL QOP
auth
auth-int
auth-conf
SASL CRAM
CRAM-SHA-512
CRAM-SHA-384
CRAM-SHA-256
CRAM-SHA-1
CRAM-MD5
DIGEST-MD5 enable
3des
des
rc4
rc4-56
rc4-40
SASL SCRAM
SCRAM-SHA-512
SCRAM-SHA-384
SCRAM-SHA-256
SCRAM-SHA-1
3des
des
rc4
rc4-56
rc4-40
SASL SCRAM
SCRAM-SHA-512
SCRAM-SHA-384
SCRAM-SHA-256
SCRAM-SHA-1
GSS-API enable
realm
KDC
principal
store key
useKeyTab
keytab
Threads
number of listening threads 5
Secure enable
number of listening threads 2
SMTP port 1024
POP3 port 1024
Outgoing SMTP server secure
defaultSMTPServers
apply
requires restart

d. DIRECTORY SETTINGS

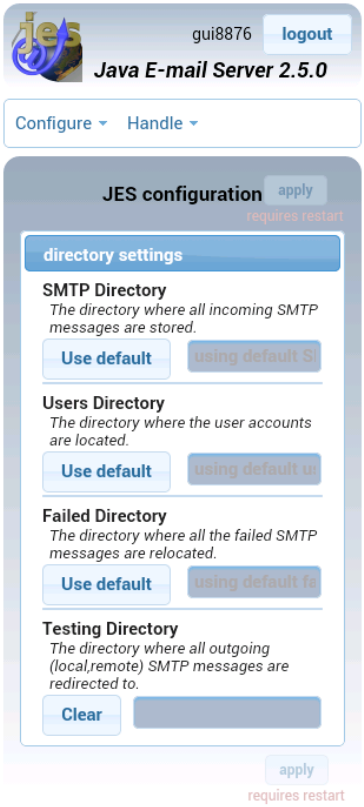
The default directory locations are:

SMTP: <jes-install-dir>/smtp

users: <jes-install-dir>/users

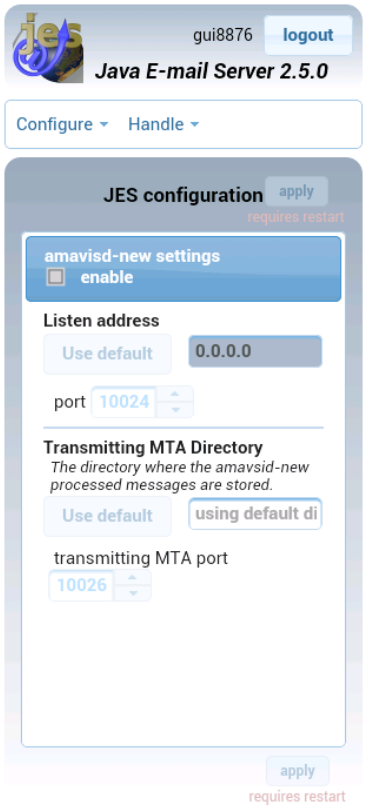
failed: <jes-install-dir>/failed

There is no default testing directory.



e. AMAVIS SETTINGS

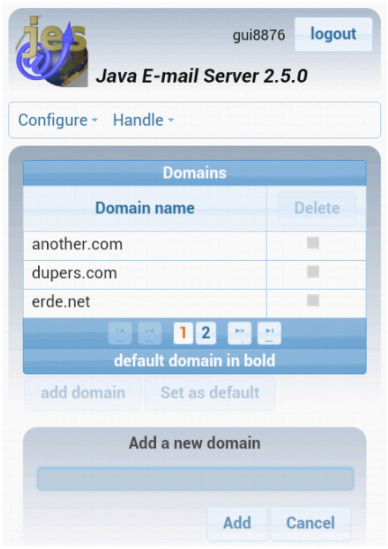
The default transmitting MTA directory is: <jes-install-dir>/amavis. If no instance of amavisd-new is running, enabling its use by JES will simply be ignored.



Handling domains, users and DIGEST-MD5 realms

a. Listing and deleting mail domains

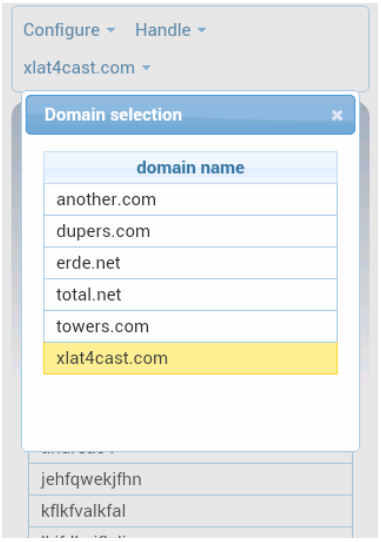
b. Adding a new domain



c. Listing a domain's users

d. Selecting a domain

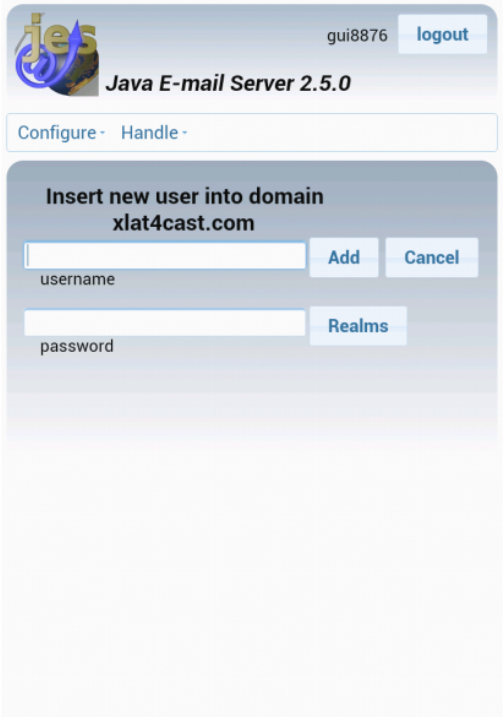
Pressing the menuButton/commandButton located above the user table enables domain selection. If up to 5 domains are administered, a drop down menu appears. Otherwise a dialog is displayed.



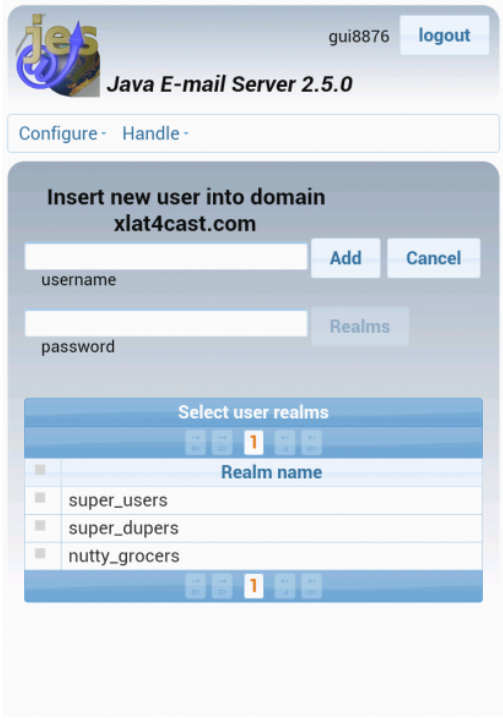
e. The <domain-name> submenu

After selecting a domain (when handling users), a submenu appears in the menu bar that offers a number of user related options. With no selected user, only the option to add a user is available.

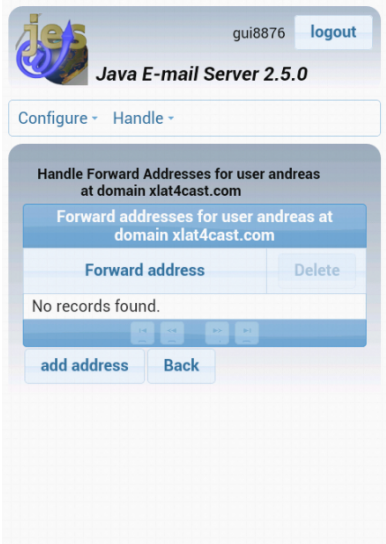
f. Adding a user



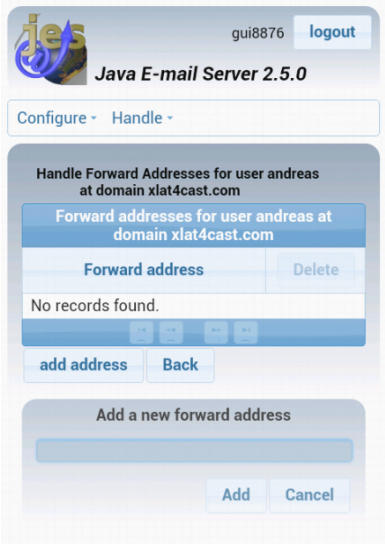
g. Selecting a new user's realms



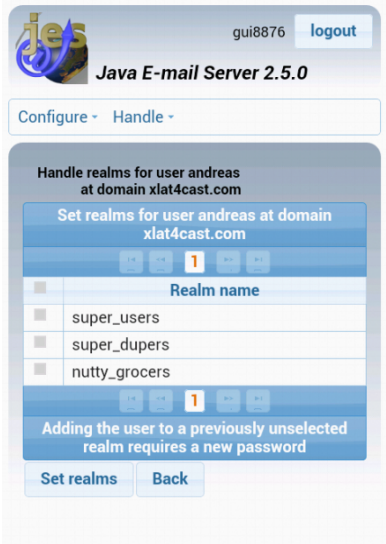
h. Listing/deleting user forward addresses



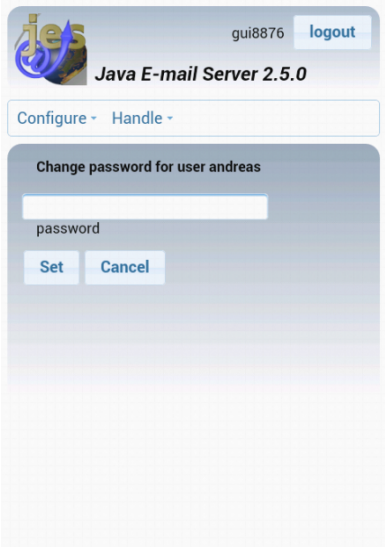
i. Adding a user forward address



j. Setting the user realms



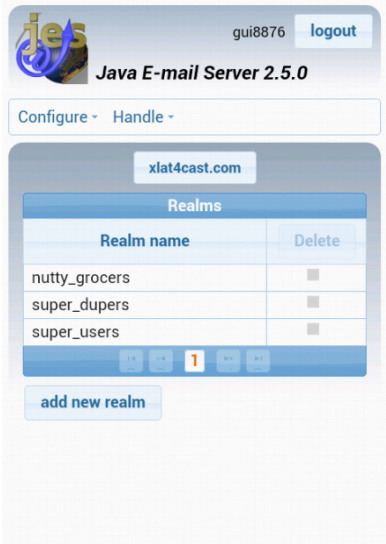
k. Changing a user password



l. Setting the default mailbox

m. Deleting a user

n. Listing/deleting a domain's realms



o. Adding a new realm