FUSION SECURITY

Setup Guide

FUSION SECURITY

Setup Guide

This guide provides information on how to deploy and configure Fusion Security

Fusion Security Version: 1.3

Document Last Updated: May 14th 2014

Contents

1 Fusion Security Dependencies

Fusion Security consists of a single Web Application Archive (war) file called FusionSecurity.war. The war file is deployed to a servlet container, such as Apache Tomcat. It requires a connection to a database and connection to an email server. Also an integral part of the architecture is the use of a robust message queue which allows for the deployment of a high availability and scalable system.

1.1 Servlet Container (required)

Fusion Security is deployed to a Servlet Container. Apache Tomcat is a popular, open source servlet container. Download links and installation instructions can be found at the following URL.

http://tomcat.apache.org/

It is recommended to use the latest version of Apache Tomcat as it will include the latest security patches. Fusion Security will not run in any version lower than 6.0.

Fusion Security **mandates** HTTPS to be enabled on the servlet container, and as such a security certificate must be made available to the servlet container. This is described later in this document.

1.2 SQL-92 Compliant Database (required)

Fusion Security makes use of an Object Relational Mapping (ORM) library called Hibernate. This allows Fusion Security to communicate with any SQL-92 compliant database.

The database must be installed and running and Fusion Security must be configured to connect to the database before Fusion Security is started. Fusion Security will automatically create the database tables on first launch.

NOTE: Version 1.1 has only been tested with MySQL.

1.3 Email Server (optional)

Fusion Security requires a connection to a Simple Mail Transfer Protocol (SMPT) email server. The email server is used in order to email users in the following circumstances:

1. When a user forgets their password.
2. If a user account if automatically locked due to excessive login failure.

2 Deploying Fusion Security

The deployment section assumes Apache Tomcat is being used, but the information provided is transferrable to any servlet container.

Place the Fusion Security war file into the "webapps" directory of your Apache Tomcat server. On Tomcat startup the war file will be deployed into the server and the contents of the war file unzipped to the hard disk.

However a successful start of Fusion Security relies upon obtaining information from a properties file called fusion-security.properties since this information is used to configure Fusion Security. This file is located within the Fusion Security war file within the directory WEB-INF/classes. It is extremely unlikely that the default values will be correct for your system.

Editing this file, supplying the correct values (see Section 3) and re-starting your application server should allow Fusion Security to start successfully.

2.1 Location of properties file

Fusion Security on startup will also check another location to see if it can find the properties file. This location is:

<user home>\FusionSecurity

The actual value of <user home> will vary depending on your Operating System. On a Windows 7 Operating System this will typically be:

C:\users\<your user name>\FusionSecurity

Whereas on a Unix Operating System, it is more likely to be located at:

/users/<your user name>/FusionSecurity

If you wish, you may create this directory and copy the file *fusion-security.properties* from WEB-INF/classes into this directory. Subsequent startups of Fusion Security will attempt to get configuration values from this location too. These values will override the values from the properties file in WEB-INF/classes.

This is optional but the reason to do this is that it allows for easy upgrading of Fusion Security. As subsequent releases of Fusion Security are released, a re-deploy simply involves removing the war file from the Tomcat Web Server and deploying a new one. Your existing properties will be read from your home directory without the need for further configuration.

The values of either *fusion-security.properties* file must be correct. See section 3 for explanation of how to correctly configure the properties file.

If you are unsure about which location is being used to obtain configuration information, look at the startup sequence in the log file. The following shows Fusion Security starting up and the properties being read initially from the WEB-INF/classes directory as well as the home directory for the user 'User1'

```
INFO localhost-startStop-1
org.springframework.beans.factory.config.PropertyPlaceholderConfigurer - Loading properties
file from class path resource [fusion-security.properties]
INFO localhost-startStop-1
```

```
org.springframework.beans.factory.config.PropertyPlaceholderConfigurer - Loading properties
file from URL [file:/C:/Users/User1/FusionSecurity/fusion-security.properties]
```

2.2 Changing the location of the properties file

It is possible to tell Fusion Security to use a different location from your home directory to use for the properties file. This can be useful if you either do not want Fusion Security reading information in your home directory or if you wish to run multiple Fusion Security instances on one server but with different configurations.

To specify a new location you need to set a Java System variable called "SecurityProperties" to the URI value of the location where you wish the properties file to be. In Apache Tomcat the easiest way to achieve this is to create a new file called `setenv.bat` (or `setenv.sh` on Unix environments) and place it in the tomcats `bin` directory. The contents of this file should state the full location of the properties file which must be in the URI format. To illustrate this:

```
SET JAVA_OPTS=-DSecurityProperties=file:///c:/dir/AFile.txt
```
(For Windows systems)

```
export JAVA_OPTS=-DSecurityProperties=file:///dir/AFile.txt
```
(For UNIX systems)

It is important to note that Fusion Security will NOT start if this value is incorrect or if this file cannot be created.

To check that this value is being used by the system, check the log during Fusion Security startup and look for a line like the following:

```
INFO [com.metadatatechnology.util.applicationserver.FusionPropertyPlaceholderConfigurer] -
<Property SecurityProperties has been specified as file:///c:/dir/AFile.txt>
```

3 Configuring Fusion Security Properties

To configure Fusion Security, edit the *fusion-security.properties* file. The default contents are shown below:

```
#########################################################################
#     DATABASE CONNECTION                                               #
#########################################################################
database.security.driver=com.mysql.jdbc.Driver
database.security.username=root
database.security.password=password
database.security.url=jdbc:mysql://localhost:3306/fusion_security
database.security.dialect=org.hibernate.dialect.MySQLDialect

#########################################################################
#     SECURITY (HTTPS PORT REDIRECTION)                                 #
#########################################################################
#Auto Redirect to http:
# any - do not auto redirect
# https - auto redirect traffic coming in on http to https, requires port mapping
information to be correct
requires.channel=any

#Only required if requires.channel=https
port.http=8081
port.https=8444

#########################################################################
#     EMAIL SERVER                                                      #
#########################################################################
mail.smtp=smtp.gmail.com
mail.port=587
mail.username=test@metadatatechnology.com
mail.password=Glw3osUpside4Bal5im1orLambda
mail.security=true

#########################################################################
#     PASSWORD RULES                                                    #
#########################################################################
security.password.timeouthours=2
security.password.dissallowed=
security.password.minlength=1
security.password.minnum=-1
security.password.minchar=-1
security.password.minlower=-1
security.password.minupper=-1
security.password.illegalchars=
max.login.attempt=-1

#########################################################################
#     STRUCTURE WEB SERVICE (ORGANISATION RETRIEVAL)                    #
#########################################################################
```

```
structure.ws=http://localhost:8080/FusionRegistry/ws/rest

########################################################################
#     AUDITING                                                         #
########################################################################
mq.username=
mq.password=
mq.host=
```

3.1 Database Properties (mandatory)

Fusion Security requires the use of a database. The following values must be set correctly and the specified schema must exist. Fusion Security will create the tables with the database schema automatically but it cannot create the database schema itself.

Property	Description
database.security.driver	The Java driver to use. For a MySQL database this must be set to: `com.mysql.jdbc.Driver`
database.security.username	The username to connect to the database.
database.security.password	The password for the user.
database.security.url	The location of your database specified as a URL. For a MySQL database this is of the form: `jdbc:mysql://<server>:<port>/<database schema>`
database.security.dialect	For a MySQL database this must be: `org.hibernate.dialect.MySQLDialect`

To change the database platform, you must:

- Provide an appropriate JDBC driver for the database on the class path,
- Change the JDBC properties (*driver, url, user, password*)
- Change the d*ialect* used by Hibernate to talk to the database

The database configuration details have been defaulted to connect to a MySQL database. Details on changing to another database are detailed in Annex 1 - Alternative Database Platforms.

3.2 HTTPS Port Redirection

The property *requires.channel* can be set to 'any' or 'https'. If this is set to https then any user attempting to access FusionSecurity over http will be redirected to https. FusionSecurity makes use of the port redirections to enable it to reroute the request from the http port to the secure https port. If *requires.channel* is set to https, then the two port properties *port.http* and *port.https* must be set to the correct values for your application server.

3.3 Email Server (optional)

The email server is set up by adding values to the *mail* properties. If the email server does not require a username or password, then these fields can be left blank and *mail.security* must be set to *false*.

3.4 Password Verification Properties

Security password rules are provided to enforce tighter controls on what passwords may be. There are a number of options which are explained in the table below:

Property	Description
security.password.timeouthours	This is the time, in hours, that the user has to reset their

	password, after they have submitted a forgotten password request.
security.password.dissallowed	This is a text file containing a list of illegal passwords. The file is relative to the security.directory property. If no restrictions on passwords are required, then this value can be left blank.
security.password.minlength	The minimum length for a password. Set to -1 if you wish there to be no minimum length.
security.password.minnum	The minimum number of numeric characters (0-9) that must be present in the password. Set to -1 if this setting is not required.
security.password.minchar	The minimum number of alphabetic characters (a-zA-Z) that must be present in the password. Set to -1 if this setting is not required.
security.password.minlower	The minimum number of lower case characters that must be present in the password. Set to -1 if this setting is not required.
security.password.minupper	The minimum number of upper case characters that must be present in the password. Set to -1 if this setting is not required.
security.password.illegalchars	Specified which characters cannot be included in a password. These should be included with no separators, for example: £$%*&^ Leave blank if not applicable.
max.login.attempt	The maximum number of consecutive login failures for a user before their account is automatically locked. Note, in such a situation both the user whose account was locked, and all of the administrators will receive an email. Set to -1 if this setting is not required.

3.5 Structure Web Service (Mandatory)

The *structure.ws* property must be set to a valid SDMX 2.1 RESTful web service. The web service is used by Fusion Security to obtain the list of Agencies, Data Providers, and Data Consumers. An example of this property value is:

http://cloud.sdmxregistry.org/ws/rest

3.6 Reset Password URL

The *security.resetpassword.url* property defines the URL which will be emailed to a user when need to reset their password. This should be set to the server and port of the server which is hosting the application.

4 Launching Fusion Security

Once Fusion Security has been configured, it can be launched. This is achieved by starting the servlet container. Once the servlet container has been started, the web user interface for Fusion Security will be available to via from a web browser at the following URL:

http://server:port/FusionSecurity

NOTE: If the *requires.channel* property is set to *https* then the above URL will automatically redirect to https.

The resulting page will be the login page: this is shown in Figure 1 below.

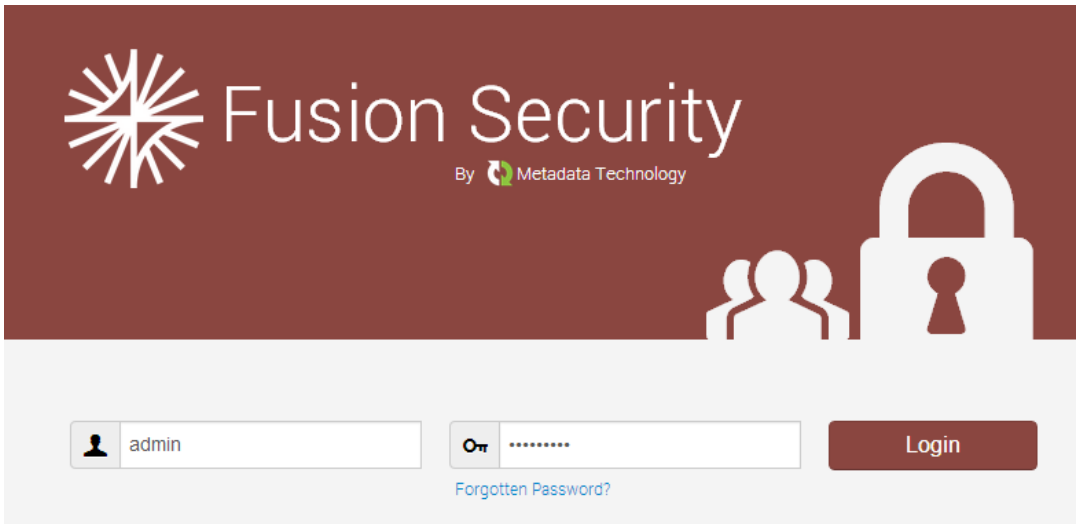

Figure 1 showing the Fusion Security login page

On first launch the database tables will be automatically created. A single user will be created which has the following credentials:

```
username: root
password: password
```

PLEASE BE AWARE

PermGen space is what Tomcat uses to store class definitions and string pools that have been interned. A single Tomcat instance is comfortable loading a single Fusion Security web application. However if you attempt to load another web application (such as Fusion Registry) into the same Tomcat, you may encounter the following error in your Tomcat log:

```
java.lang.OutOfMemoryError: PermGen space
```

The default permGenSpace in Tomcat is only set to 64Mb. To increase this value, it is recommended to create the file setenv.bat (or setenv.sh on Unix environments) and place it in the tomcats bin directory. To set the PermGenSpace to 128Mb set the contents of the file to the following:

```
SET JAVA_OPTS=-XX:MaxPermSize=128m
```

(For Windows systems)

```
export JAVA_OPTS=-XX:MaxPermSize=128m
```

(For UNIX systems)

5 Logging

Fusion Security defaults to output log events to [tomcat]/logs/FusionSecurity.log. Additionally if configured to use Fusion Audit, log events will also be audited.

The default log level is INFO. Log levels, layout and style and output location can all be configured in the following file.

[tomcat]/webapps/FusionSecurity/WEB-INF/classes/log4j.properties

The default configuration is shown below:

```
#ROOT LOGGER
log4j.rootLogger=INFO, fileout, stdout, fusionaudit
log4j.rootLogger.additivity=false

#SPECIFIC LOGGERS
log4j.logger.com.sdmxfusion = INFO
log4j.logger.com.metadatatechnology = INFO
log4j.logger.org.sdmxsource = INFO

#LOG TO FILE (LOCATION, STYLE & LAYOUT)
log4j.appender.fileout=org.apache.log4j.RollingFileAppender
log4j.appender.fileout.file=${catalina.base}/logs/FusionSecurity.log
log4j.appender.fileout.MaxFileSize=1024KB
log4j.appender.fileout.MaxBackupIndex=4
log4j.appender.fileout.layout=org.apache.log4j.PatternLayout
log4j.appender.fileout.layout.ConversionPattern=%p %t %c - %m%n

#LOG TO CONSOLE (STYLE & LAYOUT)
log4j.appender.stdout=org.apache.log4j.ConsoleAppender
log4j.appender.stdout.layout=org.apache.log4j.PatternLayout
log4j.appender.stdout.layout.ConversionPattern=%d %p [%c] - <%m>%n

#LOG TO AUDIT
log4j.appender.fusionaudit=com.sdmxfusion.sdmx.integration.manager.publisher.LogPublisher
log4j.appender.fusionaudit.bufferSize=30
```

For more information about logging please read the log4j documentation which is available online. The version of log4J used in this release is 1.2.14.

6 Configuring HTTPS

It is recommended that communication to Fusion Security is made over a secure channel (HTTPS). This is to ensure that any authentication information passed over the wire is encrypted.

A valid certificate is required to enable HTTPS this is used by the server to prove to the client that it is to be trusted. This certificate should ideally be signed by a trusted Certification Authority (CA). Untrusted certificates may be used but these will cause a slight inconvenience for your end users.

Certificates can be obtained from certificate authorities (e.g. VeriSign / Microsoft / etc.). If you are planning on running Fusion Security in a production environment is it recommended you fully understand how certificates operate and setup a trusted certificate. If you wish to just explore how Fusion Security supports HTTPS you can setup a certificate through the Java supplied application: *keytool*. This application is supplied as part of the Java Development Kit (JDK) and can be used to view the contents of key stores or create a new key store and certificate. Details of how *keytool* works can be found on Oracle's website:

http://docs.oracle.com/javase/7/docs/technotes/tools/windows/keytool.html

The following command creates an **unsigned** certificate and a keystore named *metatech.keystore* which has a password of *password*. When prompted you will need to answer questions regarding the name and organisation of the certificate. For further details please refer to the *keytool* documentation.

```
keytool -genkeypair -alias serverTrust -keyalg RSA -validity 365 -storepass password -
keystore metatech.keystore -storetype JKS -ext san=ip:192.168.4.1
```

Note: Certificates are valid for a specific domain. You may specify your domain name as the CN name of the certificate but in the example above I have specified that the "Subject Alternative Name" (SAN) is the IP address: 192.168.4.1. This means that the machine running on this IP is trusted with this certificate. If you use the example above ensure you modify the SAN value to be the IP address of the machine that is running your Tomcat server.

6.1 Tomcat Configuration

The keystore file must be copied to the `conf` directory of your Apache Tomcat.

Also within the `conf` directory, you will need to edit the file `server.xml`. Locate the section regarding SSL and enable it. It will look something like the following:

```
<Connector port="8443" protocol="HTTP/1.1" SSLEnabled="true"
           maxThreads="150" scheme="https" secure="true"
           clientAuth="false" sslProtocol="TLS"
           keystoreFile="conf/metatech.keystore"
           keystorePass="password" />
```

In the above example the secure port has been defined as 8443. If you are using a secure port other than 8443 you would need to change this value. The value for keystoreFile must be the location of your keystore file (which you just copied to your `conf` directory) and the value for keystorePass must be the correct password for your keystore.

Once the above has been configured you may start your Tomcat application server. In the startup log you will now notice that there will be an output for your new secure port. A typical example is shown below which states that there are two ports open: the first on 8080 (for http connections) and the other on 8443 (for https connections):

```
org.apache.coyote.AbstractProtocol init
INFO: Initializing ProtocolHandler ["http-bio-8080"]
org.apache.coyote.AbstractProtocol init
INFO: Initializing ProtocolHandler ["http-bio-8443"]
```

7 Annex 1 - Alternative Database Platforms

The following list details how to connect to alternative database platforms. Note that in some instances a driver class is required that is not supplied with FusionSecurity. To support a new driver add the relevant jar file into the following folder:

[tomcat]/webapp/FusionSecurity/WEB-INF/lib

7.1.1 Oracle Database Connection

```
database.driver=oracle.jdbc.driver.OracleDriver
database.username=sa
database.password=pwd
database.url=jdbc:oracle:thin:@127.0.0.1:1521:MKYONG
hibernate.dialect=org.hibernate.dialect.OracleDialect
```

7.1.2 SQL Server Database Connection

There are two drivers to connect to SQL Server; the open source jTDS and the Microsoft driver. The driver class and the JDBC URL depend on which one you use.

7.1.2.1 jTDS driver

```
database.driver=net.sourceforge.jtds.jdbc.Driver
database.username=sa
database.password=pwd
database.url=jdbc:jtds:sqlserver://<server>[:<port>][/<database>][;<property>=<value>[;...]
]
hibernate.dialect=org.hibernate.dialect.SQLServerDialect
```

7.1.2.2 Microsoft SQL Server JDBC 3.0

```
database.driver=com.microsoft.sqlserver.jdbc.SQLServerDriver
database.username=sa
database.password=pwd
database.url=jdbc:sqlserver://[serverName[\instanceName][:portNumber]];databaseN
ame=
hibernate.dialect=org.hibernate.dialect.SQLServerDialect
```

A complete list of Hibernate dialects is shown in the table below

RDBMS	Dialect
DB2	org.hibernate.dialect.DB2Dialect
DB2 AS/400	org.hibernate.dialect.DB2400Dialect
DB2 OS390	org.hibernate.dialect.DB2390Dialect
PostgreSQL	org.hibernate.dialect.PostgreSQLDialect
MySQL	org.hibernate.dialect.MySQLDialect
MySQL with InnoDB	org.hibernate.dialect.MySQLInnoDBDialect
MySQL with MyISAM	org.hibernate.dialect.MySQLMyISAMDialect
Oracle (any version)	org.hibernate.dialect.OracleDialect
Oracle 9i/10g	org.hibernate.dialect.Oracle9Dialect
Sybase	org.hibernate.dialect.SybaseDialect
Sybase Anywhere	org.hibernate.dialect.SybaseAnywhereDialect
Microsoft SQL Server	org.hibernate.dialect.SQLServerDialect
SAP DB	org.hibernate.dialect.SAPDBDialect
Informix	org.hibernate.dialect.InformixDialect

HypersonicSQL	`org.hibernate.dialect.HSQLDialect`
Ingres	`org.hibernate.dialect.IngresDialect`
Progress	`org.hibernate.dialect.ProgressDialect`
Mckoi SQL	`org.hibernate.dialect.MckoiDialect`
Interbase	`org.hibernate.dialect.InterbaseDialect`
Pointbase	`org.hibernate.dialect.PointbaseDialect`
FrontBase	`org.hibernate.dialect.FrontbaseDialect`
Firebird	`org.hibernate.dialect.FirebirdDialect`