# FUSION SECURITY

Setup Guide

FUSION SECURITY

## Setup Guide

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

Fusion Security: 2.x

Document Last Updated: January 10th 2017

Version: 20170110

# Contents

# 1 Fusion Security Dependencies

## 1.1 Scope of this document

The purpose of this document is to provide an overview of how to install Fusion Security in a Java Servlet Container. This document explains how to configure Fusion Security and specific advice on configuring aspects of the Java Servlet Container, Apache Tomcat.

It is recommended that readers have familiarity with the appropriate required dependencies which are listed in the following sections.

## 1.2 Fusion Security Distribution

Fusion Security is distributed as a single Web Application Archive (war) file: *FusionSecurity.war* it should be deployed to a servlet container such as Apache Tomcat (a free product).

## 1.3 Java (required)

A Java Runtime Environment (JRE) 1.7 or higher is required.

## 1.4 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. The Fusion products will not run in any Apache Tomcat version lower than version 6.0.

The rest of this document will only give servlet container information regarding Apache Tomcat.

## 1.5 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. This distribution has only been set up to connect to MySQL, Oracle, and SQL Server databases. The minimum version tested by Metadata Technology for these databases are: MySQL 5.5, Oracle 10g and SQL Server 2010.

If your database is not one of these types, please contact Metadata Technology, as we may be able to add your database to the list of supported database management systems.

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 required database tables on launch if they are not already present.

## 1.6 Email Server (optional, recommended)

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

## 2.1 Choice of Java Servlet Container

Fusion Security must be run within a Java Servlet Container. Metadata Technology recommends using Apache Tomcat as the Java Servlet Container, as this has been used during the testing lifecycle of Fusion Security. Metadata Technology has not tested Fusion Security when deployed on other Java Servlet Containers and therefore cannot guarantee that Fusion Security will work without further modification.

## 2.2 Deployment Using Tomcat

Fusion Security consists of a single .war file called *FusionSecurity.war*. This file needs to be copied into the directory: *<TOMCAT_HOME>/webapps* then the Tomcat server should be started. As the Tomcat application server starts, the contents of the Fusion Security war file will be unpacked into the directory:

*<TOMCAT_HOME>/webapps/FusionSecurity*

Please check the Tomcat log files to ensure that Fusion Security has deployed correctly. Once it has then you may navigate to the URL:

*http://[server]:[port]/FusionSecurity/*

The values for server and port must be replaced with the IP address and port number that the web application server is running on. For example, if the web browser is running on the same machine as the web application server and the Apache Tomcat has not had its default port settings modified, then the following address can be used:

*http://localhost:8080/FusionSecurity/*

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.

**Note:** It is extremely unlikely that the default values will be correct for your system. Please see section 3 which explains how to configure the Fusion Security properties file.

Editing the properties file, supplying the correct values and re-starting your application server should allow Fusion Security to start successfully.

Once Fusion Security has been configured, it can be launched. This is achieved by starting the servlet container (e.g. Apache Tomcat). Once started the web user interface for Fusion Security will be available to view 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 below.

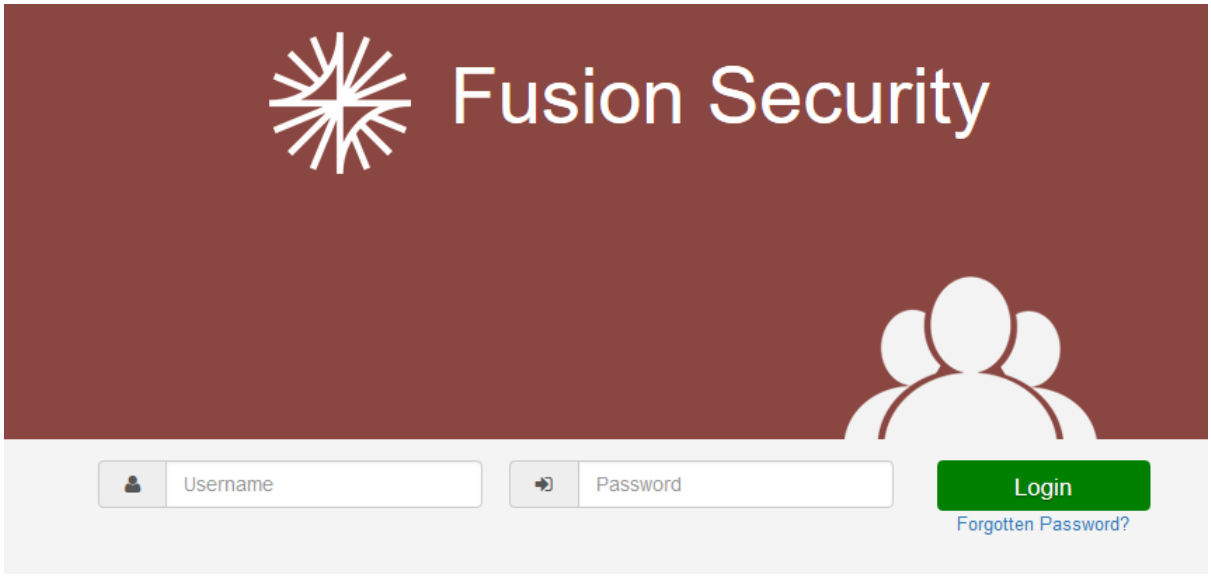


Figure 1 showing the Fusion Security login page

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 refer to the Fusion Security User Guide for further information about using Fusion Security.

## 2.3 Configuring Tomcat Memory

When using a Java version before Java version 1.8, PermGen space may become an issue. 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:

(For Windows systems)

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

(For Unix systems)

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

# 3 Fusion Security Properties File

## 3.1 Introduction

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]
```

## 3.2 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>
```

## 3.3 Property File Contents

The default contents of the properties file (fusion-security.properties) 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

#########################################################################
#     CERTIFICATE AUTHENTICATION                                        #
#########################################################################
security.p12File=
security.p12Pass=

#########################################################################
#     EMAIL SERVER                                                      #
#########################################################################
mail.smtp=
mail.port=
mail.username=
mail.password=
mail.security=true
mail.from=

#########################################################################
#     PASSWORD RULES                                                    #
#########################################################################
security.password.timeouthours=2
security.password.disallowed=
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=CloudRegistry@https://registry.sdmxcloud.org

#########################################################################
#     GENERAL SETTINGS                                                  #
#########################################################################
server.url=
```

## 3.4 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.

The table below gives information on each of the database properties.

| 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 dialect used by Hibernate to talk to the database

The database driver details have been defaulted to connect to a MySQL database. Details on changing to another database are detailed in the Annex 1 and for Single Sign On (SQL Server only) refer to section 7.

## 3.5 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.

## 3.6 Certificate Authentication

These two properties are required if the Fusion Registry is set to enforce security. If enforce security

is true then Fusion Security must authenticate itself with Fusion Registry in order to obtain any organisation details. Without these properties Fusion Security will not work against a private Fusion Registry.

The *security.p12File* property is a reference to the p12 File on the local file system, the *security.p12Pass* is the corresponding password to the p12 file.

The Fusion Registry must contain the security p12 certificate in its own Tomcat trust store. Adding certificates to trust stores in documented in the Fusion Registry Setup Guide.

## 3.7 Email Server (optional)

The email server is set up by specifying values to the *mail* properties. If you do not require an e-mail server, simply leave these fields blank. 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*. The *mail.from* value can be used to specify a particular e-mail from address.

## 3.8 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.disallowed** | This is a text file containing a list of illegal passwords. If no path is specified then the file is relative to how Fusion Security was started, it is recommended to supply the full path. If no restrictions on passwords are required, then this value can be left blank. A default file of the 10,000 most common passwords is supplied with Fusion Security (filename: "MostCommonPasswords.txt"). It is recommended at the least, that this setting is used when running Fusion Security.<br><br>**Examples**<br>**Windows: C:/passwords/pwdDisallowed.txt**<br>**Unix: /home/FusionSecurity/illegalPwdList.txt** |
| **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: £$%*&^<br>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. |

## 3.9 Structure Web Service (Mandatory)

The *structure.ws* property must be set to reference at least one valid Registry that can communicate via the SDMX 2.1 RESTful web service. This web service provides a **domain** for Fusion Security to obtain a list of Agencies, Data Providers, and Data Consumers. Fusion Security can connect to multiple domains in order to obtain known organisation information. A user account in Fusion Security can be linked to multiple organisations across multiple domains. When a Fusion Product authenticates a user with Fusion Security it will only apply the user roles if they are in the same domain that the product is deployed to.

This property takes the domain alias followed by the '@' symbol followed by the URL of the Registry. Multiple domains are semicolon separated. For example:

```
CloudRegistry@http://registry.sdmxcloud.org;FusionMatrixDemo@http://demo.metadatatechnology.com/FusionMatrix
```

**Note:** Fusion Products such as the Fusion Registry will automatically try to determine the URL that they are deployed to. It is important that the Fusion Product's deployed URL matches that of the domain set in Fusion Security. As each product allows the deployed URL to be explicitly set, we would recommend that this value is set explicitly. The installation guide for each product will describe how this is achieved.

## 3.10 General Settings

The 'server.url value' is the fully-qualified URL by which Fusion Security will be accessed by (e.g. http://security.myserver.com). It is used by Fusion Security when it needs to communicate the URL to a user and for redirection on various web pages. This is not a mandatory setting as, if not set, then Fusion Security will obtain the URL from the servlet container and this is often sufficient for Fusion Security to operate properly. However if you have deployed Fusion Security to an environment where the Web Server is performing redirection and have locked down specific ports, then you may find this value required.

For example, in the scenario where Fusion Security has been deployed to a server on port 8081 and is sitting behind an Apache WebServer, if server.url has not been set in the properties file, then Fusion Security may default the value to http://localhost:8081/FusionSecurity. If this is not accessible through the Apache WebServer, then certain links will not work. In this scenario it is necessary to specify the server.url value correctly.

# 4 Logging

Fusion Security defaults to output log events to the [tomcat]/logs directory.

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/logback.xml*

The default configuration is shown below:

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration scan="true" scanPeriod="3 seconds" >

  <appender name="STDOUT" class="ch.qos.logback.core.ConsoleAppender">
    <encoder>
      <pattern>%d{yyyy-MM-dd HH:mm:ss.SSS} %p [%c] - %msg%n</pattern>
    </encoder>
  </appender>

  <root level="INFO">
    <appender-ref ref="STDOUT" />
  </root>
</configuration>
```

For more information about logging please read the logback documentation which is available online.

# 5 Configuring HTTPS

## 5.1 Overview

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.

## 5.2 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"]
```

# 6 Root Account: Change Password and Unlock account

## 6.1 Command Line Application

In the event that the root account is locked or the root password is forgotten then the root user will not be able to access the GUI. The Fusion Security command line application (FusionSecurityCL) can be used to reset the root password, unlock the account or remove any IP restrictions on the root account.

FusionSecurityCL is a runnable jar file distributed in the Fusion Security Zip file. It is run through the command line in the following manner:

```
java –jar FusionSecurityCL.jar
```

FusionSecurityCL takes the following command-line arguments:

| Argument | Value | Description |
|---|---|---|
| **-pwd** | The new password for the root account. | Sets the root user's password to the specified value. |
| **-locked** | Either the value true or false. | Locks or unlocks the root user account. |
| **-removeip** | None. | Removes all IP restrictions from the root user. |
| **-help** | None. | Displays the help message. |

FusionSecurityCL requires a properties file to run in order to know which database to modify. It is advisable to use the same properties file as Fusion Security. The properties file must be placed in the same location as the jar file 'FusionSecurityCL.jar' in order for FusionSecurityCL to run correctly.

**Note:** You must restart Fusion Security for the changes to the root account to take effect.

## 6.2 Example Usage

To change the Root user's password to "Fusion" issue the following command:

```
java –jar FusionSecurityCL.jar –pwd Fusion
```

To unlock the Root account (without changing the password) then issue the following command:

```
java –jar FusionSecurityCL.jar –locked false
```

To lock the Root account then issue the following command:

```
java –jar FusionSecurityCL.jar –locked false
```

To remove all IP restrictions then issue the following command:

```
java –jar FusionSecurityCL.jar –removeip
```

# 7 Single Sign On (SQL Server only)

## 7.1 Enabling Single Sign On

Fusion Security 2 supports Single Sign On (SSO) when connecting to a SQL Server database. To enable SSO, this is a two part process:

1. Supply the relevant DLL to Fusion Security.
2. Edit the properties file to specify Single Sign On.

## 7.2 Obtaining and Installing SQLJDBC DLL

The DLL can be obtained directly from Microsoft. Download the "Microsoft SQL Server JDBC Drivers" package which contains a number of drivers named "sqljdbc_auth .dll" but for different systems (e.g. x86, 64 bit, etc.). Locate the appropriate DLL for your system.

This DLL needs to be supplied to the Java Runtime running your Web Application Server. There are a number of ways in which this can be achieved. Two of the simplest methods are listed below:

1. Copy the DLL file to the Java Runtime "bin" directory that is running your Web Application Server. It is important to place the DLL in the correct directory (for example: C:\Java\jdk1.8.0_92\jre\bin). Note: that modifying a Java Runtime in this manner means that all applications that use this Java Runtime will be affected.
2. Pass the DLL location to the Web Application Server on server startup. For Apache Tomcat this can be achieved by modifying the "setenv.bat" file located in the Tomcat bin directory. Locate the directory with the DLL you wish to add (e.g. c:\temp) then add the following line to setenv.bat and the Java library path will be modified allowing Tomcat to access the DLL file:

```
set CATALINA_OPTS=%CATALINA_OPTS% -Djava.library.path=C:\temp\SSO_DLL
```

## 7.3 Modification to Fusion Security Properties File

Edit the Fusion Security properties file. Remove all of your existing database property entries and replace them with the following:

```
database.security.dialect=org.hibernate.dialect.SQLServerDialect
database.security.driver=com.microsoft.sqlserver.jdbc.SQLServerDriver
database.security.url=jdbc:sqlserver://localhost:64771;databaseName\=fusion_s
ecurity;integratedSecurity\=true;
```

The value for database.security.url will need to be modified so that it communicates with your SQL Server correctly. The format is as follows:

```
jdbc:sqlserver://<server>:<port>;databaseName\=<database schema>;integratedSecurity\=true;
```

And the values for *server*, *port* and *database schema* must be supplied appropriately.

## 7.4 Troubleshooting

When starting Fusion Security, if you see an error like the following, then the DLL could not be located or is the wrong version for your system:

```
java.lang.UnsatisfiedLinkError: no sqljdbc_auth in java.library.path
```

In this scenario, please perform the following:

1. Ensure you are using the correct DLL for your system.
2. Ensure that this is being loaded correctly by your JRE.
3. Ensure that you have edited the Fusion Security properties file correctly.

# 8 External Web Services

Fusion Security runs a number of web services to provide Fusion applications with security services. If Fusion Products are deployed to a different server then Fusion Security then the relevant web services must be available to these external Fusion Products.

The web services are listed below.

| Service URL | Service Description |
|---|---|
| **/ws/auth** | Used to authenticate a login request |
| **/ws/unsecured/forgot/forgotPassword** | Used to email a reset password for the given user account |
| **/ws/unsecured/forgot/forgotPasswordGivenEmail** | Used to email a reset password for the given user account |
| **/ws/unsecured/forgot/forgotUsername** | Used to email a user their username based on their email address |
| **/ws/unsecured/forgot/resetPassword** | Resets the users password |
| **/ws/fusion/info/product** | Provides external Fusion products information about the Fusion Security product, including its version number |
| **/ws/fusion/securekey/generate** | Used by the Fusion Matrix to generate a public/private key pair |
| **/ws/fusion/securekey/retrieve** | Used by the Fusion Matrix to retrieve the private key part of the public/private key pair |

# 9 Annex 1 - Alternative Database Platforms

## 9.1 Introduction

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 Fusion Security. To support a new driver add the relevant jar file into the following folder:

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

### 9.1.1 Oracle Database Connection

To connect to an Oracle database, use the following connection properties (changing the values as required) in the file fusionsecurity.properties:

```
database.security.driver=oracle.jdbc.driver.OracleDriver
database.security.username=<user id>
database.security.password=<password>
database.security.url=jdbc:oracle:thin:@<server>:<port>:<schema_name>
database.security.dialect=org.hibernate.dialect.OracleDialect
```

### 9.1.2 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.

#### *9.1.2.1 jTDS driver*

```
database.security.driver=net.sourceforge.jtds.jdbc.Driver
database.security.username=<user id>
database.security.password=<password>
database.security.url=jdbc:jtds:sqlserver://<server>[:<port>][/<database>][;<property>=<val
ue>[;...]]
database.security.dialect=org.hibernate.dialect.SQLServerDialect
```

#### *9.1.2.2 Microsoft SQL Server JDBC 3.0*

```
database.security.driver=com.microsoft.sqlserver.jdbc.SQLServerDriver
database.security.username=<user id>
database.security.password=<password>
database.security.url=jdbc:sqlserver://[serverName[\instanceName][:portNumber]];databaseNam
e=
database.security.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` |