

## Jboss encryption module and web frontend for TIM

---

TIM encoder uses a Jboss module for encryption and to validate keys. The encryption is used e.g. for database connections and third-party applications within the configuration files in order to hide sensitive information like usernames and passwords from plain sight.

To use this module the following steps need to be done:

1. Add the encoder.war of the TIM Encryption Webarchive
2. Modiy the standalone-tim.xml
3. Generating a java key with web-interface or console
4. Use encryption with TIM

### Add the encoder.war of the TIM Encryption Webarchive

---

To enable TIM to encrypt usernames and passwords and to use those encrypted values insert the encoder.war to the standalone\deployments folder in your %JBOSS\_HOME% e.g. C:\tim\jboss-eap-7.1\standalone\deployments. Upon JBoss startup it will deploy automatically.

### Modify the standalone-tim.xml

---

```
<subsystem xmlns="urn:jboss:domain:datasources:1.2">
  <datasources>
    <datasource jta="true" jndi-name="java:/doorisPortalDB" pool-name="doorisPortalDB" enabled="true" use-java-context="true" use-ccm="true">
      <connection-url>jdbc:sqlserver://localhost:1433:databaseName=test:</connection-url>
      <driver>sqlserver</driver>
      <pool>
        <min-pool-size>20</min-pool-size>
        <max-pool-size>150</max-pool-size>
      </pool>
      <!--security>
      <user-name>test</user-name>
      <password>test</password>
      </security-->
      <security>
        <security-domain>secDomDS</security-domain>
      </security>
      <validation>
        <check-valid-connection-sql>SELECT 1</check-valid-connection-sql>
        <validate-on-match>true</validate-on-match>
        <background-validation>>false</background-validation>
        <use-fast-fail>>false</use-fast-fail>
      </validation>
    </datasource>
  </datasources>
</subsystem>
```

e

standalone.xml are made accordingly. In your subsystem datasource below your database implemet:

```
<security>
  <security-domain>secDomDS</security-domain>
```

```
</security>
```

as can be seen in the first screenshot.

In the *subsystem security* add:

```
<security-domain name="secDomDS" cache-type="default">
  <authentication>
    <login-module
code="org.picketbox.datasource.security.TimSecureIdentityLoginModule"
flag="required">
      <module-option name="username" value="$enc$c5507593f47122e"/>
      <module-option name="password"
value="$enc$-3c3702fd5f714bd0045dcdcd12584c8"/>
    </login-module>
  </authentication>
</security-domain>
```

as can be seen in the screenshot below.

```
<subsystem xmlns="urn:jboss:domain:security:1.2">
  <security-domains>
    <security-domain name="other" cache-type="default">
      <authentication>
        <login-module code="Remoting" flag="optional">
          <module-option name="password-stacking" value="useFirstPass"/>
        </login-module>
        <login-module code="RealmDirect" flag="required">
          <module-option name="password-stacking" value="useFirstPass"/>
        </login-module>
      </authentication>
    </security-domain>
    <security-domain name="jboss-web-policy" cache-type="default">
      <authorization>
        <policy-module code="Delegating" flag="required"/>
      </authorization>
    </security-domain>
    <security-domain name="jboss-ejb-policy" cache-type="default">
      <authorization>
        <policy-module code="Delegating" flag="required"/>
      </authorization>
    </security-domain>
    <security-domain name="loom" cache-type="default">
      <authentication>
        <login-module code="com.dooris.security.LoomLoginModule" flag="required"/>
      </authentication>
    </security-domain>
    <security-domain name="secDomDS" cache-type="default">
      <authentication>
        <login-module code="org.picketbox.datasource.security.TimSecureIdentityLoginModule" flag="required">
          <module-option name="username" value="$enc$3b7965db3d853e17"/>
          <module-option name="password" value="$enc$-44f6b421a454e75a"/>
        </login-module>
      </authentication>
    </security-domain>
  </security-domains>
</subsystem>
```

## Generating a java key with web-interface or console

There are two ways to encrypt your credentials. Either with the web-interface or with via the console.

The module can be called via the web-interface e.g. [http://your\\_tim\\_url:port/encoder/](http://your_tim_url:port/encoder/) . In order to

encrypt a secret insert use the text field and hit the *encrypt secret* button. To validate a encrypted secret past the secret in the correct text field and hit the *validate encrypted secret* button. This method can as well be used to encrypt any secret in e.g tim.properties or dashboard.properties.

To encode your credentials via console use the following commands:

to generate a key:

```
java -cp modules/system/layers/base/org/picketbox/main/tim-encoder-
module.jar:modules/system/layers/base/org/picketbox/main/picketbox-4.1.1.Fin
al-redhat-1.jar
org.picketbox.datasource.security.TimSecureIdentityLoginModule '123'
```

to validate password-key combination:

```
java -cp modules/system/layers/base/org/picketbox/main/tim-encoder-
module.jar:modules/system/layers/base/org/picketbox/main/picketbox-4.1.1.Fin
al-redhat-1.jar
org.picketbox.datasource.security.TimSecureIdentityLoginModule '123'
'$enc$b530c41fe274111'
```

to validate the key:

```
java -cp modules/system/layers/base/org/picketbox/main/tim-encoder-
module.jar:modules/system/layers/base/org/picketbox/main/picketbox-4.1.1.Fin
al-redhat-1.jar
org.picketbox.datasource.security.TimSecureIdentityLoginModule ''
```

'\$enc\$b530c41fe274111'

## TIM Properties

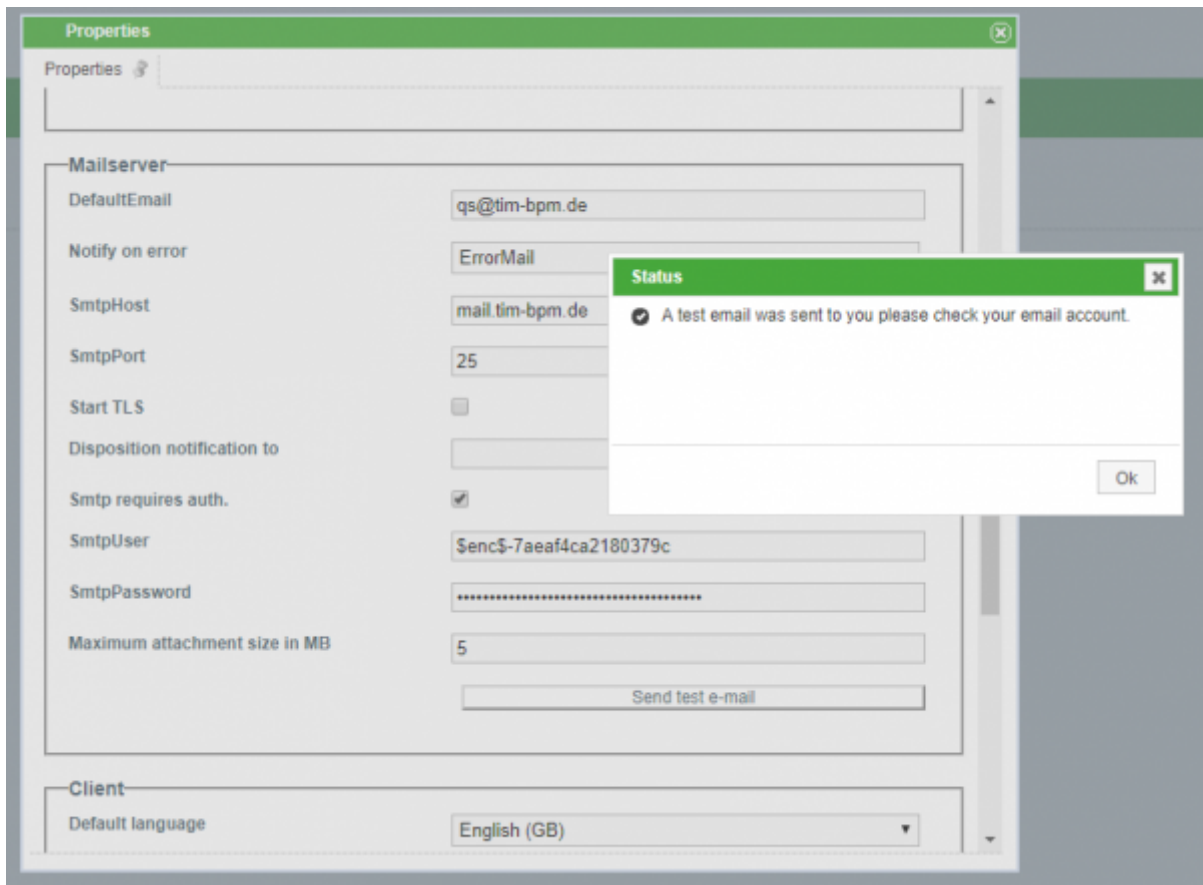
In *tim.properties* credentials can be encrypted with the encoder or the console and replace plain text usernames and passwords. As an example the image shows setting and editing the superuser and client-administrator passwords:

```
#Default init password  
initpass-super-admin=$enc$7875f8726eled2c7  
initpass-super-sys.support=$enc$-66c2ca6af02ba4c5  
initpass-x-admin=$enc$66cb4a689ca87a97  
initpass-x-sys.support=$enc$-37fa8d4f2b81e0c9  
initpass-x-others=$enc$6ec2f555069f3ca1
```

## E-Mail Configuration

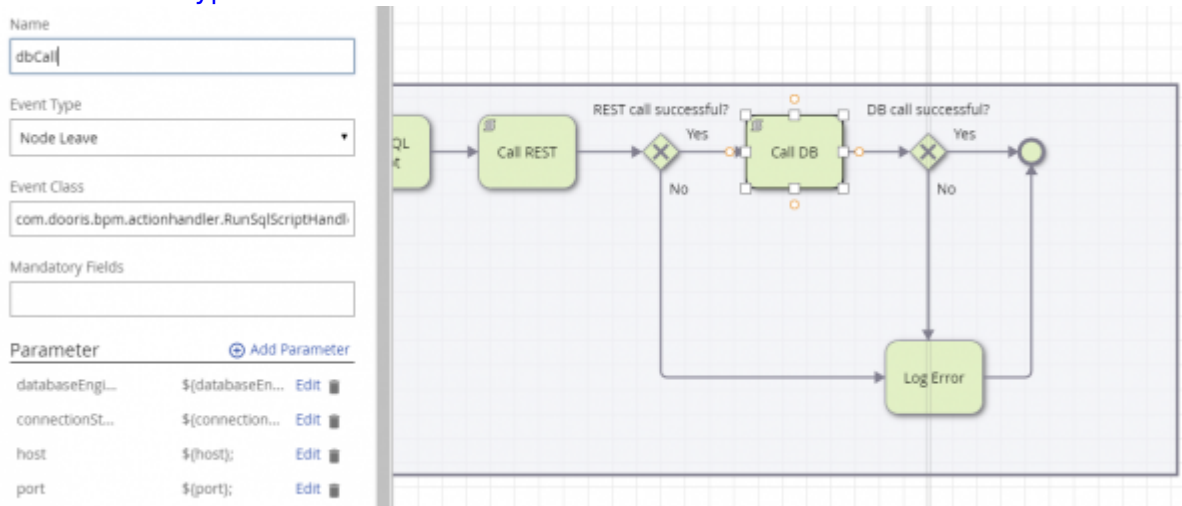
In the properties of the client under the category *mail server* are the parameters *SmtplibUser* and *SmtplibPassword*.

These credentials can be encrypted accordingly to the methods above and as shown in the example.



# Actionhandler

Actionhandler like HTTPRequestHandler or RunSqlScriptHandler use credentials that can be encrypted by said methods. The example shows the RunSqlScriptHandler where `#{user}` and `#{pass}` are passed on encrypted via smartform:



RunSqlScriptHandler

databaseEngine:

connectionString:

host:

port:

database:

User:

pass:

query:

# Timer

Just like *actionhandler* TIM can encrypt necessary credentials for *timer* as well and hide them from plain text. The example shows the Timer signalByMailReply:

Timer name	Webservice name	Webservice methode	Parameter	Intervall	Max. recursion	Act. recursion	Status	Last execution
signalProcessInstanceByNodeNameAndTime	ProcessInstanceManager	signalProcessInstanceByNodeNameAndTime	%signalProcessInstanceByNode% Activity, ts, 100	1s	1	0	-	25/06/2018, 04:49
SignalByMailReply	ProcessInstanceManager	signalByMailReply	qs@tim-bpm.de;Senc5-30992390ec9662f6c39275f6665b4a84.143.192.168.1.10_TNR-4273;SignalByMailReply-*(Wait4)*;false.helper.nein.nein;ja.ja.(Wait4?);Antwort: ?s	10m	1	0	-	09/04/2018, 09:48

From: <https://wiki.tim-solutions.de/> - **TIM Wiki** / **NEW TIM 6 Documentation**

Permanent link: [https://wiki.tim-solutions.de/doku.php?id=en:software:tim:encryption\\_tim&rev=1529935233](https://wiki.tim-solutions.de/doku.php?id=en:software:tim:encryption_tim&rev=1529935233)

Last update: **2021/07/01 09:54**

