

# eHealth Framework 2.8

## Installation Manual

Installation Manual | Version 1.0 | Status Approved  
Security Level Public

# Imprint

InterComponentWare AG  
Industriestraße 41  
69190 Walldorf, Germany  
Tel.: +49 (0) 6227 385 0  
Fax: +49 (0) 6227 385 199

Document Version: 1.0  
Document ID: a07c0bcd-6c0d-2c10-8bab-fe1b24bf1249  
Document Language: en (US)  
Security Level: Public  
Product Name: eHealth Framework  
Product Release: 2.8  
Last Change: 4/30/2009  
Editorial Staff: BAS Technology

# Table of Contents

<b>1</b>	<b>Introduction .....</b>	<b>5</b>
1.1	Document Overview .....	5
1.2	Document Conventions .....	6
<b>2</b>	<b>Installation Prerequisites .....</b>	<b>7</b>
2.1	Installation Scope .....	7
2.2	Users and User Rights .....	7
2.3	Recommended Installation Directory .....	7
<b>3</b>	<b>Installing the Components.....</b>	<b>8</b>
3.1	Installation Files.....	8
3.2	Creating the Target Installation Directory.....	9
3.3	Installing Java 6.....	10
3.4	Installing Apache Ant.....	12
3.5	Installing Apache Tomcat.....	13
3.6	Installing Apache HTTPD.....	14
<b>4</b>	<b>Configuring the Oracle Data Base.....</b>	<b>17</b>
4.1	Prerequisites for Oracle.....	17
<b>5</b>	<b>Configuring the eHealthFramework.....</b>	<b>18</b>
5.1	Proxy- and Database Property Setting for eHealth Framework.....	19
5.1.1	Audit Specific Database Connection Properties: .....	21
5.2	Setting Product-global Configuration Properties .....	22
5.3	Settings Product Instance Specific Properties .....	23
5.4	Setting Password Specific Configuration Properties.....	24
5.5	Setting Certificate Validation Specific Configuration Properties.....	25
5.6	Setting Token Service Specific Configuration Properties.....	26
5.7	Setting Token Filter Specific Configuration Properties.....	27
5.8	Deployment Preparations.....	30
<b>6</b>	<b>Creating Appropriate Grants for Oracle .....</b>	<b>31</b>
<b>7</b>	<b>Creating the Initial Values in the Database .....</b>	<b>35</b>
7.1	Importing Additional Data (optional) .....	35

<b>8</b>	<b>Deploying and Starting the Application .....</b>	<b>36</b>
8.1	Starting the application .....	36
<b>A</b>	<b>Setting up eHealth Framework with Glassfish.....</b>	<b>37</b>
A 1	Installing Glassfish.....	37
A 2	Starting and Stopping Glassfish .....	41
A 3	Deploying and Undeploying eHF in Glassfish .....	42
A 4	Logging and finding issues in Glassfish .....	42
<b>B</b>	<b>Configuring Remote JMX with Tomcat.....</b>	<b>43</b>
B 1	Remote JMX outside a Firewall without SSL .....	43
B 2	Remote JMX outside a Firewall with SSL .....	46
<b>C</b>	<b>Additional Property Settings .....</b>	<b>49</b>
<b>D</b>	<b>References.....</b>	<b>50</b>
D 1	Sun Java.....	50
D 2	Apache.....	50
D 3	Apache Ant .....	50
D 4	Tomcat.....	50
<b>9</b>	<b>Reporting Problems.....</b>	<b>51</b>

# 1 Introduction

The eHealth Framework (eHF) is a powerful platform for the development of health care solutions. InterComponentWare AG (ICW) has incorporated its extensive experience in developing and deploying solutions into the eHealth Framework, which represents the foundation of the Java Platform Enterprise Edition (Java EE) development at ICW. The ICW eHealth Framework consists of reusable software components, development tools, as well as architectural guidelines and conventions defining a full software-development and product lifecycle. From the perspective of a partner, the framework provides services and infrastructure capabilities for integrating applications within an eHF-based solution.

## 1.1 Document Overview

This Installation Manual provides support for installing eHealth Framework version 2.8. It describes the installation procedures for both the related software components and the eHF-2.8 itself.

The second chapter gives a short description of the hard- and software prerequisites that is mainly which kind of application server and database server is recommended when working with eHF.

Chapter three deals with the actual installation of eHF and the above mentioned environment or infrastructure like Java, Apache Ant, Maven, Apache HTTPD, and Apache Tomcat.

The fourth section describes the prerequisites of the database management system (DBMS) from Oracle.

Section five covers the configuration of eHF which is supported with a maven environment.

Chapter six explains how to set grants for Oracle in order to use it with eHF.

Chapter seven afterwards provides information on how to get initial data into the database.

Finally Chapter eight explains how to deploy and start eHF while the appendix chapters cover topics like alternatively set up eHF with Glassfish or remote configuration of tomcat with JMX as well as additional property setting for Oracle.

## 1.2 Document Conventions






Mark up	Explanation	Example
italic	Windows, Dialogs, Sections, Forms	Open the window:: <i>Options</i>
bold	Buttons, Fields, Tabs, Check boxes, Options, Drop-down-Menus	The Button <b>next</b> lets you navigate through the wizard.
bold and underlined	Link	Click on <u><a href="http://www.java.com">http://www.java.com</a></u> . The website will be opened.
bold and italic	Emphasis	You will find these information <b><i>only</i></b> on the vendors page.
Courier	Filenames and paths	<code>/usr/local/icw</code>
	Caution	<b>Warning:</b> Save your inputs!
	Note	<b>Note:</b> This view shows only data according to the installation process.
	Example	<b>Example:</b>
	Menu cascade	<b>Path:</b> Click <File   save as>
	Code	<b>Code:</b> <code>chmod 777 /usr/local/icw/</code>
<<Variable>>	Variable	The installation directory will be shown << INSTALL DIRECTORY>>
[CTRL+S]	Shortcut	Press [CTRL+S] to save the document
#	Command line prompt for a user with root rights (SUSE Linux)	<code># mkdir -p /usr/local/icw</code>
>	Command line prompt for a normal user (SUSE Linux)	<code>&gt; cd /usr/local/icw</code>

Table 1 Document Conventions

## 2 Installation Prerequisites

Please refer to the *Hardware/Software Prerequisites* document listed in the References chapter showing all necessary information on hardware and software prerequisites for using the eHealth Framework.

### 2.1 Installation Scope

First of all, the following installation procedure bases on SUSE Linux Enterprise Server 10 (SLES-10-i386-current) as the underlying operating system for the application server. Therefore all command line code snippets or entries in configuration files are described for the above mentioned operating system.

According to the installation files, shipped with the eHF release archive, the following components must be installed:

- Java (not shipped with the release archive)
- Apache Ant
- Apache Web Server
- Apache Tomcat or
- Sun Glassfish

As mentioned in the *Hardware/Software Prerequisites*, using eHF requires an Oracle 10g (at least 10.2.0.3 or a later patch level) with partitioning enabled.

Detailed information on how to install and configure the Oracle data base is provided in chapter four “Installing the Data Base Management System”.

### 2.2 Users and User Rights

First do not install eHF using the root user. Please create an eHF user to install and run the required services (Apache Tomcat and Apache HTTPD) later on. Nevertheless you should have root rights in order to get the HTTPD started, because the configuration is done via the privileged port 80. Afterwards this initial start you can configure a different user in order to let the HTTPD process run.

### 2.3 Recommended Installation Directory

It is recommended that you use `/usr/local/icw` as the installation directory for all the components. Of course you can change this directory name, but then you will have to make the corresponding changes whenever the installation directory is mentioned in the following procedure. The following chapter shows you at first, how to create the installation directory.

## 3 Installing the Components

This section first describes the installation files contained in the release archive. Afterwards the creation of the recommended installation directory follows. Sections 3.3 to 3.6 explain the installation of Java and the other components.

### 3.1 Installation Files

Since eHF 2.7, the release archive has a new structure. It separates the components into folders organizing the actual eHF binaries from infrastructure components and also from documentation. This eHF Installation-Manual is from now on part of the release archive as well. The content of the archive is listed below (the items in *italic* capital letters are folders):

- ehf-2.8.0-apache-ant-1.7.1-bin.zip
- digest.jar
- *DOCUMENTATION*
  - eHealthFramework-2.8-InstallationManual-en-us-1.0.pdf
- ehf-2.8.0-bin.zip
- ehf-2.8.0-oracle-template.dbt
- *INFRASTRUCTURE*
  - *APACHE-HTTPD*
    - ehf-2.8.0-apache\_2.2.8\_sles10\_64bit.tar.gz
    - ehf-2.8.0-apache\_2.2.8\_windows32.bit.zip
  - *APACHE-TOMCAT*
    - ehf-2.8.0-tomcat\_6.0.13.tar.gz
  - *GLASSFISH*
    - ehf-2.8.0-glassfish-v2ur2.tar.gz

## 3.2 Creating the Target Installation Directory

This section briefly describes how to create the installation folder and how to move the eHF archive to this directory.

It is assumed that you already have copied the release zip to a temporary folder - for example: `/tmp`. You are currently in this folder.



**Note:**

To create the installation directory `/usr/local/icw` you must have root rights. Ask your linux administrator how to get root rights or if he must provide the directory.

- 
1. For the case you got root rights, type the following.



**Code:**

```
# mkdir -p /usr/local/icw
```

---

The *parameter* forces the creation of the path mentioned above for the case the `/usr/local` directory does not exist.



**Note:**

As mentioned above for installing eHF on Linux/Unix it is recommended that the installation target folder is `/usr/local/icw`.

---

The next steps do not require root rights.

2. Extract the installation zip file from the temporary folder `/tmp` to `/usr/local/icw` and unpack by typing the following:



**Code:**

```
> tar xzfv ehf-2.8.0.tar.gz /usr/local/icw
```

The `tar` command with the parameters `xzfv` extracts the zip archive to the specified directory **and the folder structure as described above (see section 3.1) is created under `/usr/local/icw`.**

### 3.3 Installing Java 6

The eHealth Framework requires Sun JDK 1.6.0\_11 or a later patch release. You have to add the `/bin` folder to the `JAVA_HOME` entry in the `PATH` environment variable. The following code snippets show the appropriate command line entries.

As mentioned earlier, the JDK is not part of the release archive. For the case that you already have the correct version 1.6.0\_11 of the JDK installed, please make sure that the `bin` folder of the java installation is part of your `PATH` environment variable.

You can check this by typing `env` on the command line. A list of all specified environment variables is shown in the console window. If you do not have the `bin` directory added to your `PATH` variable, type the following command to do so.



**Code:**

```
> export PATH = $JAVA_HOME/bin:$PATH
```

For the case that you do not have installed any Java version on your system, follow the steps below to install it.

For more information visit the Sun Website at: <http://java.sun.com>.

Download the `jdk-1.6.0_11` from <http://java.sun.com> to the installation folder which is recommended for the whole installation procedure as the base folder: `/usr/local/icw`.

The name of the JDK binary depends on the target platform and is referred to as `jdk-1.6.0_11`.

To start the installation you have to change the mode of the downloaded file in order to make it executable.

---



**Note:**

You must have root rights for using a `chmod` command like described below!

---

Navigate to the newly created installation folder:

---



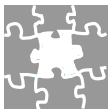
**Code:**

```
> cd /usr/local/icw
```

---

Then execute the following two commands. The second command is the actual execution of the JDK file which starts the installation.

---



**Code:**

```
# chmod 0744 jdk-1.6.0_11  
# ./jdk-1.6.0_11
```

---

When prompted, read the license agreement and type “yes” to accept it.

The Sun JDK 6 will now be installed on your system. After the installation, you can remove the file `jdk-1.6.0_11`.

## 3.4 Installing Apache Ant

Apache Ant is later used for generating the configuration file from the shipped configuration file templates for the Apache Web Server Installation.

Apache Ant is delivered as the packed file `apache-ant-1.7.1-bin.zip` in `/usr/local/icw`.

For further Information about Apache Ant visit the Apache website

<http://ant.apache.org/>



### Note:

The installation folder for the Apache components must *also* be `/usr/local/icw`

The steps and the code snippets below show also the root prompt `#` for convenience purpose only.

1. Extract `ehf-2.8.0-apache-ant-1.7.1-tar.gz` by typing the following:



### Code:

```
# tar xzvf ehf-2.8.0-apache-ant-1.7.1-tar.gz
```

2. Create a symbolic link to your Apache Ant installation for convenience.



### Code:

```
# ln -s apache-ant-1.7.1 apache-ant
```

2. Make the Ant binary is executable or accessible by typing:



### Code:

```
# chmod 0744 /usr/local/icw/apache-ant/bin/ant
```

3. Set the environment variables to complete the installation.

**Code:**

```
# export ANT_HOME=/usr/local/icw/apache-ant
# export PATH=$PATH:$ANT_HOME/bin
```

## 3.5 Installing Apache Tomcat

1. Change directory to installation folder:

**Code:**

```
> cd /usr/local/icw/
```

2. Extract the Apache Tomcat archive  
eHF-2.8.0-tomcat\_6.0.13.tar.gz by typing:

**Code:**

```
> tar xzvf infrastructure/apache-tomcat/eHF-2.8.0-
tomcat_6.0.13.tar.gz
```

The archive file will be unpacked and the directory `tomcat-6.0.13` will be created.

**Note:**

You need to edit the `startup.sh` /`bin` folder of the `apache-tomcat` folder if you are not using the standard paths. Therefore navigate to the directory `apache-tomcat/bin` and open the script with an editor. Change the following line according to your local path settings:

```
CATALINA_HOME=/usr/local/icw/tomcat
```

Note, that here the recommended installation directory path `/usr/local/icw` is used.

3. To start Tomcat, navigate to the bin directory of the tomcat installation folder and make sure that the file below is executable. Otherwise change the mode to 744 as described above. Then type:



**Code:**

```
> startup.sh
```

4. To test, if Tomcat has started correctly, navigate to the `logs` subfolder and type the following:



**Code:**

```
> tail -f catalina.out
```

The log file will appear. If the startup of Tomcat was successful, the last line will appear as follows:



**Code:**

```
server startup in [number] ms
```

## 3.6 Installing Apache HTTPD

1. Change directory to the installation folder:



**Code:**

```
> cd /usr/local/icw/infrastructure
```

2. Extract the Apache Web Server archive `eHF-2.8.0-apache_2.2.8_sles10_64bit.tar.gz` by typing:



**Code:**

```
> tar xzvf infrastructure/apache-httpd/eHF-2.8.0-  
apache_2.2.8_sles10_64bit.tar.gz
```

The archive file will be unpacked and the directory `apache_2.2.8` will be created.

3. Change into the created Apache folder and execute Apache Ant in order to generate the configuration file from the shipped configuration file templates.



**Note:**

To change the user and/or the group under which the Apache HTTPD process is running change the following section in the template file

```
apache_2.2.8/conf/httpd.conf.template
```

before you generate the final configuration with Apache Ant.

```
User daemon
```

```
Group daemon
```



**Code:**

```
> cd /usr/local/icw/ /apache_2.2.8/  
> ant
```

4. To start Apache Web Server you have to change directory to the `bin` folder of the Apache installation and then execute the `start` command.



**Code:**

```
> cd apache_2.2.8/bin  
> ./apachectl start
```



**Note:**

To verify, if the Apache Web Server is running you can use the `ps` command with the parameter list `aux` in combination with a piped `grep` command for `httpd`:

```
> ps -aux | grep httpd
```

(aux is the parameter list to show all currently running processes in BSD syntax)

---

## 4 Configuring the Oracle Data Base

This brief section covers the settings for the environment variable ORACLE\_DATA and which

### 4.1 Prerequisites for Oracle

The use of eHealth Framework with Oracle as database backend requires an Oracle 10g Release 2 Enterprise Edition (at least 10.2.0.3).



**Note:**

In order to make eHF 2.8 work properly with Oracle, you will need to have installed Oracle 10g Release 2 Enterprise Edition (or newer) with partitioning enabled. For your convenience, a configuration template showing a supported configuration for the Oracle Database Creation Assistant (dbca) is supplied as the file oracle\_database\_template.dbt.

Before using this template, it is absolutely crucial that the environment variable ORACLE\_DATA on which the database template depends is set. To do so, please export the path to your oradata directory (e.g. /vol01/app/oracle/oradata) from a shell:

```
export ORACLE_DATA="/vol01/app/oracle/oradata"
```

```
export PATH="$ORACLE_DATA:$PATH"
```



**Caution:**

If using the template provided, please be aware that during the installation of Oracle you have to manually change the value for table space (initialized by the template) with the value USERS to EHFUSERS.

Please refer to Oracle's documentation on how to install Oracle 10g using a template with the Database Creation Assistant for further information.

## 5 Configuring the eHealthFramework

As mentioned in section 3.1 the installation and configuration of eHealth Framework continues here after installing the necessary components.

Now the configuration of eHealth Framework is described in the following sections. To configure the framework, follow the steps below.

1. Navigate to `/usr/local/icw` and unpack the file `ehf-2.8.0-bin.zip`:



**Code:**

```
> unzip ehf-2.8.0-bin.zip
```

---

The zip file will be unpacked to the newly created directory `eHF-2.8.0-bin`. This is the base or installation folder for the eHF application.

2. Afterwards change directory to `/usr/local/icw/eHF-2.8.0-bin`

Configuring eHF 2.8.0 demands the modification of several properties files. The central properties file is named `configuration.properties`.

---



**Note:**

Please note that all the settings in the above mentioned *properties files* can be overwritten by including the properties of the same name in the

`configuration.properties` file.

A template for this file (named `configuration.properties.template`) is provided for your convenience. This file includes all required configuration files and may be used to overwrite properties in the included files.

---

The following property files are included in the template. You can either edit the template or the single property files. The list below shows a brief overview and description of the files:

1. `configuration.deployment.properties`:  
This file contains settings which are specific to the database on the target machine as well as to a possible proxy environment.
2. `configuration.product.properties`:  
This file contains product- and environment global settings (shared among all instances of the product).

3. `configuration.product.instance.properties`:  
This configuration file contains information specific to the instance of the product.
4. `configuration.password.properties`:  
This file contains password hashes and temporary passwords for the Oracle users.

The following sections cover the settings in the property files listed here.

## 5.1 Proxy- and Database Property Setting for eHealth Framework

The section describes the settings of the proxy- and database specific properties in the `configuration.deployment.properties` file.

1. Open this file on the root level of the `/usr/local/icw/ehf-2.8.0-bin` folder.
2. You must adapt the values of the connection settings and properties according to the type of database installed on your target system. To configure the environment for an HSQLDB based system, use the settings which are listed under the sections:



**Code:**

```
# connection settings (HSQLDB) and  
# audit specific settings (HSQLDB).
```

---

To configure the environment for an Oracle-based system, use the settings under `# connection settings (ORACLE)` and `# audit specific settings (ORACLE)`.

**Note:**

If you were using the provided template during the database setup, please be aware that you have to change the value for table space manually (initialized by the template with the value `USERS`) to a different value, if you want eHF to use a specific table space like e.g. `'EHFUSERS'`. The setting you choose here also needs to be configured in the `configuration.deployment.properties` (see below).

If you configure a separate database connection for the audit module, make sure that the database instance is set up in the same way as for the standard modules (i.e. by using the provided database template). You might want to change the SID of the audit database instance to a different value (e.g. `ehfaudit`) to indicate what this instance is used for.

The configuration properties described in the following list must be adapted to the appropriate values of the target system.

Some short descriptions of the individual properties as well as some example configuration files are given below.

Standard eHF module database connection specific properties (to be set in the `configuration.deployment.properties` file):

- **connection.driver.lib.path:** This property specifies the path to the database driver lib jar file.
- **connection.dialect:** This property must be configured for productive installations to `'org.hibernate.dialect.Oracle9Dialect'`.

The HSQLDB specific settings should only be used for development purposes.

- **connection.driver:** The class name of the database driver.
- **connection.url:** This property configures the database connection URL and should be configured for the database instance to use. For installations against an Oracle data base, the connection URL must match the following syntax:

```
jdbc:oracle:thin:@<SERVICE_NAME>
```

The service name must be identical to the Oracle service name as specified in the `tnsnames.ora` file for the Oracle Instance.

- **connection.tablespace:** The value of this property must be set to the tablespace name which you have configured in the database for the eHF schemata ( e.g. `ehfusers`).

- **connection.validation.query:** This property is used to verify the connection to the database server. A query is sent to check whether a proxy is blocking the connection.
- **connection.context:** This property can either be `oracle` or `hsqldb`. Using this property native database handling is enabled for the applications.
- **connection.datasource.maxConnectionAge:** `maxAge` is the max time in milliseconds a connection should be used. The default value is set to *120000 ms*.

### 5.1.1 Audit Specific Database Connection Properties:

The audit module eHF allows the configuration of a database connection separate from the standard eHF module database connection. The usage of a separate database instance for audit events might be desired to reduce the load on the standard eHF database instance, since logging of audit events is I/O intensive. Apart from this, an audit specific database instance might be desired for security reasons, to separate audit information physically from other application data. However, if no separate database instance is available for the audit module, the values from the standard database connection configuration can also be applied for the audit specific configuration.

- **audit.connection.driver.lib.path:** This property specifies the path to the database driver.
- **audit.connection.dialect:** This property must be configured for productive installations to

```
org.hibernate.dialect.Oracle10Dialect.
```

The HSQLDB specific settings should only be used for development purposes.

- **audit.connection.driver:** The class name of the database driver.
- **audit.connection.url:** This property configures the database connection URL and should be configured for the database instance to use. For installations against an Oracle Real Application Cluster the connection URL must match the following syntax:

```
jdbc:oracle:oci:@<SERVICE_NAME>
```

The service name must be identical to the Oracle service name as specified in the `tnsnames.ora` file for the Oracle Instance.

- **audit.connection.tablespace:** This property must be configured to the tablespace which you have configured in the database for the audit schema (e.g. ehfusers).
- **audit.connection.validation.query:** This property is used to verify the connection to the database server. A query is sent to check whether a proxy is blocking the connection.

For further database related properties and their configuration in the `configuration.deployment.properties` file please refer to the appendix.

## 5.2 Setting Product-global Configuration Properties

The `configuration.product.properties` file contains the following product- and environment global settings (shared among all instances of the product).

- **deploy.artifact.name:** The name for the main artifact (eHF-2.8.0)
- **deploy.artifact.type:** The type of the artifact (a Web Archive file)
- **database.bootstrap.pattern:** Pattern of the input file(s) for the `database:bootstrap` goal.
- **database.bootstrap.policy:** Policy file dedicated to the `database:bootstrap` goal. Defaults to '**assembly/deploy.policy**'.
- **database.import.content.pattern:** Pattern of the input file(s) for the `database:import-content` goal. This property is not set by default. Please note that empty patterns are interpreted by the plugin as empty and do not result in build failures.
- **database.import.content.policy:** Policy file dedicated to the `database:import-content` goal.
- **database.import.policy:** Pattern of the input file(s) for additional import procedures.
- **database.import.style:** Format style for additional imports.

## 5.3 Settings Product Instance Specific Properties

The `configuration.product.instance.properties` file contains all necessary properties.

- **`product.instanceidentifier.root.prefix`:** Every instance of the eHF product must be identified via a dedicated root prefix. This String identifies a system as the owner of a specific document (OID). This identifier is assigned to an individual installation by ICW. For productive installation request an instance identifier via the ICW support.
- **`adapter.context`:** This property is intended to switch between either the internal or external realization of the required adapters. The internal realization only provides limited content while the external realization accesses external services.

Upgrade and encryption related properties:

- **`upgrade.mode`:** Currently a 'SQL' and 'FILE' mode have to be distinguished. The upgrade steps are meant to execute the SQL against the database in 'SQL' mode. The 'FILE' mode enables to only write the SQL statements to a file (e.g. for testing and analysis purposes.).
- **`upgrade.path`:** A product may be delivered with many different source versions for an upgrade. With this property the folder can be identified, which matches the installation at hand. The folder must exist in your unpacked distribution directory. E.g. `upgrade.path=upgrade/ehf-5.5-eHF-6.0`
- **`upgrade.encryption.disable`:** This property indicates, whether the database is encrypted with database encryption. If so this should be set to "false", otherwise "true".
- **`encryption.mode`:** Currently a 'SQL' and 'FILE' mode have to be distinguished. The upgrade steps are meant to execute the SQL against the database in 'SQL' mode. The 'FILE' mode enables to only write the SQL statements to a file (e.g. for testing and analysis purposes.).
- **`encryption.tasks`:** Ordered list of scripts to execute when running the encryption:all goal.
- **`encryption.path`:** Relative path to location in bin directory of encryption scripts, usually 'encryption/app-version-version'

## 5.4 Setting Password Specific Configuration Properties

They must be configured in the `configuration.password.properties` file:

- **`user.regadmin.password.hash`**: This property specifies a hash value for the regadmin user's password.



### Note:

The hash value can be calculated via the `digest.jar` file provided with the installation medium. In order to generate a hashed value for your password, navigate to the installation directory for `digest.jar` and execute the following command from a shell or dos prompt:

```
> java -jar digest.jar <<password>>
```

where `<<password>>` is the password you want to get hashed.

In order to create secure passwords we recommend using the freeware tool `pw_generate.exe`. This tool is part of `rpgen` which is available from <http://www.paehl.de/rpgen.zip>.

- **`user.sysadmin.password.hash`**: This property specifies the hash value for the sysadmin user password. As is also the case with `user.regadmin.password.hash`, this hash value can be calculated by using the `digest.jar` file provided with the installation medium.
- **`user.devicemanager.password.hash`**: This property specifies the hash value for the devicemanager user's password. This hash value can be calculated by using the `digest.jar` file provided with the installation medium.
- **`user.stsclient.password.hash`**: This property specifies the hash value for the stsclient user's password. This hash value can be calculated by using the `digest.jar` file provided with the installation medium.

The `configuration.password.properties` file also contains the connection passwords for each module. Please make sure to change the individual passwords if you install this software in a productive environment.

```
connection.audit.pass=initinit  
connection.authr.pass=initinit  
connection.codesystem.pass=initinit  
connection.document.pass=initinit  
connection.record-admin.pass=initinit
```

```
connection.record-medical.pass=initinit  
connection.usermgnt.pass=initinit  
connection.composition.pass=initinit  
connection.reference.pass=initinit  
connection.record.pass=initinit
```

- **upgrade.datapump.password:** This parameter is the password for an Oracle Datapump file. Datapump is required when an upgrade step exports/imports encrypted data
- **connection.username:** For installations against Oracle this property must be configured to `ehfuser`.
- **connection.password:** This property must be configured with the password for the `ehfuser`.
- **connection.system.username:** For installations against Oracle this property must be configured to `system`.
- **connection.system.password:** This property must be configured with the password for the Oracle system user.
- **audit.connection.username:** For installations against Oracle this property must be configured to `ehfuser`.
- **audit.connection.password:** This property must be configured with the password for the `ehfuser`
- **audit.connection.system.username:** For installations against Oracle this property must be configured to `system`.
- **audit.connection.system.password:** This property must be configured with the password for the Oracle system user.

## 5.5 Setting Certificate Validation Specific Configuration Properties

They have to be configured in the `configuration.certificateValidation.properties` file.

- **certificateValidation.crlRepository.delay:** This property is specifying the the time interval (in milliseconds) which is used by the `PollingCrlRepositoryImpl` to refresh the CRL repository. The default value for this property is `3600000`.

- **certificateValidation.crlRepository.directory:** This property specifies the local directory which is used to store the CRLs in. The `certificateValidation.crlRepository.directory` has to be set to `/usr/local/icw/tomcat/conf/crls/`
- **certificateValidation.trustedCertificates.keystore.fileName:** This property is used for configuring the Java KeyStore which holds the trusted certificates and defaults to the value `test.keystore`.
- **certificateValidation.trustedCertificates.keystore.password:** This property stores the password which protects the KeyStore.
- **certificateValidation.trustedCertificates.useShutdownHook:** The hook is used to make any changes, done by the application persistent in the KeyStore. The default value for this property is set to `false`.

## 5.6 Setting Token Service Specific Configuration Properties

They have to be configured in the `configuration.tokenservice.properties` file.

Configuration parameters for the ehf Security Token Service (STS).

The STS is a Web Service conforming to the WS-Trust specification that issues security tokens that contain SAML assertions. These tokens are accepted by the Token Filter that can be used to protect arbitrary HTTP resources.

- **tokenservice.issuer:** This property holds the value of the `<Issuer>` element of created assertions. It should be an identifier for the STS system, preferably its URL.
- **tokenservice.signAssertions:** This property indicates, whether the STS signs the issued assertions. This is mandatory in production, only set to `false` for testing purposes. By default, the property is set to `true`.
- **tokenservice.tokenValiditySeconds:** This property specifies the validity (in seconds) that the STS assignments to the issued assertions. This validity constraint shall be honored by token-consuming service endpoints. By default, the property is set to `30`.
- **tokenservice.jaas.realm:** This property is identifying the JAAS realm used by the STS for authentication. The realm is for example configured in tomcat's `conf/catalina.login` JAAS configuration file. The default value for this property is `ehf`.

- **tokenservice.keystore.filename:** This property specifies the spring resource containing the keystore to use for signing. The default value applies to the example keystore that is contained in the tokenservice module.
- **tokenservice.keystore.password:** This property specifies the spring resource containing the keystore password of the contained key to use for signing. The default value applies to the example keystore that is contained in the tokenservice module.
- **tokenservice.key.alias:** This property specifies the spring resource containing the alias of the contained key to use for signing. The default value applies to the example keystore that is contained in the tokenservice module.
- **tokenservice.key.password:** This property specifies the spring resource containing the password for the signing key. The default value applies to the example keystore that is contained in the tokenservice module.

## 5.7 Setting Token Filter Specific Configuration Properties

They have to be configured in the `configuration.tokenfilter.properties` file. It contains configuration parameters for the ehf Security Token Filter. This login filter accepts SAML tokens that are received within the HTTP header as authentication credential.

- **tokenfilter.maxAgeSeconds:** This property specifies the maximum age (in seconds) that tokens are allowed to have. The age is calculated from the `IssueInstant` attribute of the assertion which contains the token's creation time. The age check is only performed, if this property is set to a value larger than null. The property is set to 5 by default.
- **tokenfilter.clockSkewSeconds:** This property is used to compensate for time differences between the issuing system and the validating system. Its value is used as the number of seconds of tolerance in time comparisons, so that e.g. a token may be `<clockSkewSeconds>` older than the maximum age allowed without being rejected. The default value used for this property is 3.
- **tokenfilter.expectedAudience:** Configure this with the SAML `entityID` that identifies this endpoint and that is expected as the value of the Audience element in the SAML token. The term `entityID` comes from the SAML specification and describes an identifier for a system entity that provides, consumes or otherwise participates in SAML-based services.

- **tokenfilter.requiresAudienceRestriction:** If this property is set to `true`, the filter will reject tokens that do not contain an audience restriction, else such tokens will not be rejected. The default value for this property is set to `false`.
- **tokenfilter.requiresSignature:** This property determines, whether the filter requires signatures. If it is set to `true` (which also is the default value for this property), unsigned tokens are rejected.
- **tokenfilter.keystore.filename:** This property specifies the spring resource containing the keystore that contains the public key used to validate received tokens. The default value applies to the example keystore that is contained in the `tokenservice` module.
- **tokenfilter.keystore.password:** This property specifies the keystore's password of the contained certificate that is used for validation. The default value applies to the example keystore that is contained in the `tokenservice` module.
- **tokenfilter.key.alias:** This property specifies the alias of the contained certificate that is used for validation. The default value applies to the example keystore that is contained in the `tokenservice` module.
- **tokenfilter.verbose.errors:** This property is used to specify whether to include detailed error messages into the response when token validation fails or not. This can be very helpful for debugging clients and is needed for testing different error conditions but not recommended in production, as it also gives hints to a possible attacker. (default: `false`)
- **tokenfilter.jaas.realm:** This property determines the JAAS realm used by the token filter for authentication. The realm is configured for example in tomcat's `conf/catalina.login` JAAS configuration file. The default value used for this property is `ehf`.

Further configurations have to be made to the following properties:

- `configuration.properties`
- `configuration.deployment.properties`
- `configuration.product.properties`
- `configuration.product.instance.properties`
- `configuration.password.properties`
- `configuration.certificateValidation.properties`
- `configuration.tokenFilter.properties`

- `configuration.tokenService.properties`



### Note: Productive Environments

For productive environments you also have to change the passwords for `sysadmin` and `regadmin`. Please navigate to the folder

```
/usr/local/icw/ehf-2.8-bin/assembly/META-INF/ehf-assembly/bootstrap/
```

and edit `usermgnt-data.xml`. The following lines demonstrate the section that has to be changed for the user `regadmin`, for the others the same lines apply



### Caution:

These changes below have to take place **before** the deployment of eHF starts.

Otherwise the passwords won't be changed.



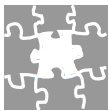
### Code:

```
<!-- password = regadmin (SHA1 digest) -->
<password>
  <hashedValue>enter_hashed_password_here</hashedValue>
</password>
```

## 5.8 Deployment Preparations

The deployment process and its necessary preparations is now supported by ant. As mentioned above the steps to be made are now ant calls and no longer maven goals.

1. Navigate to the `/usr/local/icw/eHF-2.8-bin` folder. When the archive was extracted an install subfolder was created.
2. To configure the eHF application with the values configured in the `configuration.properties` execute the command:



**Code:**

```
> ant -f install/install.xml configure:all
```

- 
3. Within the application's installation folder at `/usr/local/icw/eHF-2.8-bin/` execute the following command to set-up the database:



**Caution:**

Creating new schema tables will cause existing schemata in the target database to be dropped and information to be lost! Make sure you have the correct DB configuration! Only execute this step if this is the initial configuration.



**Code:**

```
ant -f install/install.xml database:prepare
```

---

You will see a lot of output from the SQL scripts. You can safely ignore the warning messages, if they are caused by dropping not yet existing tables in the database's schemata.

At the end, `BUILD SUCCESSFUL` is displayed.

## 6 Creating Appropriate Grants for Oracle

In contrast to the former process the creation of Oracle grants has been simplified. Nevertheless you must e.g. create an Oracle user with root rights for setting up the Oracle data base and give the user appropriate read/write permissions to the Oracle schemata.

1. The previous ant call `database:prepare` has created the following directory structure in the assembly directory:

```
<<ASSEMBLY>>/ .install/tasks/database
```

The directory contains the following files:

```
create-datasource-user.sql  
grant-schema-permissions.sql  
create-functions.sql  
drop-functions.sql
```

2. In order to set the grants in Oracle you have to become an Oracle user by typing:



**Code:**

```
> cd /usr/local/icw/ehf-2.8-bin/.install/task/database  
> su -oracle
```

---

**Note:**

For the modules that are not generated, or have a different schema name than the domain name, add the following fragment file to:

```
<<MODULE>>/src/main/config/merge/assembly/db/oracle10/  
schema/permissions.fragment
```

**Example Settings:**

```
prompt ----- Access <SCHEMA_NAME> Scheme -----;  
  
define schemaName=<<SCHEMA_NAME>>;  
define schemaPass=@@connection.<module_name>.pass@@;  
  
connect &schemaName/&schemaPass@&SID;  
  
@@create-functions.sql  
execute grantTo('&schemaName', '&datasourceUserName');  
@@drop-functions.sql
```

<<SCHEMA\_NAME>> must be replaced with appropriate schema name. eg. EHF\_USERMGNT, and <<MODULE\_NAME>> must be replaced with appropriate module name, e.g. usermgnt.

Then generate the final make grants script file by executing the  
maven ehf:release task.

This would place a file in the following folder:

```
<APPLICATION>-SNAPSHOT-bin.zip/database/scripts
```

Where < APPLICATION > could be for example eHF-SNAPSHOT-bin.zip. This file contains all the fragments from other modules, with the passwords substituted from the property file.

---

### 3. Execute the SQL script using SQL\*PLUS by typing:

**Code:**

```
sqlplus /nolog
```

---

You will see an output similar to the following:

---



**Example:**

SQL\*Plus: Release 10.2.0.3.0 - Production on Tue Jul 11 12:09:50 2006

Copyright (c) 1982, 2005, Oracle. All rights reserved.

SQL>

---

4. In order to grant permissions an Oracle user must be created first. This done by executing the appropriate script:
- 



**Code:**

```
SQL> @create-databasesource-user.sql
```

---

After starting the script you will be prompted for entering a username and a password.

---



**Note:**

The values for this prompt must match the property values you had already provided in the `configuration.deployment.properties` file for:

```
connection.username
```

```
connection.password
```

```
audit.connection.username
```

```
audit.connection.password
```

---

5. Then the schema permissions will be set by executing the `grant-schema-permissions.sql` script:
- 



**Code:**

```
SQL> @grant-schema-permissions.sql
```

---



**Note:**

The maven `ehf:release` task, expects a property file (`configuration.properties`, or `configuration.password.properties` in case of eHF) with all the passwords. The parameters are of the form:

```
connection.<module_name>.pass=password
```

---

Additional required parameters are:



**Example:**

```
main.database.username=ehfuser  
main.database.password=ehfuser  
database.oracle.sid=EHFDB  
connection.system.username=system  
connection.system.password=manager
```

- 
6. Execute the command `quit` in order to exit `sqlplus`.

## 7 Creating the Initial Values in the Database

With the creation of initial database values through the database bootstrap ant call below data like login policies, system administrator data or user roles and permission settings are imported. Further the code systems will be imported as well.

In order to make the ant call, navigate to `/usr/local/bin/ehf-2.8-bin` and execute the following command:



**Code:**

```
> ant -f install/install.xml database:bootstrap
```

---

At the end, **BUILD SUCCESSFUL** is displayed.

### 7.1 Importing Additional Data (optional)

Additional data can be imported with the ant call below. It creates demo user and sample data in database. The `database:import` call also imports (calls) test data and further demo content data. So the explicit ant calls for test data and further content data is no longer applicable.



**Note:**

This ant call can usually be executed only *after* a database bootstrap (in addition).

```
> ant -f install/install.xml database:import
```

```
-Ddatabase.import.content.pattern=classpath:/META-INF/ehf-assembly-import-context.xml
```

---

As a next step, deploy the application.

1. Navigate to the directory `/usr/local/icw/ehf-2.8-bin`
2. Copy the file `ehf.war` to the subfolder `webapps` of the tomcat directory:

## 8 Deploying and Starting the Application



**Code:**

```
> cp ehf.war /usr/local/icw/tomcat_6.0.13/webapps
```

---

The application will be automatically deployed by Tomcat.

### 8.1 Starting the application

1. Direct your browser to the URL <https://<ehf server name>/ehf/services>.

The login pop-up opens.

2. Log in as:

**User name:** regadmin

**Password:** regadmin

Those are the values, if you have configured the password.properties file as described above. The second case would be:

**User Name:** <<demo-user>>

**Password:** <<demo-user>>

Where <<demo-user>> is a user you registered for testing or demo purpose.

The eHF Web Services Page opens.

## A Setting up eHealth Framework with Glassfish

The Glassfish v2ur2 application server from SUN is also supported for eHF 2.8.

### A 1 Installing Glassfish

1. Download the Linux version of Glassfish (*glassfish-installer-v2ur2-b04-linux.jar*) from <http://glassfish.dev.java.net>. The version supported is: V2 UR2.
2. Extract the provided eHF-2.8.0-glassfish-v2ur2.zip
3. Open setup-glassfish.sh and configure:
  - a. GLASSFISH\_INSTALL\_JAR: The Installer Jar name of the Glassfish version downloaded in the previous step.
  - b. INSTALL\_HOME / GLASSFISH\_HOME: The path glassfish will be installed
  - c. X\_PORT: Any port configuration that should be changed. Note: The ADMIN\_GUI\_PORT should be accessible.
  - d. DOMAIN\_NAME: The domain to create. It is recommended that eHF is installed in a separate domain.
  - e. CERT\_ALIAS: Glassfish is configured with SSL. This is the alias of the keystore used.
  - f. CRL\_NAME: eHF uses a CRL. This is the name of that CRL.



**Code:**

```
INSTALL_HOME=/usr/local/icw
export INSTALL_HOME

GLASSFISH_HOME=$INSTALL_HOME/glassfish
export GLASSFISH_HOME

PATH=$GLASSFISH_HOME/bin:$PATH
export PATH

GLASSFISH_INSTALL_JAR=glassfish-installer-v2ur2-b04-
sunos_x86.jar

HTTP_PORT=8080
HTTP_SSL_PORT=8081

JMS_PORT=7676

IIOP_PORT=3700
IIOP_SSL_PORT=3820
IIOP_MUTUALAUTH_PORT=3920

JMX_PORT=9991

ADMIN_GUI_PORT=5666

DOMAIN_NAME=ehf-domain

CERT_ALIAS=tomcat

CRL_NAME=115c.der.crl
```

---

4. The following files are needed for the Glassfish configuration:

- tomcat.keystore: Keystore for SSL configuration
- tomcat.truststore: Truststore for SSL configuration
- catalina.login : The JAAS policy for eHF-Authentication
- default.policy: The security policy for eHF

- resolver.config: The resolver for client certificates for eHF
- 115c.der.crl: (may have another name): The CRL for eHF
- tomcat-ajp.jar: The Tomcat-AJP connector for connection to Apache. Tomcat 5.5.16 must be used
- commons-modeler-2.0.1.jar: The commons-modeler required by the Tomcat-AJP connector. version 2.0.1 must be used.
- commons-logging.jar: The commons-logging required by the Tomcat-AJP connector. version 1.1 must be used.
- log4j.xml: The log4j configuration required by eHF
- log4j-1.2.15.jar: The log4j library required by eHF



**Note:**

Only Tomcat 5.5.16 can be used with Glassfish. This is because Glassfish is based on branched code from Tomcat 5.5.

5. On the SLES machine, create an <<INSTALLATION\_HOME>> directory e.g. /usr/local/icw/glassfish-install and copy the files from the previous step and also all the glassfish scripts to this <<INSTALLATION\_HOME>>.
6. Make sure the setup-glassfish.sh script is executable:



**Code:**

```
> chmod +x setup-glassfish.sh
```

7. Make sure Java is installed on the machine. If not, please install at Java 1.5 as this is needed (see chapter 3.2)
8. Start the installation of glassfish (you will need to accept the license agreement):



**Code:**

```
> ./setup-glassfish.sh install
```



**Note:**

The installation file for SLES10, `glassfish-installer-v2ur2-b04-linux.jar`, opens an X-window displaying the license agreements. You need to accept this before continuing.

---

9. Create the glassfish domain that eHF will run in:

---



**Code:**

```
> ./setup-glassfish.sh create-domain
```

---



**Note:**

During this step you will be prompted for password and master password. You need to note these as you will need them to make changes in Glassfish. Also, be sure to use change it for the masterpassword in order to use the keystore.

---

10. Start Glassfish now for the next configuration steps

---



**Code:**

```
> ./setup-glassfish.sh start
```

---

11. Configure the required JVM settings.

---



**Code:**

```
> ./setup-glassfish.sh configure-jvm
```

---

12. Configure SSL settings for the glassfish domain that eHF will run in (this sets up the require SSL port, disables HTTP port in Glassfish):

**Code:**

```
> ./setup-glassfish.sh configure-ssl
```

---

13. Configure Log4J settings for the glassfish domain that eHF will run in:

**Code:**

```
> ./setup-glassfish.sh configure-logging
```

---

14. Configure CRL settings for the glassfish domain that eHF will run in:

**Code:**

```
> ./setup-glassfish.sh configure-crls
```

---

If any of the installation steps fail, please note the problem, try to fix it (it may be a missing file needed), and retry the step.

If you encounter an issue on the domain, you can remove the domain using `./setup-glassfish.sh remove` and start again at the create domain step. Otherwise, if the issue continues to reoccur, then you may need to delete the <<GLASSFISH\_HOME>> directory and begin again.

## A 2 Starting and Stopping Glassfish

There are several ways you can start and stop glassfish:

1. Using the `setup-glassfish.sh` script:

**Code:**

```
> ./setup-glassfish.sh start
```

```
> ./setup-glassfish.sh stop
```

2. Using the Glassfish asadmin tool in <<GLASSFISH\_HOME>>/bin (For this you need to know the domain name configured by DOMAIN\_NAME in setup-glassfish.sh):



**Code:**

```
> ./setup-glassfish.sh start-domain <<DOMAIN-NAME>>  
> ./setup-glassfish.sh stop-domain <<DOMAIN-NAME>>
```

## A 3 Deploying and Undeploying eHF in Glassfish

1. Using the Administration Console.
  - a. Open `http://<<machine-name>>:$ADMIN_GUI_PORT` in a browser.  
Note: ADMIN\_GUI\_PORT was configured in `setup-glassfish.sh`
  - b. On the left menu select: Applications-> Web Applications
  - c. Click on the 'Deploy' button
  - d. Select "Web Application (.war)" for Type
  - e. Browse for the location of the build ehf.war file.
  - f. Make sure the "Application Name": is ehf.
  - g. Select 'OK'

## A 4 Logging and finding issues in Glassfish

The glassfish log is contained at:

```
GLASSFISH_HOME/domains/DOMAIN_NAME/logs/server.log
```

To increase log levels in glassfish:

1. Open `http://<<machine-name>>:$ADMIN_GUI_PORT` in a browser.
2. Select the following: Application -> Logging -> Log Levels
3. Configure the appropriate level for the module and select 'Save'.
4. Stop and restart the Glassfish domain, see \*Starting and Stopping Glassfish\*

## B Configuring Remote JMX with Tomcat

Due to issues with the way JMX works, it is not possible to use the standard Remote JMX outside a firewall. The Tomcat delivered with eHF provides two approaches:

- Remote JMX access outside a firewall using Authentication and Authorization
- Remote JMX access outside a firewall using Authentication and Authorization using SSL

### B 1 Remote JMX outside a Firewall without SSL

1. Configure both a password and an access file.

- 1.1. The Password file e.g. `password.properties` contains a *username* and a *password* separated by a blank. You can specify several username/password combinations on separate lines:



**Code:**

```
monitorRole password1  
controlRole password2
```

- 1.2. The access file e.g. `access.properties` describes the role the username can have. There are two possibilities: *readwrite* and *readonly*. You must specify a role for every username provided in your password file. Again *username* and *access roles* are separated by a blank:



**Code:**

```
monitorRole readonly  
controlRole readwrite
```

2. Configure Tomcat as follows in the startup script (linux settings shown):

**Code:**

```
# Setup JMX Connector using Java Dynamic Agent

JAVA_OPTS="$JAVA_OPTS -Dcom.icw.agent.port=9091 -
javaagent:$CATALINA_HOME/lib/jmx-agent.jar"

# Authentication and Authorisation settings

JAVA_OPTS="$JAVA_OPTS -
Djmx.remote.password.file=$CATALINA_HOME/conf/password.properties"

JAVA_OPTS="$JAVA_OPTS -
Djmx.remote.access.file=$CATALINA_HOME/conf/access.properties"
```

The `jmx.remote.password.file` refers to the password file configured in the previous step, and the `jmx.remote.access.file` refers to the access file also configured in the previous step.

3. Start Tomcat and in the catalina output file e.g. `{{catalina.out}}` you will find the following output:

**Code:**

```
Create RMI registry on port 9091

Get the platform's MBean server

Initialize the environment map

Create an RMI connector server

Adding JMX security

Start the RMI connector server on port 9091
```

The important information is the string `Start the RMI connector server on port 9091`. This contains the port used to register the JMX Connector service. In this case the information needed is `9091`.

4. Start JConsole (both Java 5 and Java 6 versions work). No special parameters are needed:

**Code:**

```
jconsole
```

5. In this example, the Java 5 version of JConsole was used. In the Connection box, select the "Remote" tab and enter the correct information and the correct username and password configured in the files specified by `jmx.remote.password.file` and `jmx.remote.access.file`.



## B 2 Remote JMX outside a Firewall with SSL

1. Configure Tomcat as follows in the startup script (linux settings shown):



### Code:

---

```
# Setup JMX Connector using Java Dynamic Agent

JAVA_OPTS="$JAVA_OPTS -Dcom.icw.agent.port=9091 -
javaagent:$CATALINA_HOME/lib/jmx-ssl-agent.jar"

# Authentication and Authorisation settings

JAVA_OPTS="$JAVA_OPTS -
Djmx.remote.password.file=$CATALINA_HOME/conf/password.properties"

JAVA_OPTS="$JAVA_OPTS -
Djmx.remote.access.file=$CATALINA_HOME/conf/access.properties"

# SSL settings

JAVA_OPTS="$JAVA_OPTS -
Djavax.net.ssl.keyStore=$CATALINA_HOME/conf/tomcat.keystore"

JAVA_OPTS="$JAVA_OPTS -Djavax.net.ssl.keyStorePassword=changeit"

JAVA_OPTS="$JAVA_OPTS -
Djavax.net.ssl.trustStore=$CATALINA_HOME/conf/tomcat.truststore"

JAVA_OPTS="$JAVA_OPTS -Djavax.net.ssl.trustStorePassword=changeit"
```

---

2. Start Tomcat and in the catalina output file e.g. catalina.out you will find the following output:

**Code:**

Create RMI registry on port 9091

Get the platform's MBean server

Initialize the environment map

Create an RMI connector server

```
creating server with URL: service:jmx:rmi://ehf-deploy-0009:9091/jndi/rmi://server01:9091/jmxrmi
```

Adding JMX Authentication/Authorisation security

Start the RMI connector server on port 9091

```
Server started at: service:jmx:rmi://ehf-deploy-0009:9091/jndi/rmi://ehf-deploy-0009:9091/jmxrmi
```

```
Waiting on main [id=1]
```

The important information is the string `Server started at: service:jmx:rmi://ehf-deploy-0009:9091/jndi/rmi://server01:9091/jmxrmi`. This contains the machine name and port used to register the JMX Connector service. In this case the information needed is `server01:9091`

- Using SSL only works with JConsole version provided with Java 6. (The Java 6 version of JConsole provides SSL functionality that work for this solution). You must start JConsole with the following:

**Code:**

```
jconsole -J-Djavax.net.ssl.keyStore="%CERT_HOME%\tomcat.keystore" -J-Djavax.net.ssl.keyStorePassword=changeit \
```

```
-J-Djavax.net.ssl.trustStore="%CERT_HOME%\tomcat.truststore" -J-Djavax.net.ssl.trustStorePassword=changeit
```

`%CERT_HOME%` refers to the path containing the *keystore* and *truststore*.



**Note:**

The keystore and truststore used with JConsole should be the same versions configured with Tomcat e.g.

```
javax.net.ssl.keyStore=$CATALINA_HOME\conf\tomcat.keystore"
```

```
javax.net.ssl.trustStore=$CATALINA_HOME\conf\tomcat.truststore"
```

4. In the Connection box, select "Remote Process" and enter the correct information e.g. server01:9091 and the correct username and password configured in the files specified by `jmx.remote.password.file` and `jmx.remote.access.file`. For further information refer to the previous section.



**Caution:**

The full service:JMX does \*NOT\* work in JConsole for this solution.

## C Additional Property Settings

- **connection.multiple.databases:** This indicates whether there is more than one database SID installed on the Oracle RDBMS. Valid values are true or false.
- **database.type:** This property is specifying the database type for the upgrade. Currently, only the value oracle is supported.
- **database.host:** This property specifies the remote host where the database is running.
- **database.oracle.sid:** The Oracle system identifier (SID) which designates the target database instance. This property is needed for establishing a connection.
- **database.oracle.base:** The root directory of your Oracle installation. On Unix systems, this is usually /app/oracle or /u01/app/oracle. If you are unsure about the exact location, just check to which directory the \$ORACLE\_BASE environment variable is set.
- **database.oracle.home:** The directory path where the Oracle client software is installed. If you are unsure about the exact location, just check to which directory the \$ORACLE\_HOME environment variable is set.
- **database.oracle.nls.lang:** The national language, territory and character set used for displaying messages, masks and sorting sequences etc. in Oracle.

Other infrastructure components related properties to be configured in the configuration.deployment.properties file:

- **adapter.mmi.pzn.port.address:** This property specifies the URL of the medication service used by the adapter.context property.
- **adapter.termuinology.icd.port.address:** This property specifies the URL of the ICD-10 service provider used by the adapter.context property.
- **adapter.proxy.host:** This property specifies the fully qualified host-name of the proxy host which is required to access the relevant services used by the adapter.context property.
- **adapter.proxy.port:** This configuration property specifies the proxy port number belonging to the adapter.proxy.host property.
- **adapter.proxy.exclude:** This property excludes an existing intranet nearside of the proxy. The intranet based services need to be excluded from going through the proxy. For excluding all IPs from going through the proxy a wildcard (=\*) has to be used. This corresponds to disabling the proxy. Separate IPs with a pipe symbol "|" if you want to explicitly name more than one IP address.

## D References

### D 1 Sun Java

For more information about the Sun JDK please consult the main project site:

<http://java.sun.com>

### D 2 Apache

For more information about Apache Web Server please consult the main project site on apache.org:

<http://httpd.apache.org/>

The English documentation of the Apache 2.2 project can be obtained here:

<http://httpd.apache.org/docs/2.2/en/>

### D 3 Apache Ant

The main website of the Apache Ant project can be found here:

<http://ant.apache.org/>

The English manual for the Apache Ant 1.6.5 project is located here:

<http://ant.apache.org/manual/index.html>

### D 4 Tomcat

The main website of the Tomcat project can be found here:

<http://tomcat.apache.org/>

For detailed information on the Tomcat project, see the user guide which is located here:

<http://tomcat.apache.org/tomcat-6.0-doc/index.html>

## 9 Reporting Problems

If you encounter any problems with the installation of eHF, please report them as a JIRA issue.

Navigate your browser to the following URL:

<https://jiraweb.intercomponentware.com/jira/browse/BAS>