# ZEMENTIS Predictive Analytics

# Deployment Guide

10.1.0.0

# ZEMENTIS Predictive Analytics

# Deployment Guide



This document applies to ZEMENTIS 10.1.0.0 and to all subsequent releases.

Specifications contained herein are subject to change and these changes will be reported in subsequent release notes or new editions.

# Table of Contents


1. Introduction ..... 1
    1.1. Audience ..... 1
2. Installation ..... 2
    2.1. Requirements ..... 2
    2.2. Security ..... 3
    2.3. Installation Overview ..... 3
    2.4. Apache Tomcat ..... 4
        2.4.1. Install and Configure the Application Server ..... 4
        2.4.2. Add Required Libraries ..... 4
        2.4.3. Configure Security and Users ..... 5
            2.4.3.1. Using `UserDatabaseRealm` ..... 5
        2.4.4. Deploy ZEMENTIS ..... 5
    2.5. IBM WebSphere Application Server Liberty ..... 6
        2.5.1. Install and Configure the Application Server ..... 6
        2.5.2. Deploy Required Libraries ..... 6
        2.5.3. Configure Security and Users ..... 7
            2.5.3.1. Using `basicRegistry` ..... 7
        2.5.4. Deploy ZEMENTIS ..... 8
3. Product License Key ..... 9
4. Configuration ..... 10
    4.1. Repository ..... 10
        4.1.1. Repository Configuration ..... 10
        4.1.2. Repository Migration ..... 12
    4.2. Logging ..... 13
        4.2.1. File Logging ..... 13
        4.2.2. Database Logging ..... 14

# List of Tables

2.1. System Requirements ..... 2

# Chapter 1. Introduction

ZEMENTIS (ZEMENTIS Predictive Analytics) is a Java EE enterprise application and is designed to run within a Java EE Application Server. It has a web front-end component that users may access through a web browser. It also comes with a web services interface allowing applications to access it programmatically. This document describes the steps required to install and configure ZEMENTIS on IBM WebSphere Application Server Liberty and Apache Tomcat. Chapter 2 provides an overview of the installation process and outlines the detailed steps to install ZEMENTIS on IBM WebSphere Application Server Liberty, and Apache Tomcat using the default configuration. Chapter 4 describes the optional configuration modifications that can be applied to an installation in order to use an external repository or a logging database.

## 1.1. Audience

This document is intended for individuals who are fulfilling the role of a system or application administrator. This role is typically filled by a person who has working knowledge of

- Server administration for the Windows or Linux platform
- Installing and configuring the target application server (IBM WebSphere Application Server Liberty or Apache Tomcat)

# Chapter 2. Installation

ZEMENTIS is packaged as a Java Enterprise Archive (EAR) file, ready to be deployed within a Java EE Application Server. This chapter describes the steps to deploy and run ZEMENTIS within Apache Tomcat and IBM WebSphere Application Server Liberty using basic configuration.

## 2.1. Requirements

The system requirements for installing ZEMENTIS are listed in Table 2.1.

### Table 2.1. System Requirements

| Component | Requirement |
|---|---|
| Platform | A hardware and operating system configuration that is supported by the Oracle Java Development Kit (see below). This includes recent versions of Linux, Solaris, Windows, and Mac OS. Both 32 and 64 bit platforms are supported, but a 64 bit platform is preferred. |
| Memory | A minimum of 4 GB of RAM. |
| Disk Space | A minimum of 5 GB of disk space is recommended for a basic installation. Additional storage may be required for working and temporary space. The amount of the total required disk space depends on the size of files (models, data, etc.) expected to be uploaded to and processed by ZEMENTIS. |
| JDK | Oracle Java Development Kit version 7 or above, available at the Java Download Page. Please make sure you use the Java Development Kit (JDK) and not the Java Runtime Environment (JRE). |
| Application Server (one of) | Apache Tomcat version 7.0.59, 8.0.20, or 9.0.0.M10 |
| | IBM WebSphere Application Server Liberty version 16.0.0.2 |

#### Important

In case of ZEMENTIS server upgrade or server migration, it is imperative that all models and resources stored on current ZEMENTIS server installation are backed up and restored using migration tool provided in `adapa-app-10.1.0.0.zip`. For more details, please refer to Section 4.1.2.

## 2.2. Security

For security, ZEMENTIS relies on the JAAS framework and the related modules offered by the application server. This allows integration with a wide variety of authentication and authorization infrastructures. The details for configuring such modules are different for each application server (see sections following below).

Users are authorized to access ZEMENTIS if they are assigned either the `adapa-admin` or the `adapa-ws-user` role. Users with the `adapa-admin` role have unrestricted access to the ZEMENTIS Console, ZEMENTIS REST API, and the ZEMENTIS Web Services. They can manage and execute predictive models. Users with the `adapa-ws-user` role may only execute models through the ZEMENTIS REST API and ZEMENTIS Web Services. They are not authorized to use the ZEMENTIS Console or perform any management operations.

## 2.3. Installation Overview

Installing ZEMENTIS with the default configuration requires four steps:

1. Install and configure the application server.
2. Add required libraries.
3. Configure security and users.
4. Deploy ZEMENTIS.

### Important

Before proceeding with the installation, it is recommended that you set an environment variable with the name `ADAPA_HOME`. The value of this variable points to the absolute path on the server where all the ZEMENTIS artifacts will be deposited. Under default configuration, these artifacts include PMML files and Custom Resources uploaded on ZEMENTIS, and the Product License Key file (`zementis.license`).

Alternatively, if it is not possible to set an environment variable, you can also pass `ADAPA_HOME` as a JVM argument using the `-D` option to the JVM when starting the application server as follows:

```
-DADAPA_HOME=/absolute/path/to/adapa_home
```

Please make sure the directory pointed to via `ADAPA_HOME` exists and has write access.

# 2.4. Apache Tomcat

The description of the steps in this section assumed familiarity with installation and administration of the Apache Tomcat. Detailed information about any of the necessary administrative tasks can be found at the Apache Tomcat 7 Documentation, Apache Tomcat 8 Documentation, and Apache Tomcat 9 Documentation.

## 2.4.1. Install and Configure the Application Server

Download Apache Tomcat from the Apache Tomcat 7 downloads website (apache-tomcat-7.0.59.zip), Apache Tomcat 8 downloads website (apache-tomcat-8.0.20.zip), or Apache Tomcat 9 downloads page version 9.0.0.M10 (Apache Tomcat Archives Page). Decompress the downloaded file and copy the contents into your target installation directory. We recommend that you at least modify the memory settings for the Java Virtual Machine (JVM) and the file size threshold for the modules. The memory settings can be modified by adding the `JAVA_OPTS` parameter in the configuration file *`TOMCAT_HOME`*`/bin/setenv.sh` for Linux or *`TOMCAT_HOME`*`/bin/setenv.bat` for Windows (*`TOMCAT_HOME`* is the directory where Apache Tomcat is installed). If the configuration file does not exist, you may create it manually. Add the value of the parameter to include the following minimum values:

```
JAVA_OPTS="-Xms2048m -Xmx2048m -XX:MaxPermSize=512m -XX:+UseG1GC"
```

### Tip

The file size of `adapaconsole.war` exceeds the default file size threshold value of Apache Tomcat Web Application Manager. It will cause deployment failure if ZEMENTIS Console is deployed via Apache Tomcat Web Application Manager. To avoid this problem, the file size threshold for the modules can be modified by changing the value of `max-file-size` and `max-request-size` in the configuration file *`TOMCAT_HOME`*`/webapps/manager/WEB-INF/web.xml`. Add the value of the parameter to include the following threshold values:

```
<multipart-config>
    <!-- 500MB -->
    <max-file-size>524288000</max-file-size>
    <max-request-size>524288000</max-request-size>
    <file-size-threshold>0</file-size-threshold>
</multipart-config>
```

## 2.4.2. Add Required Libraries

ZEMENTIS requires three JAR files to be available in Apache Tomcat: `jsf-api-2.2.13.jar`, `jsf-impl-2.2.13.jar`, and `mail-1.4.7.jar`. Download and copy these files in the Apache Tomcat library directory *`TOMCAT_HOME`*`/lib`.

## 2.4.3. Configure Security and Users

To configure security, a login module appropriate for the target environment must be configured. This can be achieved by defining an authentication policy in the `tomcat-users.xml` file located at *`TOMCAT_HOME`*`/conf`. Below, we describe how to configure the Realm provided by the Apache Tomcat. For details on configuring other login modules, please review the Realm Configuration on Apache Tomcat 7 documentation, Realm Configuration on Apache Tomcat 8 documentation, Realm Configuration on Apache Tomcat 9 documentation, or contact Software AG

### 2.4.3.1. Using `UserDatabaseRealm`

This section describes how to configure the `UserDatabaseRealm` provided by the Apache Tomcat. This is a simple module that stores the pertinent user information in the `tomcat-users.xml`. To enable it, add the XML snippet in following listing to the `tomcat-users.xml` file.

```
<tomcat-users>
    <role rolename="adapa-ws-user"/>
    <role rolename="adapa-admin"/>
    <user username="adapa" password="adapa" roles="adapa-ws-user, adapa-admin"/>
</tomcat-users>
```

This configuration snippet defines the user with name `adapa` and password `adapa` with both the `adapa-admin` and the `adapa-ws-user` roles.

## 2.4.4. Deploy ZEMENTIS

Apache Tomcat which is a servlet container doesn't support the full java EE stack. You will need to extract the provided ZEMENTIS EAR file (`adapa-app-10.1.0.0.ear`) into three WAR files and deploy them separately (`adapaconsole.war`, `adapaws.war`, and `adapars.war`) using Apache Tomcat Web Application Manager or copying files manually.

Once ZEMENTIS has been deployed and started, point your browser to the ZEMENTIS Console. The *URL* would be

```
http://hostname:8080/adapaconsole
```

where `hostname` is the name of the host where Apache Tomcat is deployed and `8080` is the default port. These may be different according to your application server configuration. You may try to log in using the credentials of a user with `adapa-admin` role.

You may access ZEMENTIS REST API at

```
http://hostname:8080/adapars
```

You may also get a description of the ZEMENTIS Web Services by visiting the *URL*

```
http://hostname:8080/adapaws
```

# 2.5. IBM WebSphere Application Server Liberty

The description of the steps in this section assumed familiarity with installation and administration of the IBM WebSphere Application Server Liberty. Please note, ZEMENTIS deployment configuration is based on IBM WebSphere Application Server Liberty (16.0.0.2). The setting may vary in different versions of IBM WebSphere Application Server Liberty. Detailed information about any of the necessary administrative tasks can be found at IBM WebSphere Application Server Liberty Documentation.

## 2.5.1. Install and Configure the Application Server

The first step in the process is to install IBM WebSphere Application Server Liberty. The application server configuration topology is determined by the `server.xml` and more information can be found from the IBM WebSphere Application Server Liberty Documentation or refers to the snippet in Section 2.5.3.1.

We recommend that you modify the memory settings for the Java Virtual Machine (JVM). This can be modified by adding file `jvm.options` to the server. Please refer to the detail in JVM arguments setting. The following is the minimum recommended setting:

```
"-Xms2048m -Xmx2048m -XX:MaxPermSize=512m -XX:+UseG1GC"
```

### Note

Due to a restriction of IBM WebSphere Application Server Liberty in Windows, we recommend simplifying the IBM WebSphere Application Server Liberty installation directory. For details, please see IBM WebSphere Application Server Liberty Documentation.

## 2.5.2. Deploy Required Libraries

In order to make sure that ZEMENTIS Console works as expected with JavaServer Faces, you need to add two libraries: `jsf-api-2.2.13.jar` and `jsf-impl-2.2.13.jar`; In addition, ZEMENTIS REST requires application server supporting RESTful API 2. `javax.ws.rs-api-2.0.jar` should be added to library also. Put these libraries in the folder *`WLP_HOME`*`/usr/servers/`*`SERVER_NAME`*`/`*`LIB_FOLDER`*. (*`WLP_HOME`* is the directory where IBM WebSphere Application Server Liberty is installed. *`SERVER_NAME`* is the server name you created. *`LIB_FOLDER`* is the folder name you defined in `server.xml`. Take XML snippet in Section 2.5.3.1 as example, the *`LIB_FOLDER`* would be `myLib`.)

## 2.5.3. Configure Security and Users

As described in Section 2.2, users accessing ZEMENTIS must be either in the administrator or the user role. For the IBM WebSphere Application Server, this means that you must create two groups `adapa-admin` and `adapa-ws-user` in the security realm of your domain and assign the appropriate users to these groups as necessary. For details on how to create and assign groups, please consult the IBM WebSphere Application Server Liberty Documentation.

### 2.5.3.1. Using `basicRegistry`

This section describes how to configure the basic user registry provided by the IBM WebSphere Application Server Liberty. This is a simple module that uses `server.xml` for storing the pertinent user information. To enable basic user registry, add `appSecurity-2.0` Liberty feature and the XML snippet in following list to the `server.xml`.

```
<server description="LibertyProfile">
    <featureManager>
        <feature>appSecurity-2.0</feature>
        <feature>json-1.0</feature>
        <feature>jsp-2.3</feature>
        <feature>servlet-3.1</feature>
        <feature>ssl-1.0</feature>
        <feature>localConnector-1.0</feature>
        ...
    </featureManager>
    ...
    <httpSession cookieName="adapa_cookie"/>
    <logging hideMessage="SRVE9967W, CWNEN0070W"/>
    ...
    <library id="myLibrary" apiTypeVisibility="spec, ibm-api, api">
      <fileset dir="${server.config.dir}/myLib" includes="*" scanInterval="5s" />
    </library>
    <application type="ear" id="adapa" location="${server.config.dir}/apps/adapa.ear">
      <classloader apiTypeVisibility="spec, ibm-api, api" commonLibraryRef="myLibrary"
 delegation="parentLast"/>
      <application-bnd>
        <security-role name="adapa-admin">
          <group name="adapa-admin" />
        </security-role>
        <security-role name="adapa-ws-user">
          <group name="adapa-ws-user" />
        </security-role>
      </application-bnd>
    </application>
    <basicRegistry id="basic" realm="ADAPA Security Roles">
      <user name="adapa-admin" password="myPassword" />
      <user name="adapa-ws-user" password="myPassword" />
      <group name="adapa-admin">
        <member name="adapa-admin" />
      </group>
      <group name="adapa-ws-user">
        <member name="adapa-ws-user" />
      </group>
    </basicRegistry>
</server>
```

### Note

With this registry configuration, the IBM WebSphere Application Server Liberty would share a common session for web applications.

## 2.5.4. Deploy ZEMENTIS

To deploy ZEMENTIS, copy the provided EAR file (`adapa-app-10.1.0.0.ear`) into *`WLP_HOME`*`/usr/servers/`*`SERVER_NAME`*`/apps` directory and configure the application information in the `server.xml`.

Once ZEMENTIS has been deployed and started, point your browser to the ZEMENTIS Console. The *URL* would be

```
http://hostname:9080/adapaconsole
```

where `hostname` is the name of the host where IBM WebSphere Application Server is deployed and `9080` is the default port. These may be different according to your application server configuration. You may try to log in using the credentials of a user with `adapa-admin` role.

You may access ZEMENTIS REST API at

```
http://hostname:9080/adapars
```

You may also get a description of the ZEMENTIS Web Services by visiting the *URL*

```
http://hostname:9080/adapaws
```

# Chapter 3. Product License Key

ZEMENTIS requires a valid Product License Key. The Product License Key can be obtained by contacting Software AG. To install the Product License Key, login to the ZEMENTIS Console and click on the License link located on top-right corner of the user interface. This will pop-up the License Panel where the contents of the Product License Key file can be copied.

Alternatively, the Product License Key file (`zementis.license`) can be copied under the `ADAPA_HOME` directory before starting the application server. Please refer to Section 2.3 for more information about how to setup `ADAPA_HOME`.

A valid license key will unlock all the functionality of ZEMENTIS.

# Chapter 4. Configuration

ZEMENTIS has been designed to easily support customizations and/or extensions needed to meet the requirements imposed by the target environment. Using the popular Spring Framework, it allows injecting external resources either as configuration modifications or as extensions. Practically, this means that ZEMENTIS can be customized by providing an appropriate Spring context file along with the necessary custom implementations and required libraries. Such a context file must be placed in the working directory of the server (*TOMCAT_HOME*`/bin` directory for the Apache Tomcat and *`profile_root`* directory for the IBM WebSphere Application Server). One or more context files may be used. For configuration purposes and upon start-up, ZEMENTIS will examine any files in the server's working directory following the name pattern `adapaContext*.xml`. Please note that configuration changes through context files require a server restart before they can take effect.

The following sections describe some of the more common configuration tasks. Please contact Software AG for details on more advanced customization options.

## 4.1. Repository

### 4.1.1. Repository Configuration

ZEMENTIS requires a repository for storing the necessary artifacts. By default, it uses a file-based repository and stores the uploaded artifacts in a directory named `adapa-store` under `ADAPA_HOME` directory. Please refer to Section 2.3 for more information about how to setup `ADAPA_HOME`.

However, it also includes a module that uses a relational database as the repository. The module provided for the database repository leverages the Java Persistence API (JPA) . To use it, it must be configured in a Spring context file. With JPA as the persistence mechanism, the configuration simply requires associating the database back-end module to an `EntityManagerFactory`.

In the rest of this section, we provide an example of how to use a MySQL instance as the database back-end, with the Hibernate as the JPA provider. The first step is to create a Spring context file (e.g. `adapaContext.xml`) and place it in the server's working directory. The following listing presents the contents of this file.

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"
    xmlns:aop="http://www.springframework.org/schema/aop" xmlns:tx="http://www.springframework.org/
schema/tx"
    xmlns:jee="http://www.springframework.org/schema/jee"
    xmlns:context="http://www.springframework.org/schema/context"
    xsi:schemaLocation="
```

```
        http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-
beans-3.2.xsd
        http://www.springframework.org/schema/context http://www.springframework.org/schema/context/
spring-context-3.2.xsd
        http://www.springframework.org/schema/tx http://www.springframework.org/schema/tx/spring-
tx-3.2.xsd
        http://www.springframework.org/schema/aop http://www.springframework.org/schema/aop/spring-
aop-3.2.xsd
        http://www.springframework.org/schema/jee http://www.springframework.org/schema/jee/spring-
jee-3.2.xsd"
    default-lazy-init="true">

    <bean id="assetStoreBackend" class="com.zementis.adapa.store.backend.DBAssetStoreBackend">
        <property name="entityManagerFactory">
            <bean class="org.springframework.orm.jpa.LocalContainerEntityManagerFactoryBean">
                <property name="persistenceXmlLocation" value="file:persistence.xml" />
            </bean>
        </property>
    </bean>

</beans>
```

With this configuration file, we define a bean named `assetStoreBackend` as an instance of the class `DBAsset-StoreBackend`. This bean needs to be initialized with an `EntityManagerFactory`. In this example, we use the facility available in the Spring framework through the class `LocalContainerEntityManagerFactoryBean` and provide the persistence configuration parameters in a separate file named `persistence.xml`. This second file must also reside in the server's working directory. To use Hibernate and MySQL, the `persistence.xml` file should contain the following information:

```
<persistence xmlns="http://java.sun.com/xml/ns/persistence" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"
    xsi:schemaLocation="http://java.sun.com/xml/ns/persistence http://java.sun.com/xml/ns/persistence/
persistence_1_0.xsd"
    version="1.0">
    <persistence-unit name="AdapaAsset" transaction-type="RESOURCE_LOCAL">
        <provider>org.hibernate.ejb.HibernatePersistence</provider>
        <non-jta-data-source>java:mysqlds</non-jta-data-source>
        <class>com.zementis.adapa.store.backend.DBAsset</class>
        <class>com.zementis.adapa.store.backend.DBAssetSource</class>
        <class>com.zementis.adapa.store.backend.DBAssetAnnotatedSource</class>
        <class>com.zementis.adapa.store.backend.DBAssetByteStream</class>
        <properties>
            <property name="hibernate.dialect" value="org.hibernate.dialect.MySQLDialect" />
            <property name="hibernate.show_sql" value="false" />
            <property name="hibernate.hbm2ddl.auto" value="update" />
        </properties>
    </persistence-unit>
</persistence>
```

The key configuration parameters in this file are the data source and the database dialect Hibernate should employ. The data source is looked up by its JNDI name (`java:mysqlds` [1]). This requires that the data source to the appropriate database has already been configured and deployed under the same name on the application server. For details on configuring data sources, please consult the documentation of the application server. The Hibernate

[1] In some environments, the name of the data source may have to be provided without the `java:` prefix. The path of data source may also vary with application servers. (e.g. `java:/comp/env/mysqlds` in Apache Tomcat)

dialect parameter identifies the type of database that is used. For more information about the supported databases and available Hibernate database dialects, please refer to the Hibernate Dialect Javadocs.

Additional sample configuration files for other databases may be found in the `configuration/repository` directory of the ZEMENTIS package.

## 4.1.2. Repository Migration

When migrating existing ZEMENTIS installation to version 10.1.0.0, it is imperative that all models and resources stored on current ZEMENTIS server installation are backed up and restored using migration tool provided in migration directory of `adapa-app-10.1.0.0.zip`. Copying existing ZEMENTIS store to another server is not recommended as it may result in unexpected behavior, unrecoverable errors, or loss of uploaded content. `adapa-app-10.1.0.0.zip` ships with two migration scripts; `migration.sh` which is targeted for Unix (*nix) like platforms with cURL utility installed, and `migration.ps1` for Windows platforms with PowerShell framework version 2.0 or higher.

General syntax for script execution on both platforms is

```
./migration.sh $OPERATION $HOST $USER $PASS $BACKUP_DIR
```

where `$OPERATION` is either `backup` or `restore`, `$HOST` is Uniform Resource Locator (URL) to the host server, `$USER` and `$PASS` contain username and password respectively for user credentials with administrative access rights, and `$BACKUP_DIR` is a path to the backup directory. This last parameter is required only for `restore` operation, but is optional for `backup` operation. When `$BACKUP_DIR` is not provided for `backup` operation, the content is saved in a new directory `adapa-backup-YYMMDD-hhmmss` where `YYMMDD-hhmmss` represents the startTime as per UTC time zone.

For example, execution of command

```
./migration.sh backup http://my-adapa.com:8080 admin admin-pass
```

in the last second of year 2014 will result in download of all models and resources from ZEMENTIS server located at `http://my-adapa.com:8080` to new local directory `adapa-backup-20141231-235959`. Execution of command

```
./migration.sh restore https://my-new-adapa.com admin admin-pass adapa-backup-20141231-235959
```

will result in upload of all content from `adapa-backup-20141231-235959` directory to ZEMENTIS server located at `https://my-new-adapa.com`.

Help menu, usage, and examples can be accessed directly from the script by running

```
./migration.sh
```

command from *nix shell, and

```
Get-Help .\migration.ps1 -Full
```

from Windows® PowerShell command line.

# 4.2. Logging

ZEMENTIS provides a comprehensive logging mechanism for capturing data used during execution of predictive models. The captured data includes input and output values as well as information regarding invalid and missing values presented for scoring. All this information is stored in a logging store and may be used to generate reports that assist in monitoring the performance of a model, assess the health of the data provided to the model, train new models or retrain existing ones, etc. The logging mechanism is not enabled by default in ZEMENTIS. It can be enabled by providing the necessary configuration parameters in a Spring context file (e.g. `adapaContext.xml`) in the server's working directory. The logging mechanism can be configured to publish to a File System or a Database.

## 4.2.1. File Logging

This logging mechanism requires a write-access file system folder. Please refer to Section 2.3 for more information about how to setup ADAPA_HOME. Once this is done, ZEMENTIS will publish log files under `adapa-log` sub-folder of ADAPA_HOME directory. The listing below is an example of a context file that uses the file system for logging.

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"
    xmlns:context="http://www.springframework.org/schema/context"
    xsi:schemaLocation="
     http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-
beans-3.2.xsd
     http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-
context-3.2.xsd">
    <bean id="fileLogHandlingFacility" class="com.zementis.adapa.logging.file.FileLogHandlingFacility" />
</beans>
```

With this configuration, three log files are created for each uploaded model. The files are named `MODEL_Record`, `MODEL_Invalid`, and `MODEL_Missing` where `MODEL` represents the model name. The file with the suffix `_Record` contains all the input and output values (along with all the intermediate values) used during execution of predictive model. The files with suffixes `_Invalid` and `_Missing` contains occurrence counters for each invalid and missing values encountered during the scoring process. These values are published once a day at midnight.

## 4.2.2. Database Logging

This logging mechanism requires an external database. The listing below is an example of a context file that uses a MySQL database for logging purposes. Additional sample configuration files can be found in the `configuration/logging` directory of the ZEMENTIS package.

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"
    xmlns:context="http://www.springframework.org/schema/context" xmlns:tx="http://
www.springframework.org/schema/tx"
 xmlns:jee="http://www.springframework.org/schema/jee"
    xsi:schemaLocation="
     http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-
beans-3.2.xsd
     http://www.springframework.org/schema/tx http://www.springframework.org/schema/tx/spring-tx-3.2.xsd
     http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-
context-3.2.xsd
  http://www.springframework.org/schema/jee http://www.springframework.org/schema/jee/spring-
jee-3.2.xsd">
    <bean id="hibernateLogHandlingFacility"
 class="com.zementis.adapa.logging.hibernate.HibernateLogHandlingFacility">
        <property name="hibernateProperties" ref="hibernateProperties" />
        <property name="dataSource" ref="loggingDataSource" />
    </bean>
    <bean id="hibernateProperties"
 class="org.springframework.beans.factory.config.PropertiesFactoryBean">
        <property name="properties">
            <props>
                <prop key="hibernate.connection.provider_class">
                    com.zementis.adapa.logging.hibernate.HibernateDataSourceConnectionProvider
                </prop>
                <prop key="hibernate.dialect">org.hibernate.dialect.MySQLDialect</prop>
                <prop key="hibernate.hbm2ddl.auto">update</prop>
                <prop key="hibernate.default_entity_mode">dynamic-map</prop>
            </props>
        </property>
    </bean>
 <jee:jndi-lookup id="loggingDataSource" jndi-name="java:mysqlds" />
</beans>
```

Database logging has been implemented through the Hibernate framework. Similarly to the database repository (Section 4.1), the two key configuration parameters are the data source and the Hibernate dialect. The data source is looked up by its JNDI name (`java:mysqlds` [2]). This requires that the data source to the appropriate database has already been configured and deployed under the same name on the application server. For details on configuring data sources, please consult the documentation of the application server. The Hibernate dialect parameter identifies the type of database that is used. For more information about the supported databases and available Hibernate database dialects, please refer to the Hibernate Dialect Javadocs.

[2] In some environments, the name of the data source may have to be provided without the `java:` prefix. The path of data source may also vary with application servers. (e.g. `java:/comp/env/mysqlds` in Apache Tomcat)

With this configuration, three tables are created for each uploaded model. The tables are named `MODEL`, `MODEL_Invalid` and `MODEL_Missing` where `MODEL` represents the model name. The table `MODEL` contains all the input and output values used during execution of predictive model. The tables with suffixes `_Invalid` and `_Missing` contains occurrence counters for each invalid and missing values encountered during the scoring process. These values are published once a day at midnight.

## Important

If there are any changes made to the MiningSchema/Output of the model after logging mechanism is enabled, it is necessary to archive the corresponding data (tables or files) before the model is refreshed. In case of database logging, the corresponding tables need to be dropped after archiving is finished. In case of file-based logging, the corresponding files need to be deleted after archiving is finished.