MEDEXTER Healthcare

Medexter Healthcare

ACHIEVING DATABASE CONNECTIVITY IN ARDEN SYNTAX USING THE ARDENSUITE DATABASE CONNECTOR

SIRS Notification as an Example

Datum: 12/17/2019

Version: 1.5

Medexter Healthcare GmbH, Borschkegasse 7/5, A-1090 Vienna, Austria, Telephone +43-1-968 03 24, Facsimile +43-1-968 09 22 www.medexter.com, Handelsgericht Wien: FN 225564m, UID: ATU54901101, IBAN: AT602011128014988900, BIC: GIBAATWW

Summary

The aim of this *how-to* instruction manual is to show how to create, configure, and activate a database connection on an ARDENSUITE Server using a database connector. Furthermore, we show how to use an active database connection in Arden Syntax medical logic modules (MLMs) in order to read data from the connected database.

Document Information

Target Audience

This instruction manual was created for ARDENSUITE users and developers interested in developing or using MLMs in combination with databases on an ARDENSUITE Server.

Impressum

Media owners, editors, publishers:
Medexter Healthcare GmbH, Borschkegasse 7/5, A-1090 Vienna, Austria
Telephone: +43-1-968 03 24, Fax: +43-1-968 09 22, Internet: www.medexter.com
Email: office@medexter.com
CEO: Klaus-Peter Adlassnig, PhD, MSc

Editorial, project management, coordination:
Klaus-Peter Adlassnig, PhD, MSc

Vienna, December 2019

Version: 1.5

Download at www.medexter.com

Introduction

In this *how-to* instruction manual, we guide you step-by-step in the creation, configuration, and activation of a database connection on an ARDENSUITE Server. Furthermore, we illustrate how database connections can be used to retrieve data in Arden Syntax via so-called curly braces. Throughout this *how-to*, we use Arden Syntax Medical Logic Modules (MLMs) containing notification rules for the systemic inflammatory response syndrome (SIRS) as a use case. Clinically, the rules for SIRS notifications are as follows:

> **SIRS Notification**
>
> ALERT if ≥ 2 Criteria
>
> Temperature > 38°C (100.4°F) or < 36°C (96.8°F)
> and/or
>
> Heart rate > 90 beats per minute
> and/or
>
> Respiratory rate > 20 breaths per minute or arterial carbon dioxide tension (PaCO2) < 32 mm Hg
> and/or
>
> White blood cell count (>12,000/µL or < 4,000/µL or >10% immature [band] forms)

Following these provisions, MLMs were constructed that implement these notification rules and generate alerts when patient data match these criteria. In this *how-to*, we discuss two use case scenarios:

- An MLM that takes a patient ID as input parameter, performs a `select` query on the database which returns a single record, assigns values from that record to variables, uses those variables to evaluate aforementioned SIRS notification rules, and generates an alert if said rules apply.
- The second MLM is an extension of the first. In this MLM, the `select` query on the database returns multiple records for the same patient at different times. An alert is only generated if said rules apply and if no other alert was generated prior on the same day (e.g., to avoid overalerting).

Requirements

For optimal use of this *how-to*, please make sure the following software is installed on your computer or accessible from your location:

- The ARDENSUITE IDE and ARDENSUITE Server with Database Connector
- A relational database management system (DBMS) (e.g., MySQL)
- Web service client

The ARDENSUITE IDE and ARDENSUITE Server with Database Connector

In case you do not have access to the ARDENSUITE IDE or the ARDENSUITE Server with Database Connector yet, please contact us at support@medexter.com. A 30-day trial version of the ARDENSUITE IDE can also be downloaded here. If you need help installing or using the ArdenSuite IDE and Server, please visit our ArdenSuite Support Pages.

Relational Database Management System

The methods discussed in this *how-to* document are suited for any relational database that supports JDBC. For the instruction in this document, we used MySQL.

Web Service Client

MLMs deployed on the ARDENSUITE Server are called using web service communication protocols, e.g., Representational State Transfer (REST). For instructional and testing purposes, we illustrate these calls using a web browser. In this document, we recommend using Postman for all REST communication.

Files

This *how-to* is accompanied by three files (download the ZIP file from our Learning Center): an SQL script that can be used to prepare the database tables used in the use case scenarios, and two MLM files (extension `.mlm`).

- DB_SIRS Notification #1: This MLM evaluates the four SIRS notification criteria and returns an alert if two or more criteria are met. The MLM uses a database table and queries the data using a patient ID, which is supplied as a parameter. Based on the query result, which is a single record, aforementioned SIRS notification rules are evaluated.
- DB_SIRS Notification #2: This MLM is an extension of DB_SIRS Notification #1. The MLM also evaluates the four SIRS notification criteria, but only returns an alert if no alert has already been generated on the same day. In this MLM, the database query returns multiple records for the same patient at different times.

NOTE: *The MLM files can be opened using any standard text editor or viewer, but in order to compile*

and upload the MLMs, the ARDENSUITE IDE and ARDENSUITE Server are required.

Preliminaries

Before you can start with the actual *how-to* part of this manual, the database has to be created and MLMs have to be compiled and deployed on the ARDENSUITE Server before calls can be made to them.

Database Setup

The first step is setting up a database (e.g., a MySQL database using XAMPP and phpMyAdmin; the pre-installed MySQL 5.1.40 JDBC driver only supports MySQL Server up to 5.X.). For the purpose of this *how-to*, a database called `SIRSDB` should be created. With this *how-to*, you also received an SQL script (`sirsdb.sql`) that you can load and execute in order to automatically create and fill database tables. Please make sure that the database user account used with the ARDENSUITE Database Connector has the correct privileges, i.e., the user can perform a `SELECT` operation on the tables in `SIRSDB`.

Compiling and Uploading MLMs

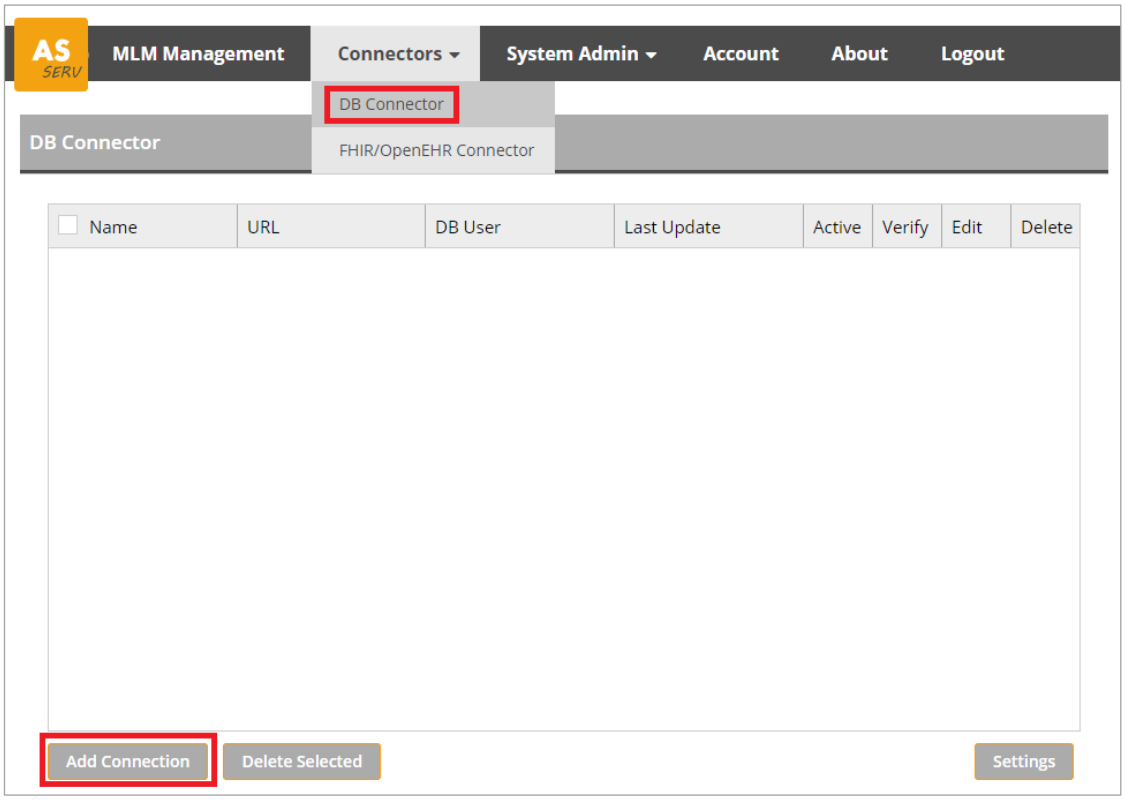
Before an MLM can be called, it first needs to be compiled and uploaded onto the ARDENSUITE Server. To do this, we refer to the corresponding *how-to* document, which can be found here. This tutorial requires you to compile and upload all accompanied MLMs.

Database Connectivity

In this *how-to* we guide you step-by-step in the creation, configuration, and activation of a database connection on an ARDENSUITE Server and we will show you how to use a database connection within Arden Syntax MLMs.

Database Connection Setup

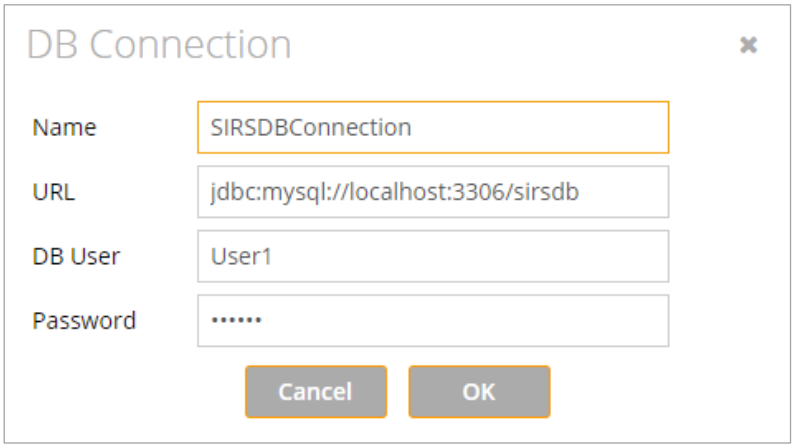
In order for MLMs on the ARDENSUITE Server to retrieve data from our created database, a connection between the ARDENSUITE Server and the DBMS has to be established and configured. After logging into the ARDENSUITE Server frontend, select `DB Connector` in the menu bar, and click on the `Add Connection` button on the bottom left (see figure below).



Upon clicking this button, a dialog window will open. Please enter the following information:

- **Name:** This name is solely used for displaying purposes in the Database Connector frontend in order to make it easier for the user to identify a connection. Thus, any name may be chosen.
- **URL:** The actual database URL. In this *how-to*, we use a MySQL database that was locally installed, hence the address is `localhost`, and the port is `3306`, which is the default listening port for MySQL. Thus, the complete URL is: `jdbc:mysql://localhost:3306/sirsdb`
- **DB User:** The database account username for authentication.
- **Password:** The corresponding database account password for authentication.

For this *how-to*, an example database connection in the dialog window could look like this:



Once a connection has been created, please activate and test the connection. The connection can be activated by clicking on the icon in the `Active` column. This also verifies if the ARDENSUITE Server is able to connect to the database. If the database connection was successfully established, the icon in the `Active` column turns green (see figure below).

Note: *Keep in mind that the MySQL database name is case-sensitive. Furthermore, if the ARDENSUITE Server is restarted, the connection needs to be reactivated (even if it appears to be active).*

Database Use in MLMs

SIRS Notification #1

In this MLM (`DB_SIRS-Notification1.mlm`), an alert is generated if a patient record with a user-specified identifier matches the defined criteria for SIRS. Database interaction usually takes place in the `data` slot of an MLM, in a curly brace block, through SQL statements:

```
data:
  testID:= Argument;
  (temperature,heartRate,respRate,PaCO2,WBcellCount,immatureBand)
    := READ {SELECT temperature, heartRate, respRate, PaCO2,
WBcellCount, immatureBand FROM sirsvalues WHERE IDPatient = testID};
```

In this Arden Syntax code snippet from our first use case, the patient identifier is read from `Argument` and assigned to the variable `testID`. Next comes a `READ` statement, which signals the beginning of a data read in a curly brace block. Within this curly brace block, an SQL query is performed that selects values from the `temperature, heartRate, respRate, PaCO2, WBcellCount,` and `immatureBand` columns in the `sirsvalues` table for all records where the value in the `IDPatient` column matches the value of `testID`. These values are then assigned to the homonymous Arden Syntax variables, after which they are used in the `logic` slot for evaluation of the SIRS notification rules.

Note: *In case the variable* `testID` *used in the query is a string, it should be surrounded by single quotes in order for the DBMS (at least MySQL) to accept it. The following code suffices:*

```
if testID is string then
  testID := "'" || testID || "'";
endif;
```

To call this MLM, start the Postman browser plugin and construct the REST call. For more information on how to call MLMs using REST, we refer to the corresponding *how-to* document, which can be found here. After starting Postman, use the following URL for the MLM REST call:

```
http://localhost:8080/REST/CALLMLM?mlmName=DB_SIRS-
Notification1&mlmInstitution=Medexter%20Healthcare,%20Vienna,%20Aust
ria
```

Note: *Please double-check the URL after copy&paste into Postman. Some PDF readers delete dashes (-) when copying out of a PDF.*

If we select patient identifier `123`, the JSON data that has to be supplied in the `Body` segment of the REST call is:

```
{
  "type": "number",
  "value": 123,
  "applicability": 1,
  "primaryTime": null
}
```

If everything worked out as it should and no errors occurred, the server returns an alert for SIRS. The alert looks like this:

```
{
  "type": "string",
  "primaryTime": null,
  "applicability": 1,
  "value": "Alert for SIRS"
}
```

Note that if you were to select patient identifier `125`, which is a patient that has no SIRS, the server would return null. The output would look like this:

```
{
  "type": "null",
```

```
  "primaryTime": null,
  "applicability": 1
}
```

SIRS Notification #2

This MLM (`DB_SIRS-Notification2.mlm`) is an extension of the MLM previously discussed. Here, an alert is generated if a patient record with a user-specified identifier matches the defined criteria for SIRS, but only if no other alert was generated in the previous 24 hours. Since we now have to account for multiple records, we first define an object `Patient` to logically group patient values:

```
Patient := object [temperature, heartRate, respRate, PaCO2,
WBcellCount, immatureBand];
```

Using this object and the argument provided by the user, we again perform a query, this time on table `SIRSvalues2`:

```
allValues:= READ AS Patient {SELECT temperature, heartRate,
respRate, PaCO2, WBcellCount, immatureBand, Date as PrimaryTime FROM
SIRSvalues2 WHERE IDPatient = testID ORDER BY Date};
```

Since we group values in the object `Patient`, we use the `READ AS` statement. As such, the variable `allValues` contains a list of `Patient` objects, each assigned values of a single record. In the remainder of the MLM, the latest `Patient` object is evaluated first, after which—in case the values in this `Patient` object are eligible to generate an alert—values of the previous 24 hours are evaluated as well. In case one of these were eligible to generate an alert as well, it is assumed an alert was generated, and the current alert is suppressed.

To call this MLM, use the following URL for the MLM REST call:

```
http://localhost:8080/REST/CALLMLM?mlmName=DB_SIRS-
Notification2&mlmInstitution=Medexter%20Healthcare,%20Vienna,%20Aust
ria
```

Note: *Please double-check the URL after copy&paste into Postman. Some PDF readers delete dashes (-) when copying out of a PDF.*

If we select patient identifier `123`, the JSON data that has to be supplied in the `Body` segment of the REST call is:

```
{
```

```
  "type": "number",
  "value": 123,
  "applicability": 1,
  "primaryTime": null
}
```

If everything worked out as it should and no errors occurred, the server returns null because an alert for patient 123 has already been generated in the last 24 hours:

```
{
  "type": "null",
  "primaryTime": null,
  "applicability": 1
}
```

Note that if you were to select patient identifier `124`, which is a patient that has SIRS but for whom no alert had been generated in the last 24 hours, the server would return an alert notification. The output would look like this:

```
{
  "type": "string",
  "primaryTime": null,
  "applicability": 1,
  "value": "Alert for SIRS"
}
```