medexter
clinical decision support

Arden Syntax Server: Manual for Version 3.3. – Administrator, User, and Developer Guide

Karsten Fehre
Software Engineer / Medical Knowledge Engineer

Medexter Healthcare GmbH
Borschkegasse 7/5, A-1090 Vienna, Austria

+43-1-968 03 24 tel
+43-1-968 09 22 fax
kf@medexter.com

2014-11-20

Table of Contents

1. Introduction

The manual at hand aims at describing the handling of the Arden Syntax server created by Medexter Healthcare. The manual is divided into three parts: The administrator guide will provide the local administrator with the tools necessary for installation, configuration, and maintenance. The user guide is intended to show the end user the functionality of the web front end, whereas the third part of this manual, the developer guide, will offer developers information about the interfaces available for communication with the Arden Syntax server. Also, it contains directions on how to integrate the server into their own software/applications.

We consider it the overall goal of our Arden Syntax software systems to offer tools for the representation and processing of medical knowledge based on a standardized syntax. For this purpose, we developed the Arden Syntax engine, which executes so called Medical Knowledge Modules (MLMs) written in Arden Syntax. The Arden Syntax Server is built around this engine in order to enable service-oriented access for arbitrary client applications. Additionally, the Arden Syntax server allows easy management of MLMs and compiled knowledge files using our Arden Syntax compiler.

A specific feature of the Arden Syntax engine, and thus the Arden Syntax server, is that it fully implements Health Level Seven (HL7) International's Arden Syntax specification [1-3].

The Arden Syntax server can be further characterized by the following features:

- *Concurrency*: The Arden Syntax server can handle concurrent calls from outside by executing the respective MLMs in parallel or by scheduling them according to priority.
- *Multiple executions*: The Arden Syntax server permits multiple executions of MLMs with one single call.
- *Asynchrony calls*: A requesting instance may use the Arden Syntax server to call an MLM and retrieve the results of the execution later.
- *Independence from Java*: By providing an external interface that communicates with the environment via XML messages, there is no need for external resources to use Java for their implementation.

Additionally, the Arden Syntax server provides a Graphical User Interface that permits end users to add and remove MLMs on their own.

With the aim being to handle read-and-write requests from the Arden Syntax MLMs to the host information system, the Arden Syntax server has the ability to process curly brace expressions and to return the results of these requests to the Arden Syntax engine. Curly brace expressions are sent to an external component that has to provide a simple interface for communication with the Arden Syntax server. Since data sources depend on the environment where the Arden Syntax server is used, this external component has to be adopted whenever a new data source is used (see chapter 4.1 for more information).

One of the main purposes of the Arden Syntax server is to provide the Arden Syntax engine with simple web service interfaces for easy integration. By using these interfaces, a knowledge-based clinical decision support functionality, or the like, can be added to any existent application. The communication with the Arden Syntax server follows the Arden Syntax server protocol, which is based on sending and receiving documents according to the rules of XML. Figure 1 illustrates a schematic document for calling an MLM:

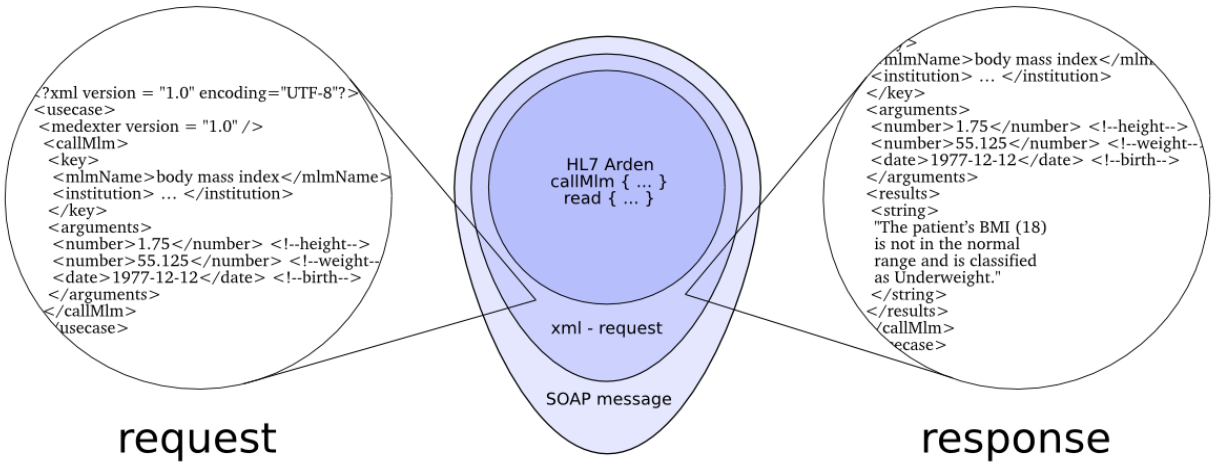

Figure 1: SOAP wrapped Arden Syntax communication

Each XML document is wrapped into a SOAP (Simple Object Access Protocol) message, which is transported via HTTP (Hypertext Transfer Protocol). For the sake of completeness, the Arden Syntax server also provides a description of its functionality given by a WSDL-formatted document.

2. Administrator Guide

2.1. Setup

This section describes the setup procedure for the Arden Syntax server. It is recommended to follow the instructions step-by-step.

2.1.1. System Requirements

Java SE Runtime Environment 6

2.1.2. Unzip

Unzip ArdenSyntaxServer.zip to a folder on your hard drive (the new folder will be named $SERVER_HOME throughout this document).

2.1.3. Start/Stop Server

The Arden Syntax server can be started using the following command:

Linux:

```
# $SERVER_HOME/startup.sh
```

Windows:

```
# $SERVER_HOME/startup.bat
```

If the Arden Syntax server has been started correctly, the default Glassfish welcome page should be displayed when opening the URL http://$servername:4848 in a web-browser.

NOTE: If the port 8080 is already used by another application on your computer, the $SERVER_HOME/glassfish/domains/tamAS/conf/domain.xml must be adjusted. The following entry must be changed to an open port on the host:

```
<network-listener port="$YOUR_PORT " protocol="http-listener-1" transport="tcp"
name="http-listener-1" thread-pool="http-thread-pool" />
```

The Arden Syntax server can be stopped using the following command:

Linux:

```
# $SERVER_HOME/shutdown.sh
```

Windows:

```
# $SERVER_HOME/shutdown.bat
```

2.1.4. Initialization

To initialize the Arden Syntax server, please open the following URL in your browser:

```
http://$SERVERNAME:8080/ArdenSyntaxServerFrontend/init.zul
```

Click "Initialize Arden Syntax Server". The server then generates an internal database and all necessary tables. Subsequently, the admin user for the web front end is created automatically. The admin user can be configured using the configuration file (see section 2.2.1).

NOTE: To prevent an unintended reinitialization of the Arden Syntax server, we suggest that you perform one of the following tasks:

- deleting the file:

```
# $SERVER_HOME/glassfish/domains/tamAS/applications/AS_WebFrontend/init.zul
```

- moving the file to any backup-folder (to eventually restore the file) outside the $SERVER_HOME folder

2.1.5. First Start

After initializing the Arden Syntax server, the web front end can be accessed via:

```
http://$SERVERNAME:8080/ArdenSyntaxServerFrontend/login.zul
```

The administrator can now login with the configured admin username (see 2.2.2).

2.2. Configuration

The Arden Syntax server configuration will be described below. It will be shown how the configuration file can be used to specify the default administrator and other user groups as well as to set the runtime language and remote connections, among others. Additionally, we will describe how to configure the Arden Syntax server components to run on different physical computers, if needed.

2.2.1. Configuration File Location

The configuration file can be found on the following location:

```
# $SERVER_PATH\glassfish\domains\tamAS\config\ardenServer.properties
```

Any changes in this file will apply after restarting the Arden Syntax server.

2.2.2. Admin Configuration

The admin section of the configuration file determines the default administrator and the two basic user groups (admin and simple user group) as well as their description. During initialization, those entries will be used to create the default administrator account and – in addition – the above mentioned two groups:

```
#+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
# ADMIN CONFIGURATION
#
# Used to specify the default administrator user and its password
#+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
server.defaultAdmin.login=admin
server.defaultAdmin.password=pass
server.defaultAdmin.groupDescription=administrator, can additionally manage user
accounts, restore MLMs and see statistics
server.defaultAdmin.simpleGroup=simple User
server.defaultAdmin.simpleGroupDescription=normal User, can upload and delete MLMs
```

2.2.3. Engine Configuration

The engine section contains settings for the runtime language of the Arden Syntax engine. It enables MLMs with multilingual resources to determine which language to use:

```
#+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
# ENGINE CONFIGURATION
#
# Used to configure the arden engine
#+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
# localization, possible values are 'DE' and 'US'
server.engine.locale=US
```

2.2.4. Remote Connections

On the one hand, the remote connections part of the configuration file specifies the connection to the remote application that is used to evaluate curly brace expressions (see section 4.1 for more details). If you do not change the default settings, a dummy component that is shipped with the Arden Syntax server is used. This dummy component

is not able to evaluate any curly brace expressions.

On the other hand, the remote connections part is responsible for the configuration of asynchrony calls. If the web service is called asynchronously, these settings are used to find a remote web service for sending the response to:

```
#++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
# REMOTE CONNECTIONS
# These entries are for the configuration of the call-back mechanism of the
# arden syntax engine. If evaluation of curly braces etc. should be
# done by the local implementation in AS_RMI_Default, both RmiServiceHost
# and RmiServicePort must be defined but RemoteServiceUrl must be empty.
# If RemoteServiceUrl is set to any remote AS implementation,
# the engine will send all its call-backs to this server, ignoring the
# other rmi-settings.
#++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
RmiServiceHost=localhost
RmiServicePort=18989
RemoteServiceUrl=

Remote.async.user=test
Remote.async.password=test
Remote.async.namespaceUri=http://services.server.arden.medexter/
Remote.async.localServiceName=WebServiceASService
Remote.async.localServicePort=WebServiceASPort
Remote.async.endpoint=http://exampleURL
Remote.async.methodName=evaluateXMLRequest
Remote.async.prefix=ser
Remote.async.argument=arg0
```

2.2.5. Logging Configuration

The logging section allows to specify the storage location of all access statistics:

```
#++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
# LOGGING CONFIGURATION
#
# Used to specify the location of log files
#++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
server.jamon.logfile=../logs/jamonstatistics.html
```

2.2.6. Communication Settings

It is possible to distribute the Arden Syntax server components to different hosts (e.g., for performance reasons). That is, the web front end, back end, and web service interfaces could be running on different servers. This section consists of all settings concerning communication between the Arden Syntax server components. Use the following configurations to enable communication between them:

```
#+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
# COMMUNICATION SETTINGS
#
# These settings are used to configure the communication between
# the Arden Syntax server components.
#+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
communication.xml.initial=com.sun.enterprise.naming.SerialInitContextFactory
communication.xml.pkgs=com.sun.enterprise.naming
communication.xml.state=com.sun.corba.ee.impl.presentation.rmi.JNDIStateFactoryImpl
communication.xml.ORBInitialHost=
communication.xml.ORBInitialPort=
communication.frontend.initial=com.sun.enterprise.naming.SerialInitContextFactory
communication.frontend.pkgs=com.sun.enterprise.naming
communication.frontend.state=com.sun.corba.ee.impl.presentation.rmi.JNDIStateFactor
yImpl
communication.frontend.ORBInitialHost=localhost
communication.frontend.ORBInitialPort=3700
```

2.3. Interface Access Configuration

Both the SOAP and the REST interface to the Arden Syntax server have access restrictions. Only users that are members of groups with corresponding permissions have access to those interfaces (see 3.3 for more details).

2.4. Backup

The Arden Syntax server allows to backup MLMs, user groups, user accounts and database connector data in backup files. These backup files can be used to restore the state of the Arden Syntax server at the time of their creation. Only a user with administrative rights is allowed to create these backup files. It is possible to either create one backup file containing all Arden Syntax server data or, alternatively, to create several files containing the following parts individually:

- MLM Management: Data of all uploaded MLM files
- User Management: Data of all user groups, permissions and user accounts
- DB Connector Data: All stored database connections

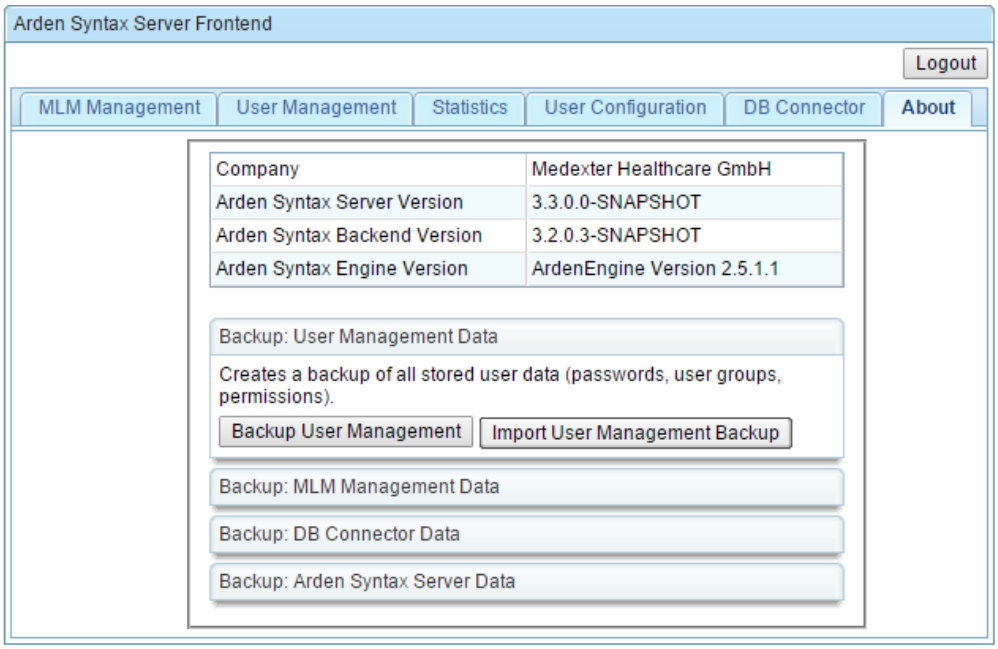

Figure 3: Backup functionality

In order to create a backup file, please click on the respective "Backup: …" row in the bottom half of the "About" tab ("Backup: User Management Data", "Backup: MLM Management Data", "Backup: DB Connector Data", "Backup: Arden Syntax Server Data"). In the dialogue that opens, there are two options available: For backup, please press the respective "Backup…" button. In case you want to import an existing backup file, the file can be uploaded using the respective "Import … Backup" button.

NOTE: All existing data (within the category to be restored) will be deleted before restoring a previous state.

3. User Guide

After start-up and configuration of the Arden Syntax server, the management web front end is available. It enables the user to manage the MLMs and user accounts available on the Arden Syntax server.

3.1. Login

The Arden Syntax server web front end can be accessed using the following URL:

```
http://$SERVERNAME:8080/ArdenSyntaxServerFrontend/login.zul
```

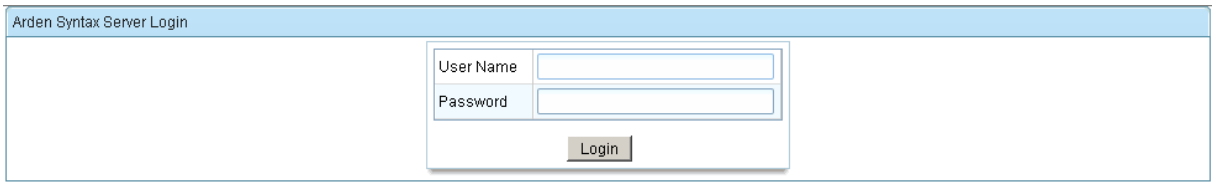

Figure 4: The "Login" window

The first screen shows the login window (Figure 4). For first access or administrator login, please use the password created in section 2.2.1.

3.2. MLM Management

Once logged in, the user will have the possibility to add and remove MLMs from the Arden Syntax server on the "MLM Management" tab (Figure 5, see below). MLM management is available for all users.

To add an MLM to the Arden Syntax server, the compiled MLM (.mlmobj.gz extension is required) has to be uploaded using the "Upload" button on the "MLM Management" tab. To upload more than one MLM, the MLMs must be packed into a zip file.

An MLM can be removed from the server by pressing the MLM's respective "Delete" button at the end of the row. The MLM will then be marked as "Deleted" and is no longer available for execution. An MLM that is marked "Deleted" can be restored in the "Deleted MLMs" window (see Figure 6) which is available by pressing the "Manage Deleted MLMs" button.

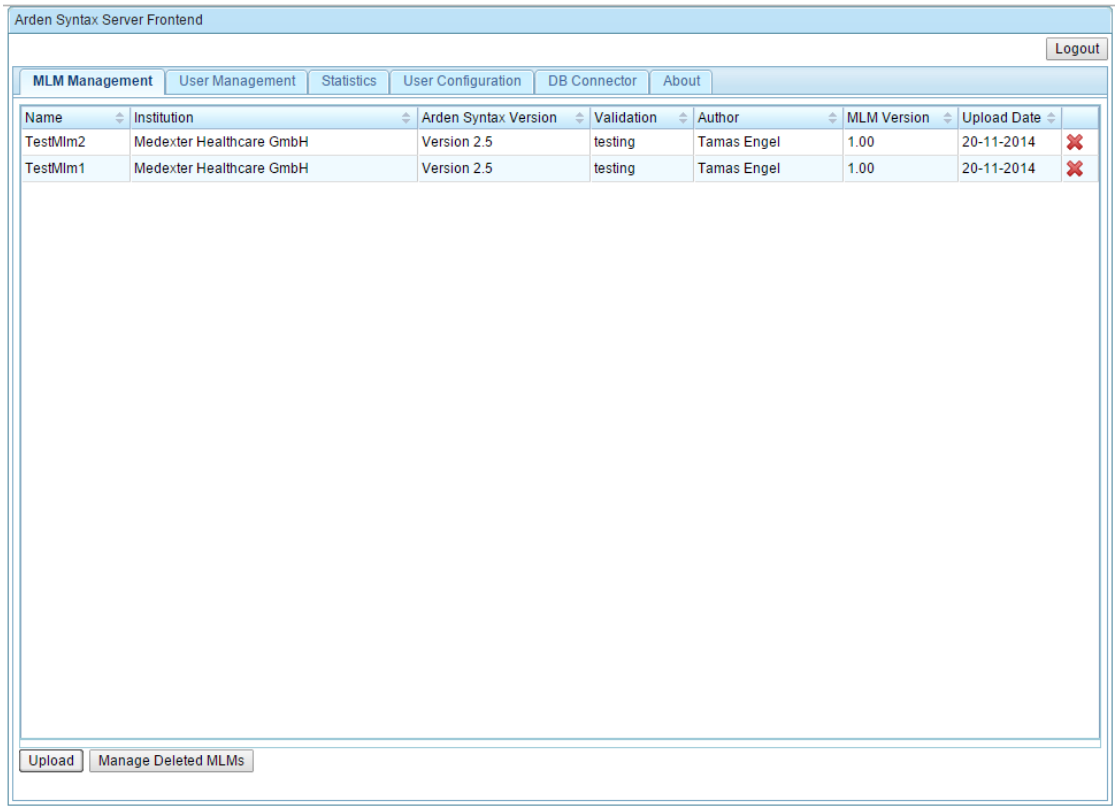

Figure 5: The "MLM Management" tab

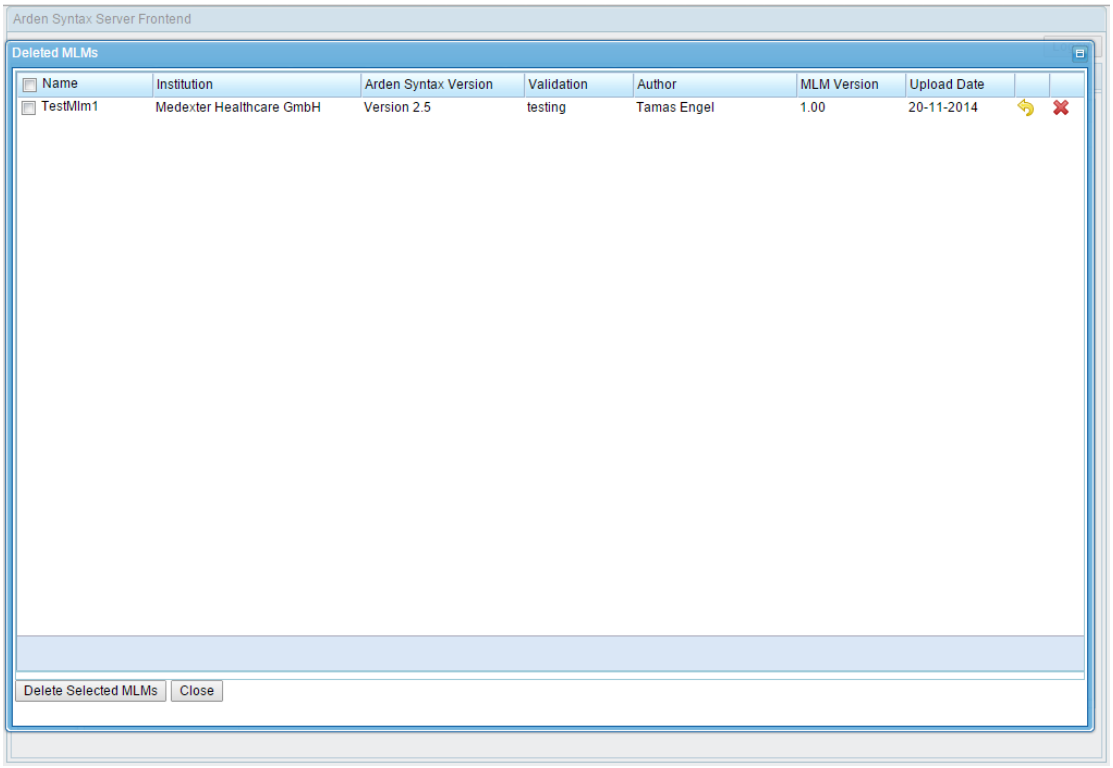

Figure 6: The "Deleted MLMs" window

3.3. User Management

The "User Management" tab allows adding, removing, and updating the Arden Syntax server's groups and user accounts (see Figure 8). Initially, the following two user groups exist (more user groups can then be created):

- Simple user: Can use MLM management and edit own account details.
- Administrator: Can additionally restore deleted MLMs and has access to statistics and user management.

The "User Management" tab is only available to the server administrators. Therefore, in order to access the window displayed below (Figure 8), you must either log in with the default or with a created administrator account (see section 2.2.1 for configuration of the administrator account). In the list in the upper screen section you can add, select, edit and delete user groups. The user accounts that belong to the selected group will be shown in the list below. To add a new group, click "Add Group" at the left bottom of the top list. In the dialogue that opens, group name, description and permissions for the group can be specified (Figure 7, e.g., the ability to call REST/SOAP service methods, database connector access, …). Each group member inherits the group's permissions. To add a new user, click "Add User" at the left bottom of the screen. An existing group or user can be modified or deleted by clicking the corresponding icon at the end of the user's or group's row.

NOTE: Avert deleting all administrator accounts! If all administrator accounts are deleted, the Arden Syntax server must be reinitialized.

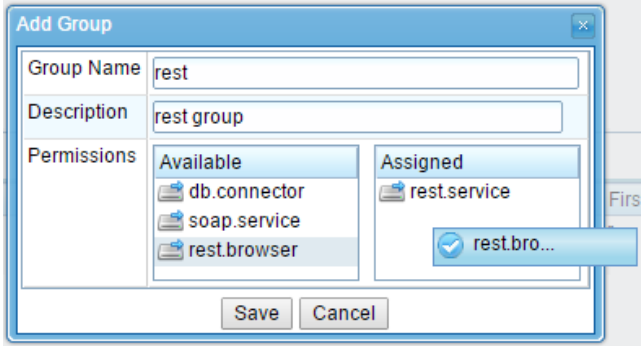

Figure 7: The "Add Group" window

Permissions may be assigned by drag and drop.

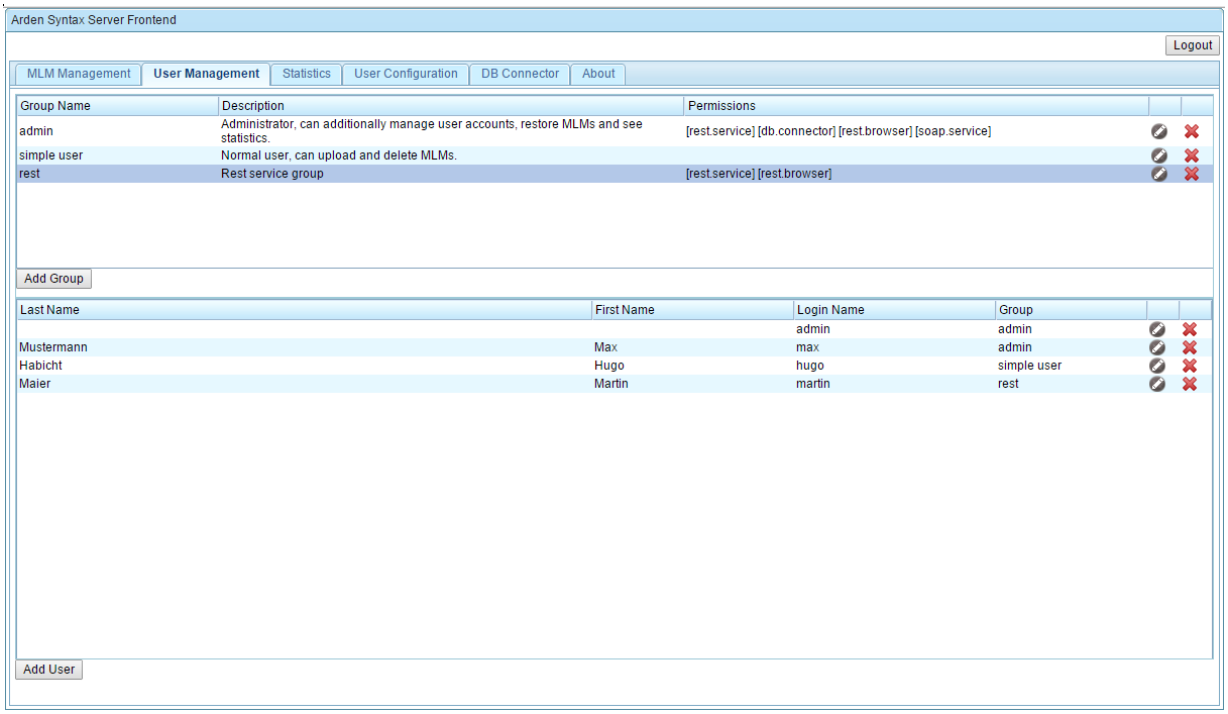

Figure 8: The "User Management" tab

Editing of own user details is possible in the "User Configuration" tab:

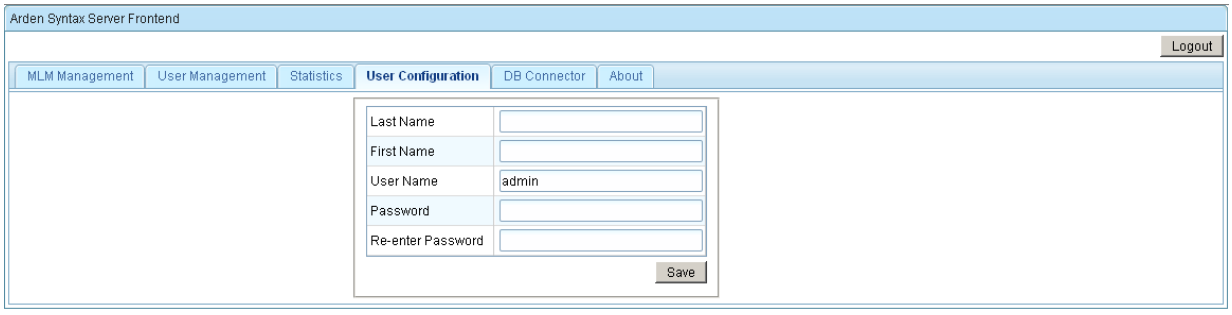

Figure 9: The "User Configuration" tab

3.4. Statistics

The Arden Syntax server further provides statistics about the usage of some internal activities (e.g., "Call MLM", "Add MLM" or "Add User") (see Figure 10). For each of these activities, the number of calls, average time spent, last access, and further information is available. The "Statistics" tab can only be accessed by administrator users:

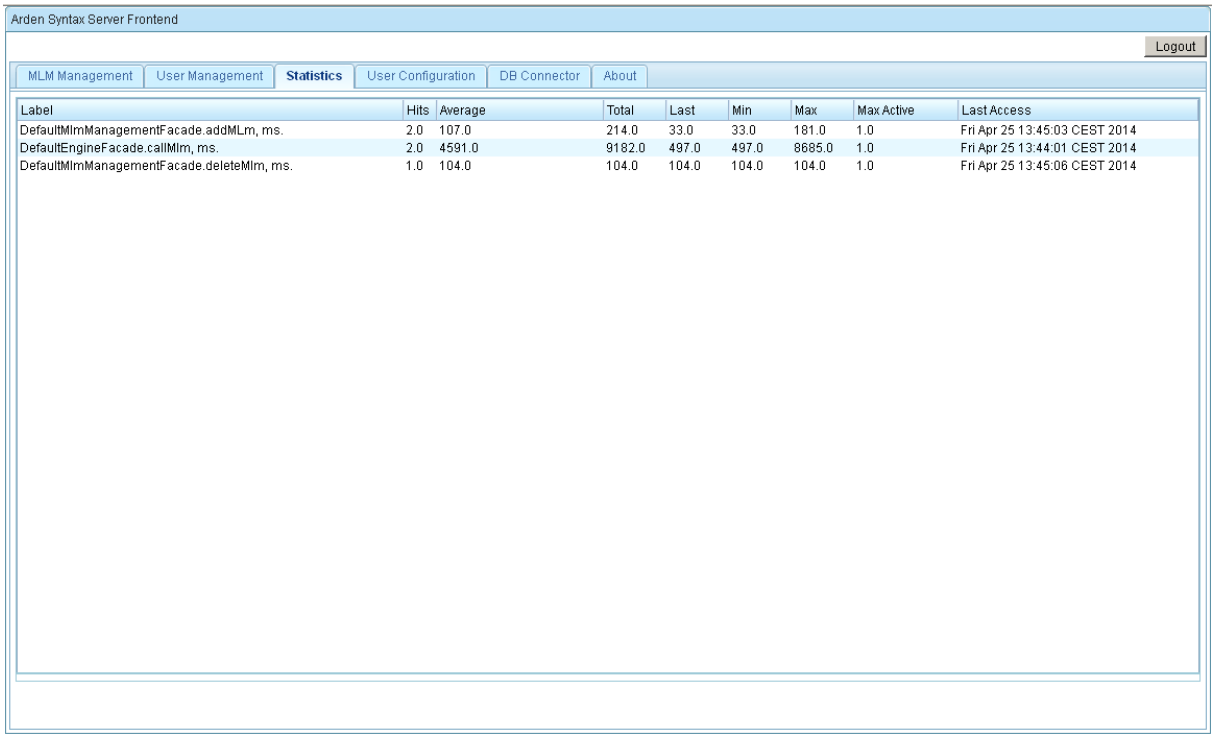

Figure 10: The "Statistics" tab

3.5. Simple MLM Overview

Based on the REST interface, the Arden Syntax Server provides an additional web front end that can be accessed using a browser via the following URL:

```
http://$SERVERNAME:8080/REST/CALLMLMS
```

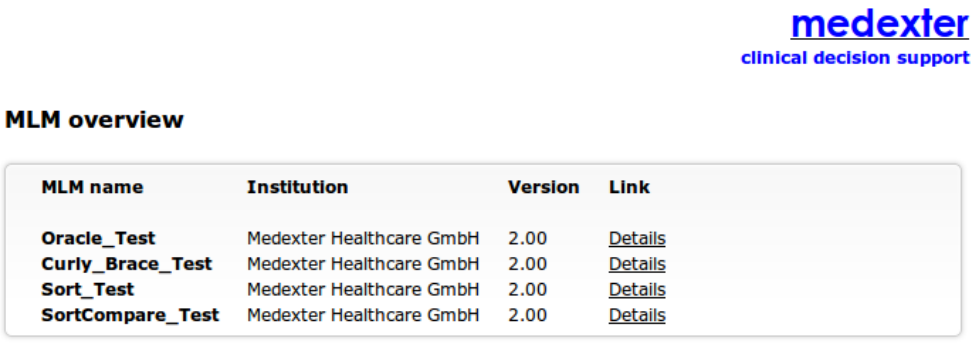

Figure 11: MLM overview using REST interface

A list containing all available MLMs is shown. By clicking on "Details", a detailed view for

the selected MLM opens. It provides the content of the the MLM's maintenance category.

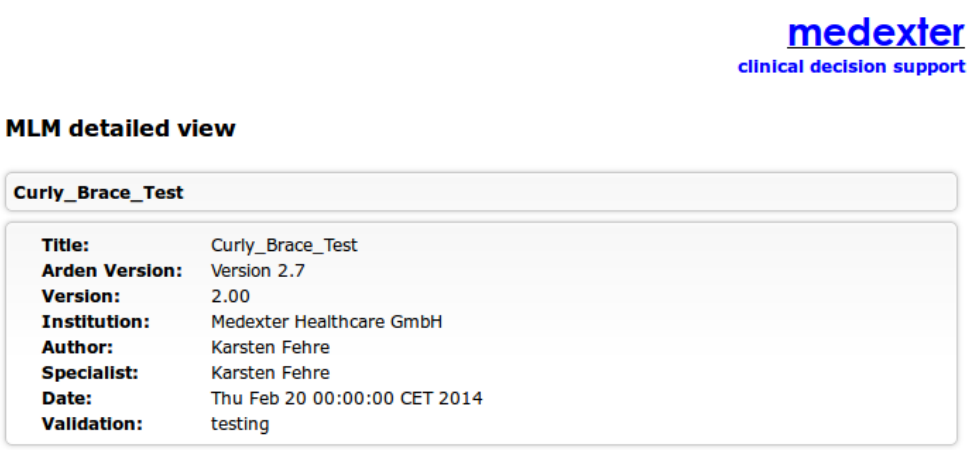

Figure 12: Detailed view for selected MLMs using REST interface

To access this simple web interface, a username and password must be provided. In order to have access to this component, users must be configured as described in chapter 2.3.

4. Developer Guide

This section contains the information necessary for use and possible extension of the Arden Syntax server in a given environment. We will describe the implementation of institution-specific data access; additionally, information will be given on how to utilize the Arden Syntax server web services within an arbitrary client application.

4.1. Evaluation of Curly Brace Expressions

The Arden Syntax server provides an external interface for the evaluation of curly brace expressions and for write operations. Figure 13 shows a schematic illustration of this situation. For the evaluation of curly brace expressions, a remote provider (via RMI – remote method invocation) is needed. You need to set the port and the IP of this provider in the configuration file (see also the shipped sample remote provider):

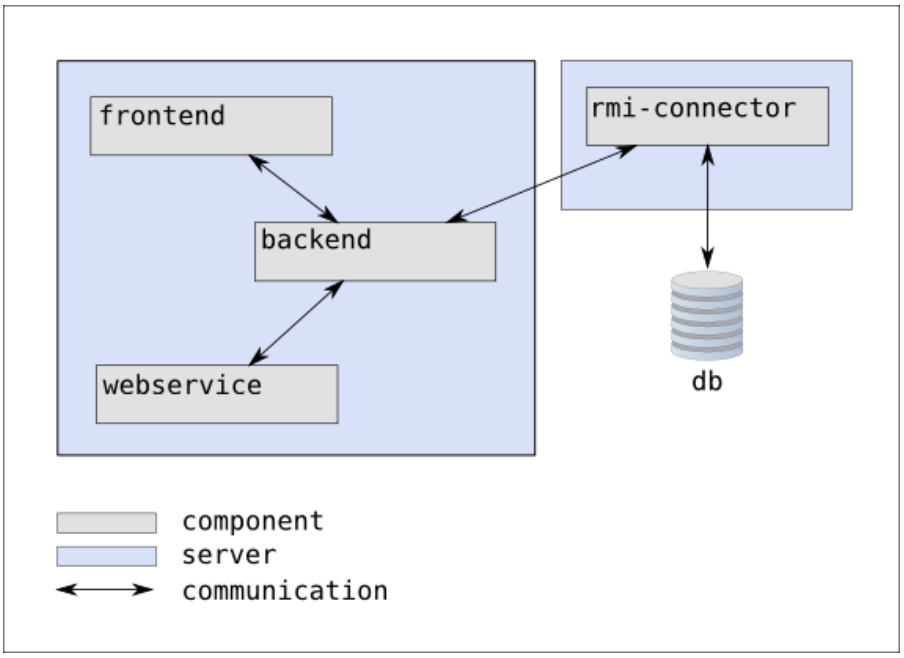

Figure 13: Arden Syntax server architecture

There are two possibilities for the implementation of the remote provider:

- As local stand-alone application
- As component to be deployed on the Arden Syntax server

4.1.1. General Considerations

Whenever the Arden Syntax engine of the Arden Syntax server has to evaluate a curly brace expression (e.g., in a read statement) the engine forwards this expression to an internal interface. The internal interface uses the configured parameter (`RmiServiceHost` and `RmiServicePort`) to locate the remote provider and sends the curly brace expression to this provider for evaluation.

Both, the stand-alone application as well as the component must implement an application interface called "`RmiHostInterface`", which enables the Arden Syntax server to communicate with the remote provider.

This application interface contains several methods (e.g., "`evaluateRead(String mappingClause)`") that correspond to actual Arden Syntax statements (e.g., read statements). That is, the curly brace expression of an executed read statement is forwarded to the "`evaluateRead`" method in the remote provider as "`mappingClause`" parameter.

The methods in this "`RmiHostInterface`" can contain any institution-specific access logic (e.g., consume remote web services, query a database, or any other kind of data access).

4.1.2. Stand-Alone Application

One way to supply the Arden Syntax server with the remote provider is to run an implementation of the "`RmiHostInterface`" as a stand-alone application. This application must be executed on a host that is available to the Arden Syntax server over a network.

A sample implementation of a stand-alone remote provider is available on our website (http://www.medexter.com/arden-syntax/arden-syntax). The archive contains a ready-to-use implementation of the "`RmiHostInterface`" with a sample implementation of one of the methods.

For an Arden Syntax "read" statement with the curly brace expression "`{this-is-a-`

test}", the sample implementation returns a list of strings. All other methods are not implemented and are returning a simple string:

```
@Override
public DataValue evaluateRead(String mappingClause) throws RemoteException {
   DataValue retVal = null;
   if("{this-is-a-test}".equalsIgnoreCase(mappingClause)) { // this is for our
test-mlm


      // Assume we get the following data from a database query
      String[] resultOfQuery = { "id", "name", "forename", "street", "city",
"postalCode",
      "phone", "email" };


      // we convert this into the internal Arden DataTypes
      ArdenList alList = ArdenList.createEmptyList(null); // result type must be a
ArdenList
      for (int i = 0; i < resultOfQuery.length; i++) {
        alList.addElement(ArdenString.createString(null, resultOfQuery[i]));
      }
      retVal = alList; // set the return value
   } else {
      retVal = ArdenString.createString(null, "not implemented yet");
   }
   return retVal;
}
```

To compile and start the sample remote provider, please carry out the following steps:

- The interface implementation "RmiHostInterfaceImpl" can be adjusted (see the given example for the evaluateRead method): [RMIProvider_Sample\src\medexter\arden\server\rmi\RmiHostInterfaceImpl.java]

- To compile the provided sources, the following command (on Windows) must be executed in a command shell in the folder where RMIProvider_Sample.zip was unpacked:

```
# javac -cp lib\asBackendAPI-2.3.0.17.jar;lib\ArdenLang-2.3.1.3.jar
src\medexter\arden\server\rmi\*.java
```

- The following command (on Windows) has to be used to create an executable jar:

```
# jar -cfvm RMIProvider.jar manifest.txt -C src \ lib\
```

- To run the created jar, please use the following command (on Windows). Subsequently, the console should show a "thread start". This indicates that the RMIProvider is ready to receive requests:

```
# java –jar RMIProvider.jar
```

If using other operating systems than Windows, please adjust the paths (see the

manifest.txt, which is also part of the above-mentioned RMIProvider_Sample.zip).

After successfully starting the application, the configuration of the Arden Syntax server must be adjusted (see 2.2.4) with the current IP and the port of this application. You then restart the Arden Syntax server and all evaluation requests for curly brace expressions are sent to this application.

NOTE: If both the Arden Syntax server and a local remote provider are running on the same host, the default remote provider must be undeployed from the Arden Syntax server.

The remote provider can be undeployed by opening the following URL:

```
http://$SERVERNAME:4848
```

Select "applications" in the left menu and then `"ArdenServerRmiService"` in the list of deployed applications. Finally, hit the "undeploy" button to remove the default remote provider from the Arden Syntax server.

4.1.3. Arden Syntax Server Component

Another way to run the remote provider is to deploy the application to the Arden Syntax server in such a way that start and shutdown of the remote provider is managed by the Arden Syntax server. Again, a sample implementation is available on our website (http://www.medexter.com/arden-syntax/arden-syntax). This component has to replace the `"ArdenServerRmiService"` dummy component that is shipped with the Arden Syntax server.

4.2. Using External Web Service Interfaces in a Client Application

The Arden Syntax server provides two external web service interfaces that can be used by an application in order to access the Arden Syntax engine capabilities (e.g., run MLMs with provided parameters). A description of both interfaces – namely REST web service and SOAP web service – will be given in the chapter at hand.

4.2.1. REST Web Service

The REST web service has two POST methods:

Call an MLM: Evaluates an MLM identified by the given parameter and is available under

the URL: http://$SERVERNAME:8080/REST/CALLMLM/

Call an event: Evokes an event (and evaluates all associated MLMs) identified by the given parameter and is available under the URL:
http://$SERVERNAME:8080/REST/CALLEVENT/

The CALLMLM method has three URL parameters that are used to identify the MLM located on the Arden Syntax server:

`"mlmName"`: The name of the MLM

`"mlmVersion"`: The version of the MLM

`"mlmInstitution"`: The institution of the MLM

The CALLEVENT method only has one parameter: `"event"`. It prompts the event to evoke.

In order to provide the evaluated MLM with an input parameter, it is possible to send data in JSON format as body to the web service methods.

The following listing shows a working example of how to consume the CALLMLM method in the programming language PHP:

```
$json_req = "data in json format";


 $uri =
"http://ardenserver.medexter.com/REST/CALLMLM/?mlmName=borreliosis_main&mlmInstitut
ion=Medexter Healthcare GmbH, Vienna, Austria";


 $uri = str_replace ( ' ', '%20', $uri );
 $ch = curl_init($uri);
 curl_setopt($ch, CURLOPT_ENCODING, "UTF-8");
 curl_setopt($ch, CURLOPT_CUSTOMREQUEST, "POST");
 curl_setopt($ch, CURLOPT_POSTFIELDS, $json_req);
 curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);    // return data/not display
 curl_setopt($ch, CURLOPT_HTTPHEADER, array(
  'Accept: application/json',
  'Content-Type: application/json',
  'Content-Length: ' . strlen($json_req))
 );
 $result = curl_exec($ch);
 curl_close($ch);
```

The variable `$json_req` should contain input data for the MLM `"borreliosis_main"`. As an example, the following listing is such a JSON string that represents a list of two Arden Syntax objects:

```
{"type":"list", "values":[
 {"type" : "object", "declaration": "parameter", "fields" : { // object 1
  "key":{"type":"string", "value": "some value"},
  "val":{"type":"datatype", "value": "some data"}
 }},
 {"type" : "object", "declaration": "parameter", "fields" : { // object 2
  "key":{"type":"string", "value": "some value"},
  "val":{"type":"datatype", "value": "some data"}
 }},
]}
```

If the client system is the web service consumer and is using JAVA as programming language, it is possible to link our API library and create the input parameter for the MLM in an object-oriented way. No explicit conversion to JSON is needed. The following listing contains a working example in the JAVA programming language:

```
String url = "http://ardenserver.medexter.com/rest/REST/myresource/CALLMLM/";


ArdenString input = new ArdenString("yourText", null, null); // sample input as
JAVA-Object


ClientConfig config = new DefaultClientConfig();
config.getClasses().add(ObjectMapper.class);
config.getClasses().add(JacksonJsonProvider.class);


Client client = Client.create(config);
WebResource webResource = client.resource(url);
MultivaluedMap<String, String> map = new MultivaluedMapImpl();
map.putSingle("mlmInstitution", "Medexter Healthcare GmbH");
map.putSingle("mlmName", "sampleMLM");


DataValue response = webResource.queryParams(map)
             .accept(MediaType.APPLICATION_JSON)
             .type(MediaType.APPLICATION_JSON)
             .post(DataValue.class, input);
System.out.println("Response " + response);
```

4.2.2. SOAP Web Service

In addition to the REST web service, the Arden Syntax server also provides a SOAP web service interface described by a WSDL (Web Service Description Language) file.

WSDL

The WSDL file represents the technical description of the service. For each client

application, the file generates the parts of the program that can access the service.

The WSDL interface description is platform independent. It can be used, for example, for creating .NET (software framework from Microsoft) client stubs. The following chapters show how a sample JAVA client application is created.

Generate Stubs from WSDL

The first step would be generating appropriate classes that abstract the access to the service, so the programmer does not have to worry about the actual network communication.

For this, the `wsimport` tool from the JDK (Java Development Kit) is required.

The necessary call is made in the Windows command line or Unix shell as follows:

```
Wsimport -p medexter.arden.server.serviceStubs -keep <WSDLUrl>
```

The above command creates JAVA classes in the current folder that can be imported into a Java project. "`<WSDLUrl>`" contains the URL or path to the WSDL service description.

Call the Service

Once the JAVA classes have been integrated into the project, the web service can be called.

The following example shows the invocation of services, including basic authentication:

```
  public static void main(String[] args) {
    Authenticator MyAuth01 = new MyAuthenticator();
    Authenticator.setDefault(MyAuth01);
    WebServiceASService serviceEndpoint;
    try {
      serviceEndpoint = new WebServiceASService(new
URL("http://localhost:8080/asXMLService/WebServiceASService?wsdl"), new
QName("http://services.server.arden.medexter/", "WebServiceASService"));
      WebServiceAS ruleEngine = serviceEndpoint.getWebServiceASPort();
      String result = ruleEngine.evaluateXMLRequest(
"<?xml version=\"1.0\" encoding=\"UTF-8\"?>" +
          "<usecase>" +
          "<callMlm>" +
          " <key>" +
          "   <mlmName>myTest</mlmName>" +
```

```
          "   <institution>Medexter Healthcare GmbH</institution>" +
          "   <version>T1000</version>" +
          " </key>" +
          " <arguments></arguments>" +
          " <payload>Ladung</payload>" +
          "</callMlm></usecase>");

      System.err.println(result);
    } catch (MalformedURLException e) {
      e.printStackTrace();
    } catch (Throwable  e) {
      e.printStackTrace();
    }
  }


  static class MyAuthenticator extends Authenticator {
    public PasswordAuthentication getPasswordAuthentication() {
      return new PasswordAuthentication("admin", "s3cret".toCharArray());
    }
  }
```

The above client is calling the method "`evaluateXMLRequest`" of the Arden Syntax server. While doing so, please ensure that the service URL is set up correctly. The method "`evaluateXMLRequest`" calls the Arden Syntax server to evaluate the given request. The request string must contain an XML document, corresponding to the Arden Syntax server protocol. The XML schema for those messages is available under:

http://www.medexter.com/phocadownload/xsd/ardenServerXMLProtocol.xsd

A request may look like this:

```
<?xml version="1.0" encoding="UTF-8"?>
<usecase>
<callMlm>
        <key>
            <mlmName>myTest</mlmName>
            <institution>Medexter Healthcare GmbH</institution>
            <version>T1000</version>
        </key>
        <arguments></arguments>
```

```
    </callMlm>
</usecase
```

The return string contains an XML document as well, with the corresponding result, for example:

```
<usecase>
  <medexter version="1.0" />
    <callMlm>
    <key>
      <mlmName>BMI</mlmName>
      <institution>Medexter Healthcare GmbH</institution>
      <version>T1000</version>
    </key>
    <arguments></arguments>
    <results><string applicability="1.0">test</string></results>
  </callMlm>
</usecase>
```

4.3. Tests with SoapUI

To test the above mentioned web services, we recommend the tool SoapUI, which is able to generate clients from WSDL files. More information about how to use SoapUI for web service calls can be found on the corresponding web site: http://www.soapui.org/.

References

[1] Health Level 7 (HL7) International (2013), The Arden Syntax for Medical Logic Systems, Version 2.9., Health Level Seven, ANSI/HL7 Arden V2.9-2013, 214pp. http://www.hl7.org/implement/standards/product_brief.cfm?product_id=290, accessed 23 December 2013.

[2] Health Level 7 (HL7) International (2013), Arden Syntax Implementation Guide, Release 1, Health Level Seven, 30pp. https://docs.google.com/document/d/1Fu0f5FIcg_ZSpTkMe04QwSNCgvakeZ7SCcfdv2A25tw/edit?pli=1, accessed 23 December 2013.

[3] Medexter Healthcare (2013), How to Write Arden Syntax MLMs – an Introduction, Medexter Healthcare, 49pp. http://www.medexter.com/phocadownload/pr/arden%20syntax%20tutorial.pdf, accessed 23 December 2013.