

FP7-287811

MobiGuide

Guiding Patients Anytime Everywhere

Collaborative projects - Large-scale integrating project (IP)

Start date: 1-Nov-2011

Duration: 48 months

Deliverable 7.4: Working prototype (including PHR with APIs, large-scale data import, KDOM extension)

Delivery due date: 31-May-2014

Actual submission date: 12-November-2014

Coordinator	HU (University of Haifa)
Deliverable Leading Partner	Atos
Contributing Partners	ZORG, HU
Revision	Version 0.13

Project co-funded by the European Commission within the Seventh Framework Programme (2007-2013)		
Dissemination Level		
PU	Public	X
PP	Restricted to other programme participants (including the Commission Services)	
RE	Restricted to a group specified by the consortium (including the Commission Services)	
CO	Confidential, only for members of the consortium (including the Commission Services)	

Table of Contents

1. Executive Summary	5
2. DataIntegrator	6
2.1 Description	6
2.2 Status check.....	7
2.3 Deployment.....	11
2.4 Folder structure	12
2.5 BaseX	12
2.6 Backup	13
2.7 Property files	13
2.8 Logs and traces.....	14
2.9 Security	15
3. Profile Server 3.0.0-RC5	16
3.1 Introduction	16
3.2 Architecture overview.....	16
3.3 Rest API screenshots.....	17
3.4 Security	17
3.5 Requirements.....	18
3.6 Installation instructions.....	18
3.7 Configuration files	19
3.7.1 flyway.properties.....	19
3.7.2 configuration.properties	19
3.7.3 logback.xml.....	20
3.8 Update instructions	20
3.9 Health check status.....	20
4. Kernel server 1.1.0-RC1	21
4.1 Introduction	21
4.2 Architecture overview.....	21
4.3 Security	21
4.4 Requirements.....	22
4.5 Installation instructions.....	22
4.6 Configuration files	23
4.6.1 configuration.properties	23
4.6.2 logback.xml.....	23
4.7 Update instructions	23
4.8 Backup and restore instructions	24
4.9 Health and status monitoring.....	24
4.10 Troubleshooting	24
5. KDOM (HU).....	25
5.1 Introduction	25
5.2 Requirements.....	27
5.3 Deployment.....	27

5.4	Configuration Files	29
5.5	Logs and traces.....	29
5.6	Security	30
5.7	KDOM client side GUI application - manual	30
5.7.1	Screens	30

List of Figures

Figure 1: MobiGuide general Architecture (in red, the DataIntegrator component)	5
Figure 2: Example of successful response to the DI check	6
Figure 3: CCP main page.....	8
Figure 4: CCP executeQuery example.....	9
Figure 5: VMR of a patient returned by the CCP	9
Figure 6: Examples of different fields values on the CCP	10

1. Executive Summary

The present deliverable is a technical report of the DataIntegrator, Kernel and KDOM components, intended to be used as a manual for deployment and maintenance of such component. It will not address any of the components internal details, as these have been already covered in other deliverables. Its target audience includes future MobiGuide IT system administrators and System Integrators.

2. DataIntegrator

2.1 Description

The DataIntegrator is a Java-based application which serves as the access-point to the PHR for the MobiGuide components' sake. The following picture shows it within the general MobiGuide Architecture:

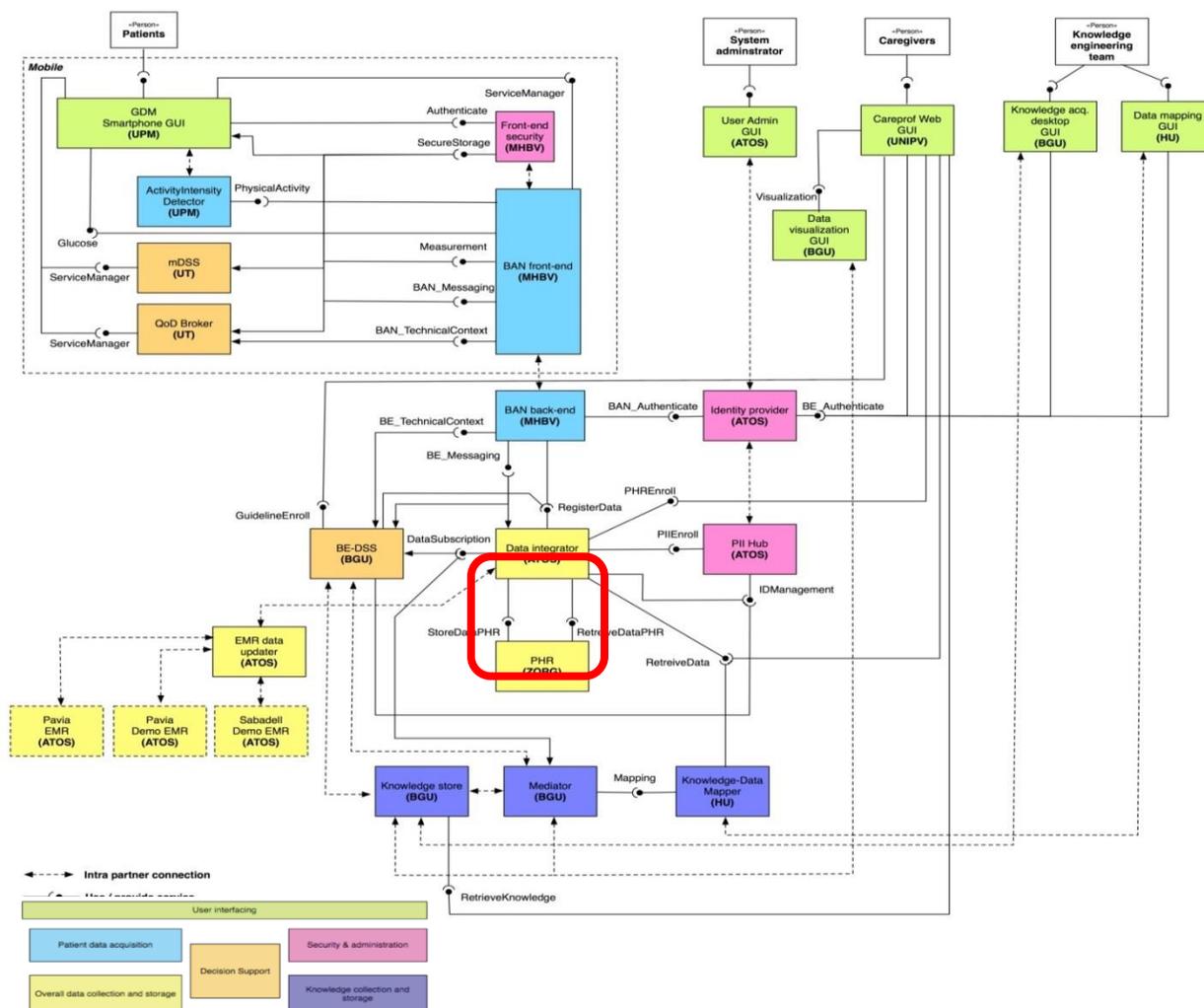


Figure 1: MobiGuide general Architecture (in red, the DataIntegrator component)

It enables the access to the data of the system by providing a uniform, VMR-based API which is accessible via HTTP calls. These calls can be roughly divided into three categories:

1. Calls to insert data into the PHR. They are accessible through the *RegisterVMRDataIF* interface.
2. Calls to retrieve data from the PHR. They are accessible through the *RetrieveVMRDataIF* interface.
3. Calls to Data Notification subscription subsystem. They are accessible through the *DataSubscriptionIF* interface.

2.2 Status check

All these interfaces must be operative to ensure that the DataIntegrator is working properly. The easiest way to check the DataIntegrator status is to call the following URL:

```
https://<SERVER>/DataIntegrator/services/RetrieveVMRDataIF/executeQuery?source=GUI&userid=<USER>&mgid=<MGID>&xQuery=db:open("PHR", "<MGID>.xml")/*:vmr&response=application/xml
```

where:

- <SERVER> is the server where the DataIntegrator has been deployed
- <USER> is a valid USER ID that is on PiiHub DataBase
- <MGID> is a valid MGID of a patient who has been already enrolled into MobiGuide

The expected result of such a call would be the following:

```
<ns:executeQueryResponse
xmlns:ns="http://interfaces.dataintegrator.atos.mobiguide">
  <ns:return>
    <mg_di:diResponse
xmlns:mg_di="http://mobiguide.atosresearch.eu/dataIntegrator">
      <mg_di:result>OK</mg_di:result>
      <mg_di:message>Query successfully
executed</mg_di:message>
      <mg_di:additionalInfo>
        <_VMR_HERE_>
      </mg_di:additionalInfo>
    </mg_di:diResponse>
  </ns:return>
</ns:executeQueryResponse>
```

Figure 2: Example of successful response to the DI check

with the <__VMR HERE__> tag being replaced by the full content of the patients VMR (only non-demographic information).

This successful call implies that:

1. The DataIntegrator server is up and running
2. The DataIntegrator is able to receive calls from the components
3. The DataIntegrator is able to access the PHR
4. The DataIntegrator is able to access the PiiHub component

This call, however, does not check:

1. If the DataNotification subsystem is working
2. If the data in the PHR is correct
3. If the demographic data in the ZORG kernel is accessible

These other conditions will have to be checked one by one afterwards. For the DataNotification, the collaboration of DSS, KDOM and Mediator components is needed, as a whole round of subscription → insertion of data → checking of the notification loop is required.

The correction of PHR data is more difficult to perform, as the DI only checks that the PHR data is syntactically correct. The semantic correctness is impossible to check in an automated way, as it requires validation by clinical experts.

Regarding the connection with the ZORG kernel, a call to the getDemographics method on a previously enrolled patient should be enough to check it.

For performing such calls the DataIntegrator contains a built-in functionality named CCP (Program Control Center), whose main page is shown on Figure 3.

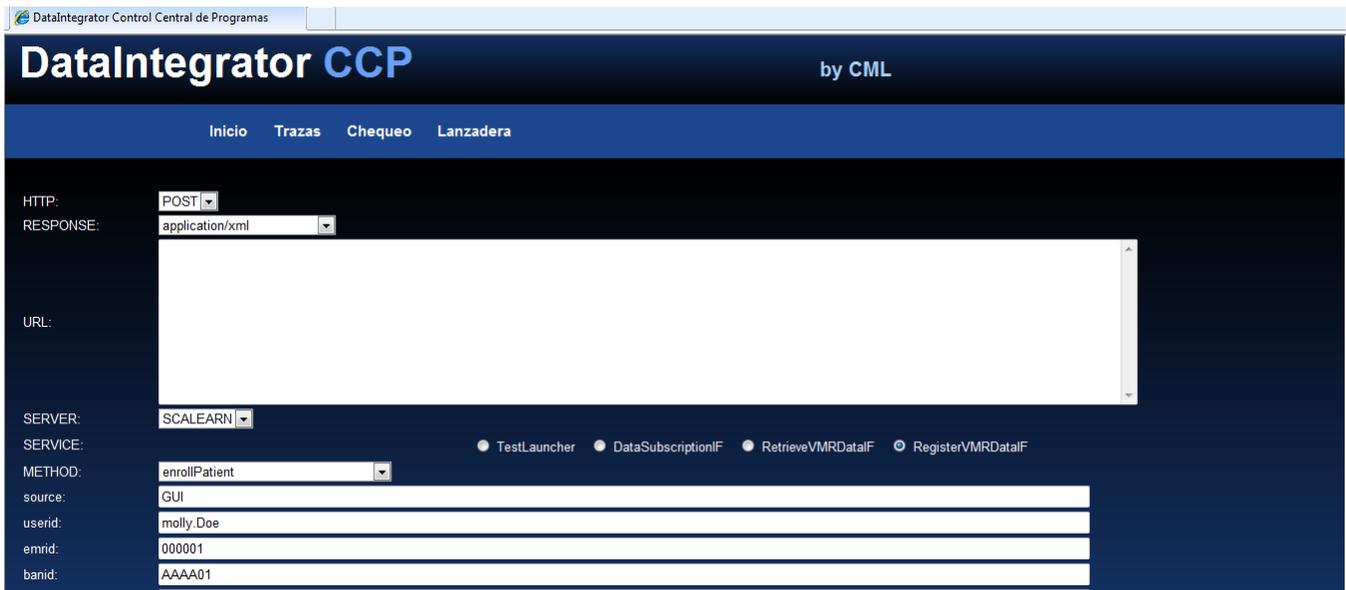


Figure 3: CCP main page

The CCP is a HTML+JavaScript web page which contains all the methods of the DataIntegrator and several examples of its attributes. By changing the “METHOD” drop-down list, the attributes of each one of the methods are shown along with concrete values for them, which eases the process of testing DataIntegrator functionality.

The CCP is strictly a development tool, not intended to be used during production phases of the project, but it can be useful for checking DataIntegrator status.

The URL for accessing CCP is -

https://<server>/DataIntegrator/html/lanzadera_crossbrowser.html being <SERVER> the name/IP of the machine where the DataIntegrator instance is being running. Once the user is authenticated and the page is loaded, the easiest way to check if the DI is running is by selecting:

- SERVICE: RetrieveVMRDataIF
- METHOD: executeQuery

Just by doing this, the CCP should show a list of all the patients enrolled so far in the system, identified by their MGID (see Figure 4).

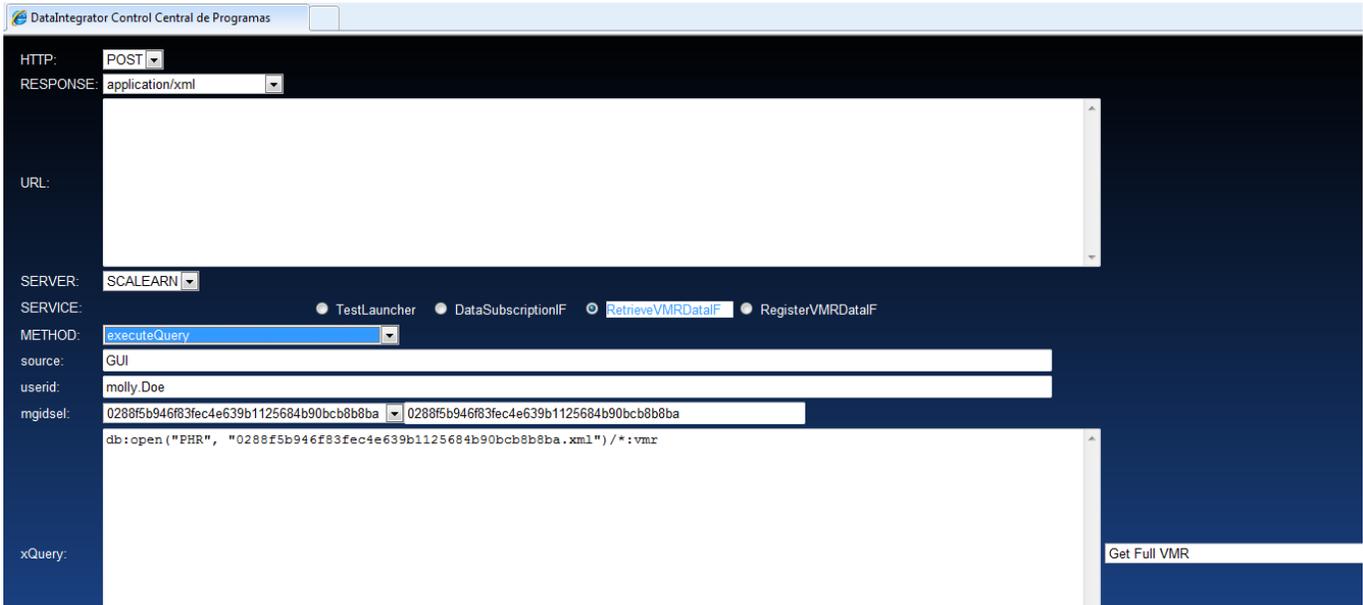


Figure 4: CCP executeQuery example

If the list is empty is because there is no patients enrolled within the system or because there has been some error in the connection between the CCP and the DataIntegrator.

Once one of the patients is selected and the user has clicked on lower button, the entire XML of the patients is returned (see Figure 5).

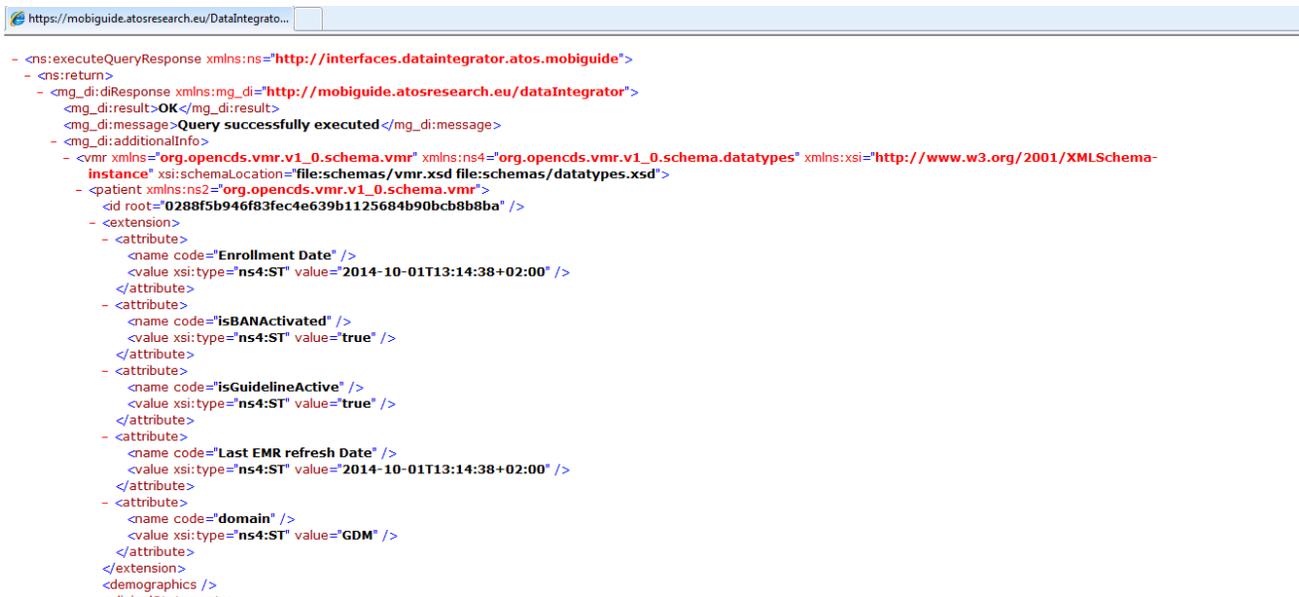


Figure 5: VMR of a patient returned by the CCP

By changing the xQuery of CCP, different actions or searches can be performed over the VMR of the selected patient (or a list of them). Any other of the DataIntegrator methods (enrolling a patient, disbanding a patient, notify of changes on data, etc.) can be achieved by selecting the appropriate menus of the CCP.

On the most used cases, the CCP provides an additional drop-down list on the right part of the page, which fills the fields with working examples of them.

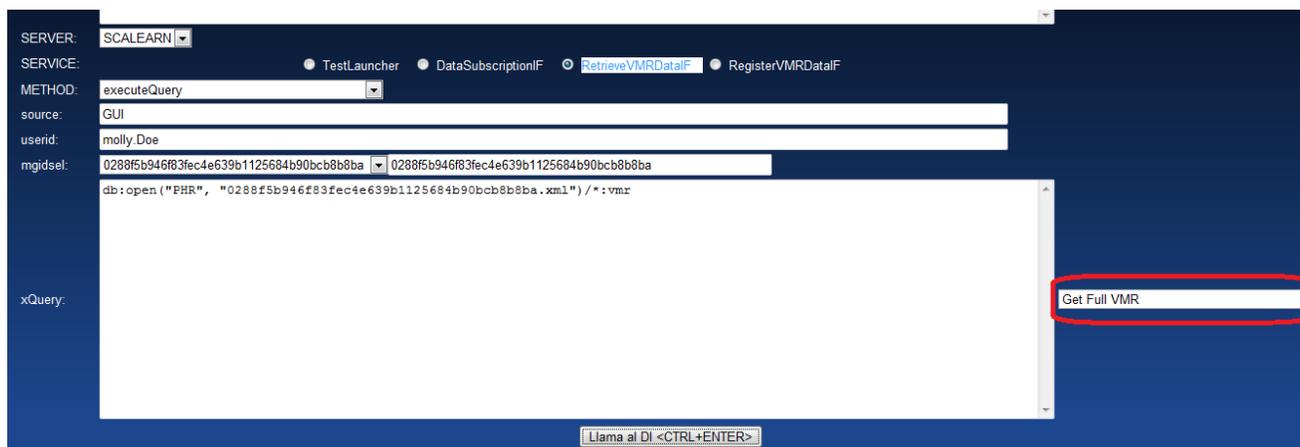


Figure 6: Examples of different fields values on the CCP

In the example given (see Figure 6), if the user selects “Get Full VMR” from the drop-down list, the CCP fills the xQuery field with the value “db:open(“PHR”, “0288f5b946f83fec4e639b1125684b90bcb8b8ba.xml”)/*:vmr”, thus saving typing time from the user.

2.3 Deployment

Usually the DataIntegrator is deployed as a WAR application. On both environments used in MobiGuide, this WAR was deployed on a regular Apache Tomcat v7.0.47 running on a Linux machine, although it could be used on other web servers/servlets containers (not tested) as well.

On both Hospitals environment the installation of a new version of the WAR is usually done by stopping the tomcat instance, replacing the WAR and starting it again, although the possibility of a hot deployment has been already tested and it should work.

2.4 Folder structure

In MobiGuide, two environments were set up for each one of the pilots. As the Architecture solution chosen is based on Virtual Machines, the result is that both environments share the same structure.

Regarding the DataIntegrator specific case, there is a dedicated Virtual Machine for it on each one of the separated environments, both of them remotely accessible by VPN connections.

Once logged in, what can be found there is the following folder structure:

- Folder `/opt/apache-tomcat-7.0.47`. This is the root folder for the Tomcat installation
- Folder `/opt/apache-tomcat-7.0.47/bin`. This is the folder where the startup/shutdown scripts are located
- Folder `/opt/apache-tomcat-7.0.47/webapps`. This is the deployment folder, where the WARs are located and also where they are deployed. Besides the DataIntegrator and BaseX WARs, it also contains the JavaDoc of DataIntegrator API.
- Folder `/var/log/tomcat`. This is the folder where DataIntegrator logs are written.

2.5 BaseX

Along with the DataIntegrator, a BaseX instance is provided to serve as PHR for the non-demographic data storage. BaseX is a XML-based database which can be set up in several ways. The one used within MobiGuide is by deployment a BaseX WAR on the same folder as the DataIntegrator, as this only action enables all of the BaseX storage capabilities that will be used in the project.

During BaseX deployment, the Database starts a process on local port 1984, which should be free for its use. During upgrades on BaseX database is quite common that a hot replacement of the WAR causes this process to hang, which could block further deployments. To fix this, is necessary to stop Tomcat server, entirely delete the BaseX instance (which, of course, empties the PHR Database of patients), manually kill any BaseX process, copy the new BaseX WAR and start the Tomcat again.

2.6 Backup

Periodic backups should be performed on the BaseX database following instructions of <http://docs.basex.org/wiki/Backups>. This is basically done by invoking the CREATE BACKUP command over PHR database. The need for a backup comes from the local laws (Implementing regulation (RLOPD) of December 21 st 2007 -Art. 102 and the Legislative Decree no. 196 of 30 June 2003) described in the D8.1 Security and Privacy Analysis in MobiGuide.

2.7 Property files

The DataIntegrator property files are located on “conf” folder within the deployment folder structure. There can be found the following files:

1. **dataintegrator.properties**. This is the main configuration file and contains several items with can be modified to change DataIntegrator behaviour:
 - `general.sources`. This is a list of the different modules that are accepted as valid DI callers.
 - `general.basex.access`. This is the way that the DI uses to connect to the underlying BaseX DataBase. 1 means REST services, 2 Java Client and 3 XQJ
 - `general.enhancedParametersLog`. This enables a more verbose log on the parameters part
 - `general.phr.basex.serverName/path/port/dbname/usuarioAdmin/claveAdmin/usuario/clave`. Those are the parameters to connect to the underlying BaseX DataBase.
 - `general.debugsec`. This enables a more verbose log on security part
 - `general.idp.activate`. This is a switch that enables/disables the use of IDP security component
 - `general.idp.server/port/path`. Those are the parameters to access the IDP security component
 - `general.piihub.activate`. This is a switch that enables/disables the use of PiiHub security component
 - `general.piihub.server/port/path`. Those are the parameters to access the PiiHub security component
 - `general.piihub.trustStore.path/pass`. This is the access path (and password) to the TrustStore where the certificates are stored
 - `general.piihub.keyStore.path/pass`. This is the access path (and password) to the KeyStore where the certificates are stored

- `general.emr.pavia.server/port/path`. In the case of FSM EMR, those are the parameters to connect to its webservice
 - `general.emr.sabadell.pathtofiles`. In the case of Sabadell EMR, this is the path to where the EMR files are located.
 - `general.kernel.enable`. This is a switch that enables/disables the use of ZG kernel component
 - `general.kernel.server/path`. Those are the parameters to access ZG kernel component
2. **log4j.properties**. This file contains the configuration information of the Log4j system.
 3. **skeleton.xml**. This is a template xml for the creation of the VMR.

2.8 Logs and traces

The DataIntegrator contains a fully working logging system that will enable a system administrator to detect errors on the normal functioning of the MobiGuide system. The `log4j.properties` file can be modified to provide more detailed logs. For now, six log files are configured:

- **DataIntegrator.log**. This file contains all the logs generated by the application. It should only be enabled during testing, as it can significantly affect the performance.
- **DataIntegratorMG.log**. This file contains all the logs generated exclusively by the DataIntegrator packages and the classes called from them. It can be disabled during production.
- **DataIntegratorErr.log**. This file only contains the errors and warnings generated by the application. It should be examined from time to time to ensure that there are no significant errors.
- **DataIntegratorSummary.log**. This file contains a summary of the calls done to the DataIntegrator by external components. It can be useful for system administrator as it shows a trace of the actions of the entire MobiGuide system (that involve the DI)
- **DataIntegratorNotifier.log**. This is a file that only contains the Notifier subsystem logs, so they can be examined differently from the rest of the system.
- **DataIntegratorPerformance.log**. This file contains the execution times of each of the calls received by the DataIntegrator and also those done by it to external system. It can be used to detect bottlenecks and reasons for low performance.

In addition to these logs, the DataIntegrator sends a special trace to the Security components for audit logs: each time a component tries to access the PHR data, this is recorded. Such traces can be seen later on the Security components console.

2.9 Security

All the connections to/from the DataIntegrator are done through SSL/HTTPs secure channels. The DataIntegrator will not accept calls coming through unsecure channels, which is done relying on the Tomcat container security.

Following security recommendations, the demographic and non-demographic calls are separated at all levels, including the physical, as they are stored on the kernel component (demographic data) and the BaseX DataBase (non-demographic data). The BaseX DataBase includes no encryption capabilities, so some adequate protection should be provided on its files at the Operating System level. Following Security Directives, the entire Virtual Machine should be then encrypted by using Linux built-in capabilities.

Regarding the DI internal MySQL DataBase, it only stores notification subsystem-related data, so no additional security is needed.

3. Profile Server 3.0.0-RC5

3.1 Introduction

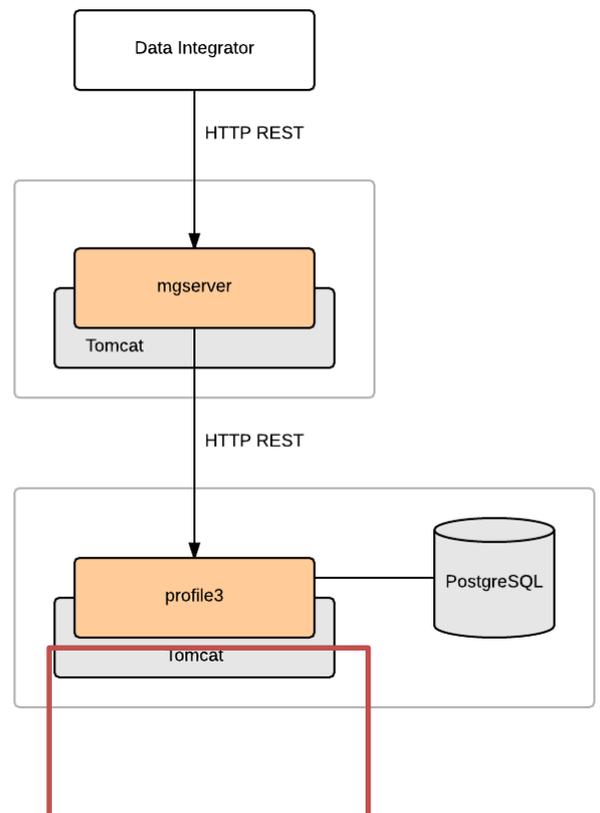
This document describes how to install the profile server version **3.0.0-RC5** as in use by the MobiGuide project. The profile server stores the demographics information and is used as storage backend for the MobiGuide server.

3.2 Architecture overview

The system used by MobiGuide consists of the following components:

- Postgresql server
- profile server
- MobiGuide server

The profile server takes care of storing and retrieving demographic information in the PostgreSQL server.



3.3 Rest API screenshots

500 (INTERNAL_SERVER_ERROR) An unhandled error occurred.

503 (SERVICE_UNAVAILABLE) A service on which this resource depends is not available.

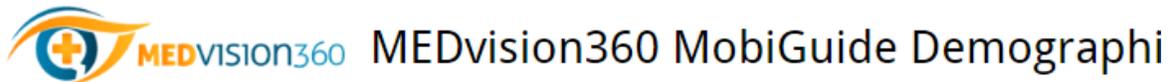
[Try it out!](#) [Hide Response](#)

Request URL

http://mgserver.test.medvision360.org/mgserver/demographics/9c5174700f599f1d5e97f3f4e83ba22ac73d2c98

Response Body

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="no"?>
<vmr xmlns="org.opencds.vmr.v1_0.schema.vmr" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="file:schemas/vmr.xsd">
  <patient>
    <demographics>
      <name xmlns:ns2="org.opencds.vmr.v1_0.schema.vmr">
        <part type="GIV" value="PRUEBA"/>
        <part type="FAM" value="PRUEBA2"/>
        <part type="FAM" value="PRUEBA3"/>
      </name>
    </demographics>
  </patient>
</vmr>
```



Use this page to explore and **try** various API functions. Some functions require login. See [this page](#) for more information.

[BASE URL: <http://mgserver.test.medvision360.org/mgserver> , API VERSION: 1.1.0-rc1]

/demographics

DELETE	/demographics/{id}
GET	/demographics/{id}
PUT	/demographics/{id}

3.4 Security

The profile server does not provide any means of restricting access itself. It is up to the systems administrator to restrict the access to the database and the REST API of the profile server when needed by using the methods provided by the operating system, postgresql, tomcat and/or a firewall.

For the MobiGuide project the profile server is only accessible by the MobiGuide server.

3.5 Requirements

The profile server is tested using the following configuration:

- Ubuntu 12.04
- PostgreSQL 9.3
- Tomcat 7.0.34

However, the profile server does work with other operating systems, application servers and other versions of PostgreSQL.

3.6 Installation instructions

The easiest way to install the profile server is by installing the provided Debian package. This package works on most Debian based Linux distributions, including vanilla Debian.

The Debian package assumes a working Tomcat installation using the default location (`/var/lib/tomcat7/webapps`) for deploying WAR files. If using the Debian package based installation method is not an option, the WAR file can be deployed manually.

Use the following steps to install the profile server:

- install the Debian package

```
dpkg -i mv-profile3-3.0.0-RC5.deb
```

Because no configuration exists, the installation will end with a warning the configuration does not exist yet.

- create the database
- `cat <<EOF | sudo -u postgresql psql`
- `CREATE ROLE profile3 WITH LOGIN PASSWORD 'profile3';`
- `CREATE DATABASE profile3 ENCODING 'UTF-8' TEMPLATE template0;`
- `GRANT ALL PRIVILEGES ON DATABASE profile3 TO profile3;`

EOF

- configure the profile server

Several example configuration files (see following section for detailed description) are created in `/etc/medvision360/profile3`

Copy these templates and modify them as needed and run the setup script again:

- `cd /etc/medvision360/profile3`

- for i in *in; do cp \$i `basename \$i .in`; done
- vi configuration.properties
- vi flyway.properties
- vi logback.xml

```
/var/lib/dpkg/info/mv-profile3.postinst
```

- restart tomcat7
- After every change of the configuration file tomcat needs to be restarted:

```
service tomcat7 restart
```

3.7 Configuration files

The profile server uses the following configuration files:

3.7.1 flyway.properties

- This configuration file is used during installation and upgrade of the profile server to migrate the database to a newer version. It contains information, which is used to connect to the database. If the database is created using the default names and passwords, the template version provided by the installer is correct.
- The most important options in this file are: `flyway.url`, `flyway.user` and `flyway.password`
- These options are explained in the comments surrounding these options in the configuration file.

3.7.2 configuration.properties

This file contains the configuration of the profile server and must be modified as needed:

option	description	suggested value
<code>jdbc.user</code>	The database user used to connect to the database	<code>profile3</code>
<code>jdbc.password</code>	The password used to connect to the database	<code>profile3</code>
<code>jdbc.url</code>	The location of the database	<code>jdbc.url=jdbc:postgresql://localhost/profile3</code>

<code>lib.server.config.logback onfigfile</code>	The location of the logging configuration file	<code>/etc/medvision360/profile3/ logback.xml</code>
<code>gaping.hole.bypass.google. for</code>	Unused option left in for compatibility reasons.	

3.7.3 logback.xml

- By default this file configures the profile server to create two logfiles. These logfiles are `/var/log/medvision360/profile3/errors.log` and `/var/log/medvision360/profile3/access.log`
- The `errors.log` file contains generic log messages of the server. The `access.log` file contains an Apache like access log.
- By default the log level is set to debug, the logfiles are rotated every day and deleted after 30 days. Please see the logback manual for more information on how to configure logging.

3.8 Update instructions

- To upgrade the application, follow the normal installation instructions and install the new Debian package over the old version. The Debian package will take care of migrating the database if needed.
- When upgrading it is not necessary to alter the configuration files or run the install script manually.
- It is recommended to periodically backup the PostgreSQL database. This can be done by using tools like [autopostgresqlbackup](#)
- There are no special instructions required regarding backing up or restoring data.

3.9 Health check status

- To check if the system is running point a webbrowser at:
`http://<yourserver>:<yourport>/profile3/apidocs/`
- The webbrowser should open the API documentation and provide means to test the APIs.

4. Kernel server 1.1.0-RC1

4.1 Introduction

This document describes how to install the ZORG Kernel version 1.1.0-RC1 as in use by the MobiGuide project. The MobiGuide server converts the XML files from the data integrator and allows the demographic data to be stored in the profile server.

4.2 Architecture overview

The system used by MobiGuide consists of the following components:

- Postgresql server
- profile server
- MobiGuide server

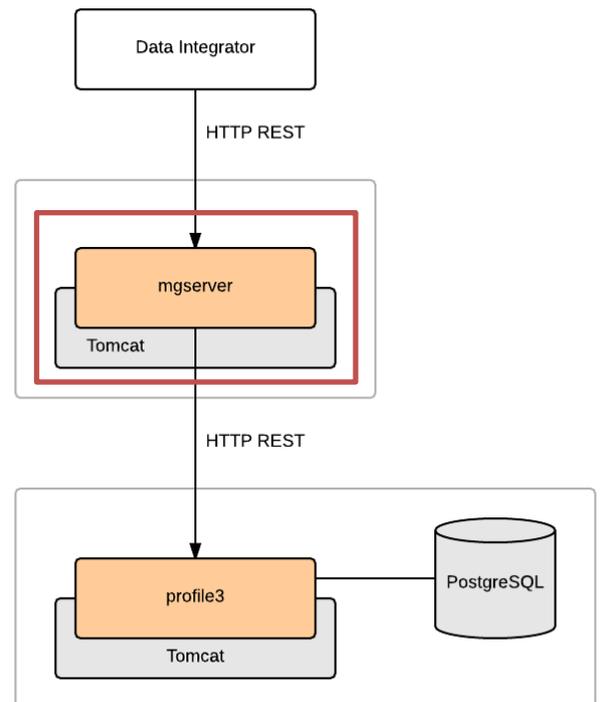
4.3 Security

The MobiGuide server (Kernel) does not provide any means of restricting access. It is up to the systems administrator to restrict the access to the REST API of the MobiGuide server when needed by using the methods provided by the operating system, tomcat and/or a firewall.

SSL protection is provided by an Apache proxy in front of tomcat. This Apache proxy only allows connections using a valid SSL client certificate to pass. Setting up this Apache proxy is beyond the scope of this document and depends on the available infrastructure and certificate provider.

Details about the installation process on the FSM servers can be found in MedVision's MobiGuide Run Book and

MP-5 - Installation our software on the new virtual server in the hospital REVIEW.



4.4 Requirements

The profile server is tested using the following configuration:

- Ubuntu 12.04
- Tomcat 7.0.34

However, the MobiGuide server does work with other operating systems and application servers.

4.5 Installation instructions

The easiest way to install the MobiGuide server is by installing the provided Debian package. This package works on most Debian based Linux distributions, including vanilla Debian.

The Debian package assumes a working Tomcat installation using the default location (`/var/lib/tomcat7/webapps`) for deploying WAR files. If using the Debian package based installation method is not an option, the WAR file can be deployed manually.

Use the following steps to install the profile server:

- install the Debian package

```
dpkg -i mv-mgserver-1.1.0-rc1.deb
```

- configure the mobiguide server

Several example configuration files are created in `/etc/medvision360/mgserver`

(see more detailed description in the following section). Copy these templates and modify them as needed:

- `cd /etc/medvision360/mgserver`
- `for i in *in; do cp $i `basename $i .in`; done`
- `vi configuration.properties`

```
vi logback.xml
```

- `restart tomcat7`

After every change of the configuration file tomcat needs to be restarted:

```
service tomcat7 restart
```

4.6 Configuration files

The MobiGuide server uses the following configuration files:

4.6.1 configuration.properties

This file contains the configuration of the mobiguide server and must be modified as needed:

option	description	suggested value
<code>profile3.url</code>	The base URL where the profile3 server is located.	<i>none</i>
<code>lib.server.config.logbackconfiguration</code>	The location of the logging configuration file	<code>/etc/medvision360/mgserver/logback.xml</code>

4.6.2 logback.xml

- By default this file configures the profile server to create two logfiles. These logfiles are `/var/log/medvision360/mgserver/errors.log` and `/var/log/medvision360/mgserver/access.log`
- The `errors.log` file contains generic log messages of the server. The `access.log` file contains an Apache like access log.
- By default the log level is set to debug, the logfiles are rotated every day and deleted after 30 days. Please see the logback manual for more information on how to configure logging.

4.7 Update instructions

- To upgrade the application, follow the normal installation instructions and install the new Debian package over the old version.
- When upgrading it is not necessary to alter the configuration files.

4.8 Backup and restore instructions

Since the MobiGuide server does not store information, there is no data, which needs to be backed up.

4.9 Health and status monitoring

- To check if the system is running point a webbrowser at:
`ttp://<yourserver>:<yourport>/mgserver/apidocs/`
- The webbrowser should open the API documentation and provide means to test the APIs.

4.10 Troubleshooting

Please check the log file `/var/log/medvision360/mgserver/errors.log` for error messages. If there is nothing shown in the file, it means Tomcat failed to start the application. In this case, check the Tomcat log files.

5. KDOM (HU)

5.1 Introduction

The Knowledge-Data Ontological Mapper (KDOM) is a Java-based application for mapping computerized clinical guidelines to electronic medical records. In the MobiGuide project, KDOM was implemented in the following structure:

Client Side

The KDOM is Java-based standalone application with a desktop GUI, which allows the definition and management of mappings used in the following described services.

Server Side

The KDOM is a set of several Java-SOAP based web service (WS) applications:

- **Data Retrieval Service** – The Mediator component (BGU) uses KDOM for retrieving data from the PHR files xml based database, managed by the DataIntegrator component (Atos).
- **Notifications Service** – The Mediator component (BGU) uses KDOM to subscribe for notifications regarding new data arrivals into the PHR files xml based database, managed by the DataIntegrator component (Atos).
- **KDOM Managerial Service** – The KDOM GUI application uses this service to manage the mappings used on the server side services.

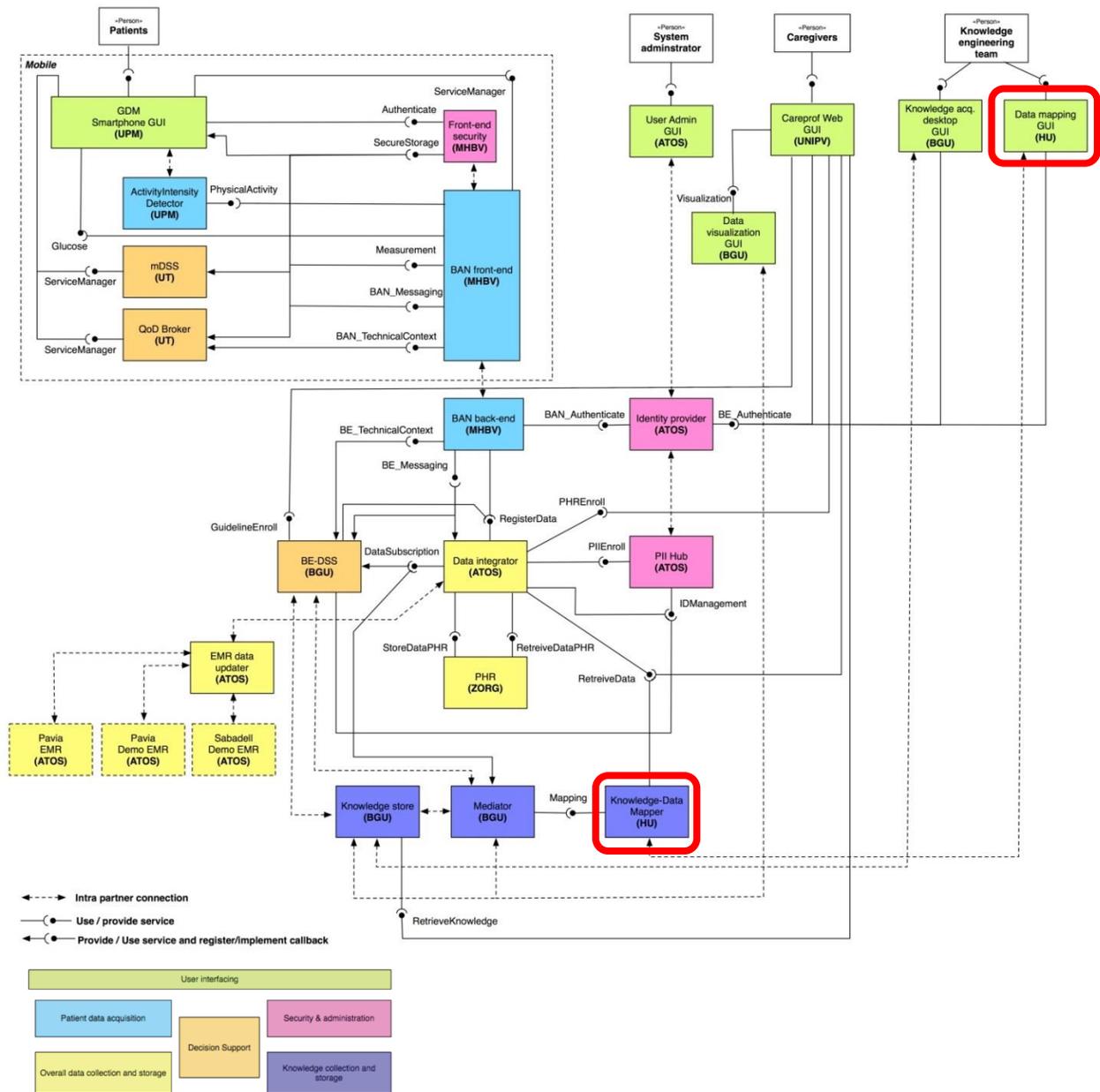


Figure 5: MobiGuide general Architecture (in two red squares, the KDOM server and client sides applications)

5.2 Requirements

The KDOM server side application (all web-services) is using and was tested at the two hospital sites with the following configuration:

- Ubuntu 12.04.3
- Tomcat 7.0.47
- OpenJDK Runtime Environment (IcedTea6 1.12.6) (6b27-1.12.6-1ubuntu0.12.04.4)

It was tested also locally (University of Haifa server) with the following configuration:

- Ubuntu 13.04
- Tomcat 7.0.47
- OpenJDK Runtime Environment (IcedTea 2.4.4) (7u51-2.4.4-0ubuntu0.13.04.2)

It can be deployed (not tested) on any operating systems and application servers.

The KDOM GUI application is using and was tested on:

- Windows 7 64-bit
- Java(TM) SE Runtime Environment (build 1.8.0-b132).

5.3 Deployment

The KDOM server side is combined of several web-services, each of the services should be deployed as a WAR application. It can be deployed without stopping the tomcat server (hot deployment), or by stopping and starting it after the deployment (cold deployment). Both ways were tested locally and in the two hospital sites.

In MobiGuide, the system architecture is based on several virtual machines (each for different components). This system architecture was deployed in two environments (two hospital sites), thus both environments share the same structure.

The KDOM server side application is deployed on a dedicated virtual machine in both environments, it's remotely accessible by a VPN connection. The following folders structure can be found in both environments:

- Tomcat's root folder: /opt/apache-tomcat-7.0.47.
- Tomcat's startup\shutdown scripts folder: /opt/apache-tomcat-7.0.47/bin.
- Tomcat's web-applications deployment folder: /opt/apache-tomcat-7.0.47/webapps.
- Tomcat's logs folder: /opt/apache-tomcat-7.0.47/logs.

- KDOM's logs folder: /home/haifauni/MG/KDOM/{kdom_environment}/logs.
- {kdom_environment} is one of the following: qa, dev, prod.
- KDOM's configurations folder: {AppPath}/configurations.

Along with the KDOM server side application, a BaseX XML-based database is installed to serve as KDOM's mappings storage. BaseX can be installed in several ways, we've chosen the following on our Ubuntu machines:

- 1) Run: apt-get install basex
- 2) Run: basexserver start -S -p1983

The second step starts a process on local port 1983, which should be free for its use.

On BaseX update:

- 1) The process should be stopped with: basexserver stop
- 2) Update BaseX with: apt-get
- 3) Run: basexserver start -S -p1983

The KDOM client side application should be installed on a machine having at least JRE8 (build 1.8.*), because it's developed based on JavaFX. It connects to the KDOM server side managerial WS via a VPN connection (as specified by the IT staff of each of the hospital sites) with proper credentials and certificates, it also uses a SSH tunnel with local port forwarding and the following configurations (set in the KDOM_GUIconfig file):

Connect FSM:

- Host: 10.7.59.121
- Port: 2255
- Protocol: SSHv2
- Tunnel local port: 9999
- Tunnel remote host: 192.168.0.5
- Tunnel remote port: 9090

Connect CSTP:

- Host: 10.24.0.30
- Port: 2255
- Protocol: SSHv2
- Tunnel local port: 9999
- Tunnel remote host: 192.168.0.5
- Tunnel remote port: 9090

The KDOM client side application connects the KDOM Managerial WS via SSL, thus a SSL HTTP/1.1 Connector in the Tomcat7 server.xml configuration file should be defined on port 9090 (or any other "Tunnel remote port").

5.4 Configuration Files

The KDOM WS includes the following configuration files:

- **KDOMconfig.xml** configuration file, the file is within the "configurations" folder under the deployed KDOM WS app folder. It is the main configurations file of the application and it's scanned every 30 minutes for changes.
- **Logback.xml** configuration file, the file is within the "configurations" folder under the deployed KDOM WS app folder. It configures the LOGBack framework's default settings (e.g. Appenders) this file is scanned every 30 minutes for changes.

The KDOM GUI includes the following configuration files:

- **KDOM_GUIconfig.xml** configuration file, the file should be placed in the same folder of the GUI application. The file is scanned in each GUI app execution.

5.5 Logs and traces

The KDOM server side contains a logging mechanism based on the LOGBack framework, which allows the system administrator to detect warnings and errors of the deployed KDOM component (by default it is configured to log warnings and errors, could be changed in the logback.xml configuration file). Three log rolling by day files are configured (removed automatically after 30 days):

- **Kdom_login-{date}.log**: This file contains logs of all KDOM's client components that tried to use its services.
- **Kdom_notification-{date}.log**: This file contains logs of KDOM's notifications subscription service.
- **Kdom_retrieval-{date}.log**: This file contains logs of the retrieval service.
- **Kdom_managerial-{date}.log**: This file contains logs of the managerial service.

5.6 Security

All the connections to/from the KDOM server side web-services are done through SSL/HTTPS secure channels. The KDOM services will not accept calls coming through unsecure channels (accomplished by Tomcat container security). Furthermore, an OpenAM Apache Tomcat Policy Agent is installed at KDOM hospitals server side machines. The agent filters out any unauthorized calls directed to the KDOM services.

The KDOM client side GUI application has an authentication process for its users when they login into the application; the authentication is done by the MobiGuide IDP (Atos) component, which returns an authentication token to be used with the policy agent.

The BaseX DB includes no encryption capabilities, so some adequate protection should be provided on its DB files at the Operating System level. Following Security Directives, the entire Virtual Machine should be then encrypted by using Linux built-in capabilities.

5.7 KDOM client side GUI application - manual

The KDOM client side standalone application is implemented with a desktop JavaFX based GUI.

5.7.1 Screens

Login Screen

Execute KDOM client side application by double clicking the kdom_v#.##.jar file, a login screen should be opened:



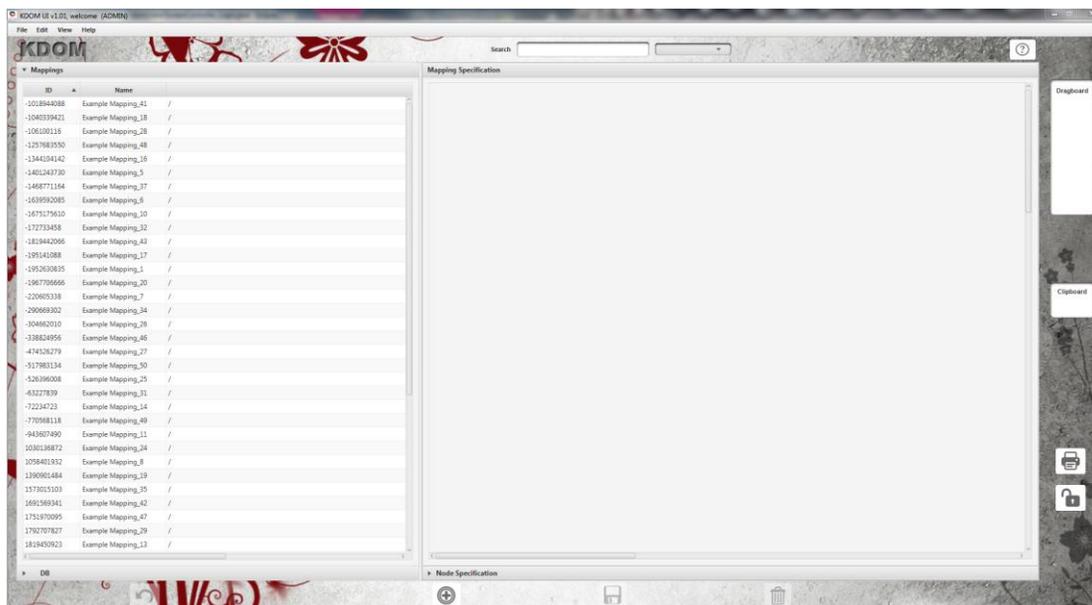
- 1) **User** – The user name mandatory field, should be filled by the user to login into the system.
- 2) **Password** – User's password mandatory field, should be filled by the user to login into the system.
- 3) **Choose Environment** – A combo box that holds a list of all the environments which the system can connect and manage mappings for, the user should pick the desired environment.
- 4) **Login** - A login button. should be pressed to initiate a login process attempt.

The user (e.g. mapping definition expert) should insert her credentials (i.e. user name & password) and choose the desired environment that he\she wants to connect for the purpose of managing its mappings:

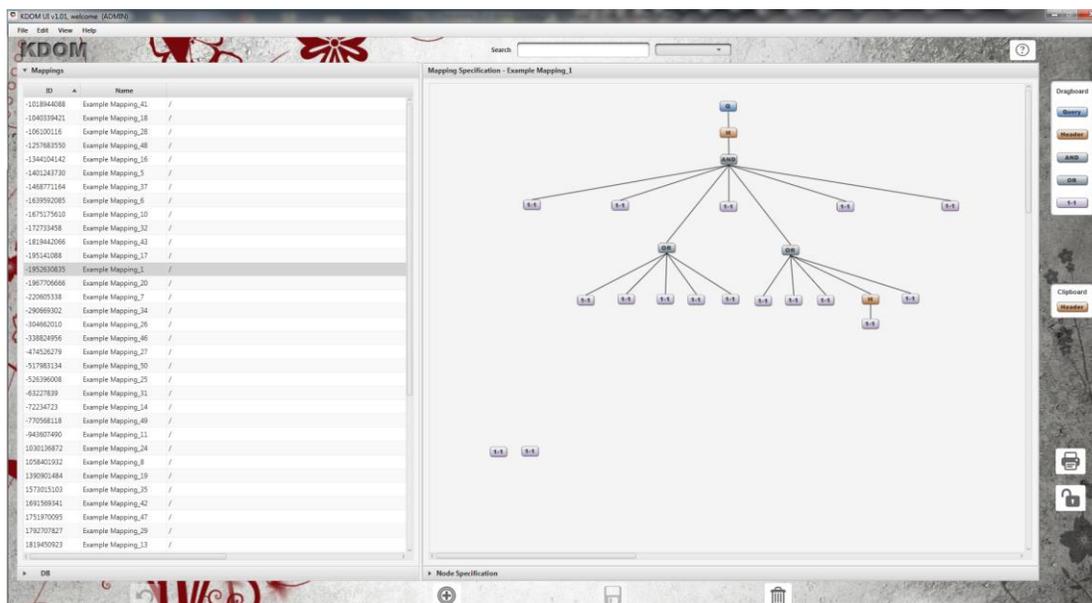


Mappings Management\Definition Screen (Main Screen)

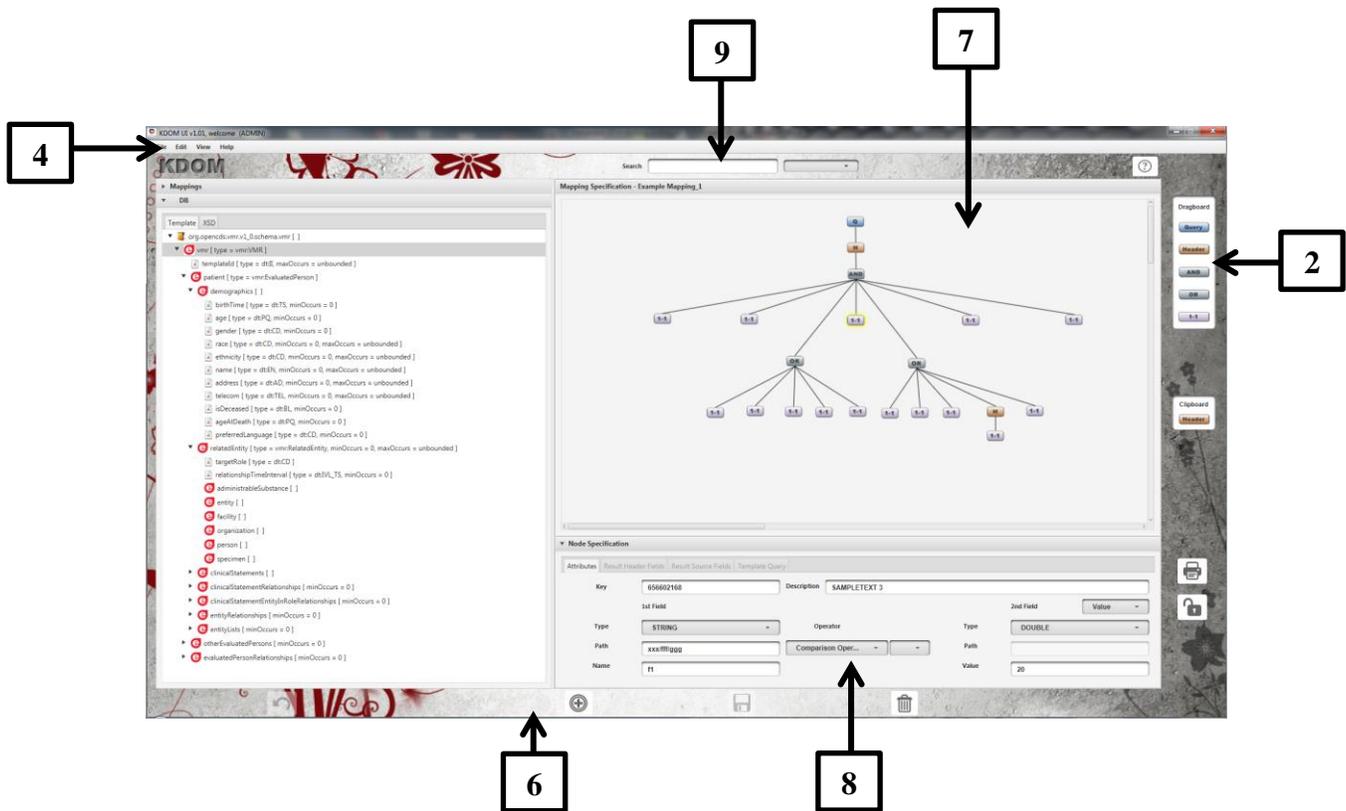
After a successful login, the following mappings management\definition screen will open; this is the main screen of the system. At first, no mapping will be selected from the list of mappings on the left, thus no mapping definitions will be shown in the mapping specification area on the right.



When a mapping is selected in the left mappings panel, its mapping specification will be shown in the mapping specification area on the right:



When a node is selected in the mapping specification, its node specification will be shown on the bottom of the screen (will slide in), the selected node will be marked in yellow and the database (DB) XML template will be shown on the left (will slide in):



1) **Left panel** – shows a container with two sliding panels, on which the user can click the header to open one of the following:

- **Mappings** - a sliding panel with a table that shows a list of all the defined mappings from the environment the system is connected to.
- **DB** - It's a sliding panel with few tabs that show different information regarding the data base (DB), which the defined mappings should map to:
- **Template** – A generated xml template tree view from the database XML schema definition (XSD) file. The use can drag and drop the different elements into other parts of the system (will be explained).
- **XSD** – The XML schema definition of the database that the mappings are defined to me mapped to.

2) **Dragboard** – A board that shows the different node types that can be created by dragging and dropping them into the mapping specification area. The nodes will be shown only when a new node can be added to the mapping specification (i.e. when a mapping is selected from the mappings table and it doesn't have unsaved changes). All nodes can be connected to one "parent" upper node, but can be connected to different quantity of sub-nodes according to its type:

- **Query node** - The main node of each mapping, its created automatically when a new mapping is created (see the new mapping creation sequence section) and cannot be deleted. Or can be drag and dropped into an existed



mapping as a sub-query node (see the root query and sub- query section).
Can be connected to one sub node.

- **Header node** – A node that can be connected to one sub-node, it's a logic condition definition node on a sub tree of nodes (e.g. to add a negation logic of "not" to a sub tree of nodes).
- **AND node** – A logic AND connection between sub-nodes, can be connected to **OR node** – A logic OR connection between sub-nodes, can be connected to many sub-nodes.
- **1-1 node** – It's a "leaf" node that cannot be connected to any sub-nodes. This node defines a criterion for the mapping (e.g. a certain field's attribute's value should be bigger or equal to '3').

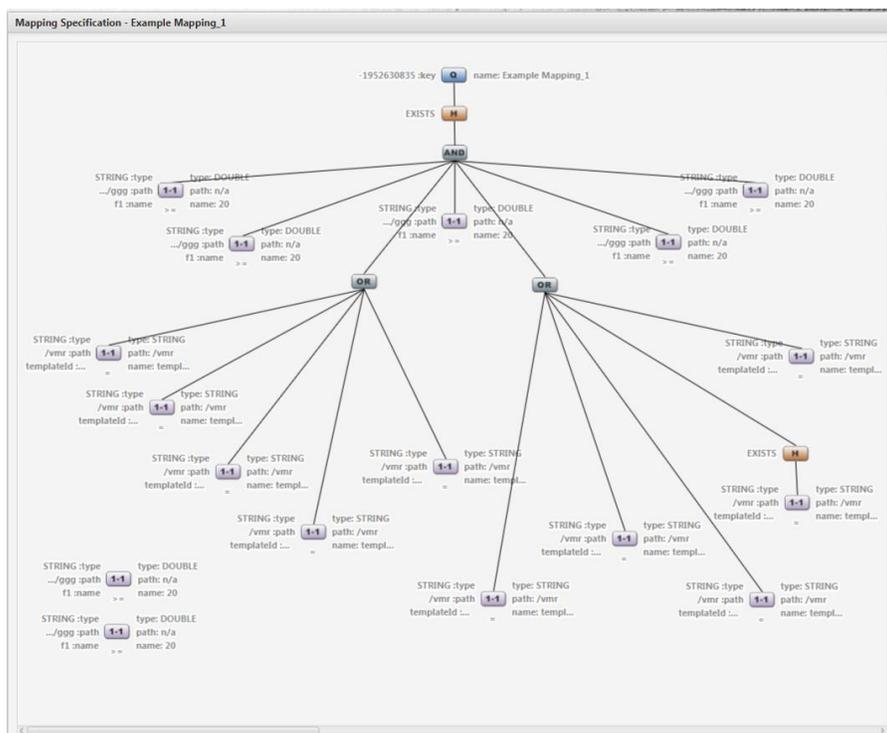
- 3) **Clipboard** – A board that can hold a duplication of a certain selected node by clicking the right mouse button on a desired node that we want to duplicate → copy to → clipboard. If node was copied to clipboard, it will stay there until another node will override it. The stored node duplication in the clipboard can be drag and dropped into the same mapping it was copied from (as a different node, but with same specification and sub-tree of nodes), or it can be dragged and dropped into a different mapping (by selecting a different mapping from the mappings sliding panel on the left) and it can be copied as many times needed (each copy created new nodes with same tree structure and specification).



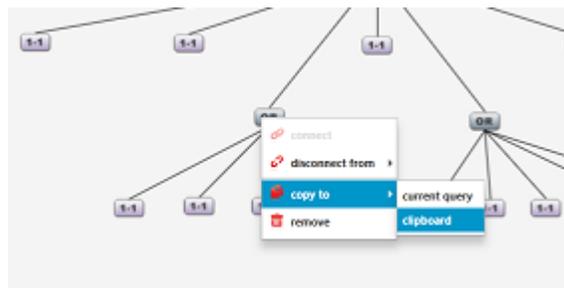
- 4) **Menu bar** – A menu bar with the following options:
- **File** – Basic system actions:
 - **Print** – prints current screenshot.
 - **Exit** – Exits the system.
 - **Edit** – Mapping management/definition actions, all actions under this menu are the same as the buttons' actions at the right and bottom buttons areas. Thus, please see the specification of the different buttons.
 - **View** – View actions:
 - **Detailed view** – Turns on/off the detailed nodes' view, a detailed nodes view is the visual representation of all nodes of the currently selected/opened mapping in the mapping specification area.
 - **Help** – Help actions:
 - **About** – systems about screen.
- 5) **Right buttons area** –
- **Print** – Prints current screenshot.
 - **Lock** – Indicates if the current mapping is locked/unlocked. If it's locked, KDOM services won't be able to use this mapping (e.g. for situations when the mapping is in progress of refinement and it is not in its final approved version). This button changes it's icon from a locked lock to an unlocked one and vice versa to indicate the current locking status of the mapping.

- 6) **Bottom buttons area** –

- **Undo** – Undoes the last user modification on the currently opened mapping specification, the modification can be: a change of any attribute from the node specification tabs, adding new node to a mapping (by dragging and dropping from the clipboard or the dragboard etc. If there are no modifications to the currently opened mapping, this option will be disabled.
 - **New Mapping** – Creates a new mapping instance. If the currently opened mapping has unsaved pending changes, this button will be disabled.
 - **Save Mapping**– Saves any pending unsaved changes to the currently opened mapping. If there are no unsaved pending changes, this button will be disabled.
 - **Delete Mapping**- Deletes currently opened mapping. If no mappings are selected\ opened, this button will be disabled.
- 7) **Mapping Specification** – It's a draggable surface, in which the definition of a mapping is visually constructed as a tree of nodes. The user has full control over the mapping instance:
- **Drag and drop** – The user can move nodes and drag and drop new nodes from the dragboard.
 - **Node selection** – A node can be selected by clicking the left mouse button on it; it will be marked with a yellow bright border. When a node is selected, its specification sliding container will open at the bottom.
 - **Detailed view** – The mapping can be shown in a detailed view (i.e. each node will be surrounded with a short info of its specification), can be turned on\off at menu bar → view → detailed view (see below):



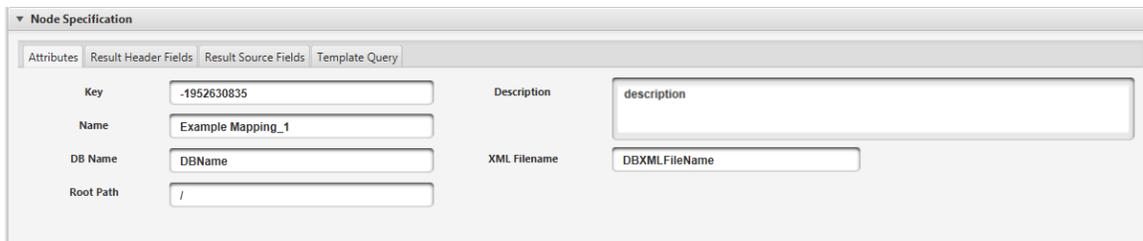
- **Node menu** – The user can open a node's menu by clicking the right mouse button on it:



The menu has the following options (see below):

- **Connect** - For connecting the current node (i.e. node which the "connect" option was selected on) as a "parent" upper node to a sub-node. After clicking this option, for connecting to a sub-node click left mouse button on the desired sub-node in the mapping specification. If the sub-node wasn't connected to other "parent" node, a connection between the two will be established (i.e. a line will be drawn between the two). This option will be enabled only for nodes that can connect to more sub-nodes than they have now (i.e. before the current connection attempt).

- **Disconnect from** – Opens a sub-menu with options to indicate which nodes to disconnect from. Has the following options:
 - **Parent** – will disconnect from parent node.
 - **Sub-nodes** – will disconnect from all sub-nodes.
 - **All** – will disconnect from parent and sub-nodes.
 - **Copy to** – Opens a sub-menu with options to indicate the destination of the copying. Has the following options:
 - **Current Mapping** - Will copy the current node (i.e. node which the "Copy to" option was selected on) to the current mapping's specification (i.e. the node and it's sub-tree of nodes will be copied).
 - **Clipboard** – Will copy the current node and it's sub-tree of nodes to the clipboard.
 - **Remove** – Deletes selected node from the current mapping.
- 8) Node Specification** – A sliding container with several tabs which describe a specific selected node from the mapping specification area (selected node will be marked in yellow). The user can drag and drop elements and attributes from the mappings sliding panel (left panel), into certain fields to fill the specification automatically (e.g. field's path and name in the destination database, by dragging and dropping the field element). The tabs are:
- **Attributes tab** – According to the selected (in the mapping specification) node's type, its attributes will be shown at this tab. This tab is used by all node types, and each node tab has different attributes (i.e. each node type has different purpose\definition):
 - **Query node** – Has the following attributes:

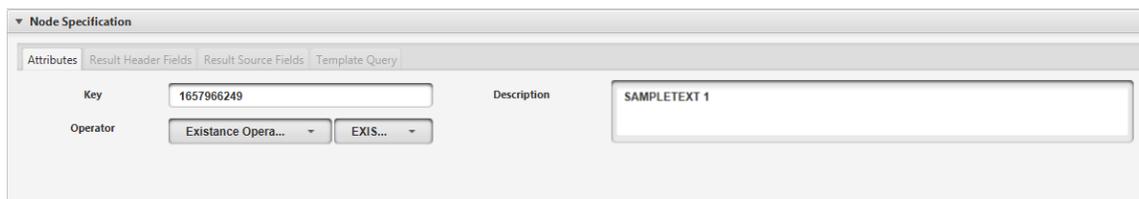


Node Specification			
Attributes Result Header Fields Result Source Fields Template Query			
Key	<input type="text" value="-1952630835"/>	Description	<input type="text" value="description"/>
Name	<input type="text" value="Example Mapping_1"/>	XML Filename	<input type="text" value="DBXMLFileName"/>
DB Name	<input type="text" value="DBName"/>		
Root Path	<input type="text" value="/"/>		

- **Key** – The primary identification unique ID of a mapping, KDOM services will be able to recognize the mapping according to this ID (set by the user).
- **Name** – The name of the mapping.
- **Description** – Mapping's description.
- **DB Name** – The destination database name, which the mapping is mapped to.
- **XML File Name** – The XML file name from the database that the mapping is mapped to.

- **Root Path** – The scope\level (deep) of the xml database on which the mapping should run.

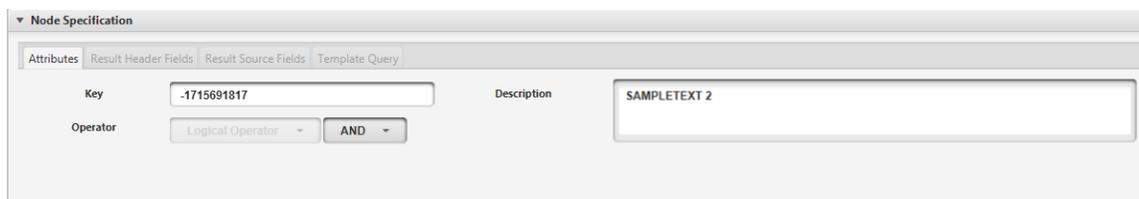
- o **Header node** – Has the following attributes:



The screenshot shows a 'Node Specification' form with tabs for 'Attributes', 'Result Header Fields', 'Result Source Fields', and 'Template Query'. The 'Attributes' tab is active. It contains a 'Key' field with the value '1657966249', a 'Description' field with 'SAMPLETEXT 1', and an 'Operator' section with two dropdown menus: 'Existence Opera...' and 'EXIS...'.

- **Key** – A unique node identifier (set automatically by the system).
- **Description** – A description of the node.
- **Operator** - Two combo boxes that allows you to choose the operator type and the operator itself (e.g. negation operator NOT).

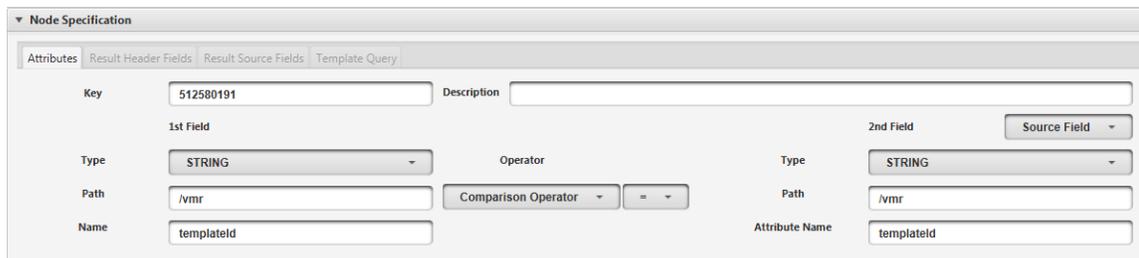
- o **AND/OR node** – Has the following attributes:



The screenshot shows a 'Node Specification' form with tabs for 'Attributes', 'Result Header Fields', 'Result Source Fields', and 'Template Query'. The 'Attributes' tab is active. It contains a 'Key' field with the value '-1715691817', a 'Description' field with 'SAMPLETEXT 2', and an 'Operator' section with two dropdown menus: 'Logical Operator' and 'AND'.

- **Key** - A unique node identifier (set automatically by the system).
- **Description** – A description of the node.
- **Operator** – Can chose the operator from the logical operators type (e.g. AND, OR means that AND node can be changed to OR node in vice versa).

- o **1-1 node** – A node that defines logical criterion between two fields (i.g. fieldA < fieldB), a field to a value (e.g. fieldA < 8) or a field to a predefined value (e.g. fieldA < \$value - a placeholder to be replaced in run time when executed by KDOM services on the server) criterion. The actual comparison is done with a comparator that can be selected. The fields can be drag and dropped from the database template, the path and field name attributes will be filled automatically. It has the following attributes:



The screenshot shows a 'Node Specification' form with tabs for 'Attributes', 'Result Header Fields', 'Result Source Fields', and 'Template Query'. The 'Attributes' tab is active. It contains a 'Key' field with the value '512580191', a 'Description' field, and two columns for field specifications. The left column has '1st Field' with 'Type' set to 'STRING', 'Path' set to '/vmr', and 'Name' set to 'templateId'. The right column has '2nd Field' with 'Source Field' dropdown, 'Type' set to 'STRING', 'Path' set to '/vmr', and 'Attribute Name' set to 'templateId'. An 'Operator' section between the columns has 'Comparison Operator' and '=' dropdowns.

- **Key** - A unique node identifier (set automatically by the system).
 - **Description** – A description of the node.
 - **Type (left field)** – The left field's type (e.g. String, Integer etc.)
 - **Path (left field)** – The left field's path.
 - **Name (left field)** – The left field's name.
 - **Operator** – Two combo boxes that allow to choose the operator type and the operator itself (e.g. math operator '=').
 - **2nd Field** – The right field can be one of the following types: source field (i.e. regular), value (regular value) and pre-defined, the first field is a source field and cannot be changed.
 - **Type (right field)** – The right field's type (e.g. String, Integer etc.)
 - **Path (right field)** – The right field's path.
 - Name (right field) – The left field's name.
- **Result header fields tab** – At this tab there is a table of fields definitions, can be modified by double clicking cells in the table. This fields represent the header fields that will be returned in the result of the execution of the mapping against the PHR database via KDOM, they are defined in the header tag of the template XQuery:

<KDOM-reply predefinedAttribute=AttributeValue>{ ...

▼ Node Specification

Attributes | Result Header Fields | Result Source Fields | Template Query

Type	Modification Type	Name/Value
STRING	Pre Defined	\$patientID

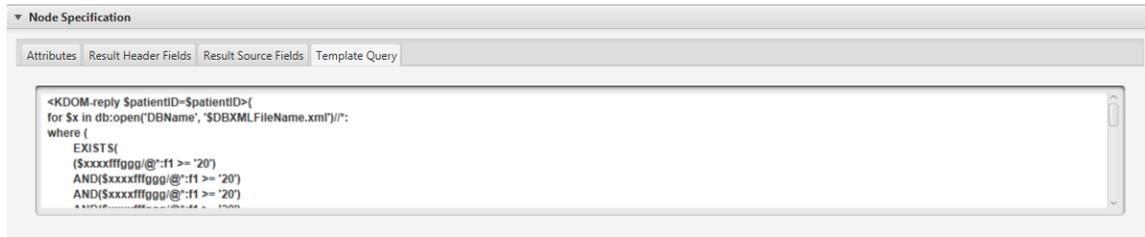
- **Result source fields tab** – At this tab there is a table of fields definitions, can be modified by double clicking cells in the table. This field represents the return clause of the query, and those fields will be returned for each found concept by the defined mapping criteria (i.e. nodes tree structure).

▼ Node Specification

Attributes | Result Header Fields | Result Source Fields | Template Query

Type	Path	Name/Value
STRING	/sd/sd/sd/s	sss

- **Template query** – A representation in real time (i.e. each saved change triggers regeneration of the template query) of the mapping instance in query language syntax. The KDOM in MobiGuide project is a mapping tool to a XML based database, thus the querying language the template queries are defined in is XQuery.



In future KDOM version releases\updates, there will be support for other querying languages and databases. The intension is to be able to support big data solutions:

- Hadoop based solutions like Impala, Hive and HBase etc.
- Cloud based solutions like Amazon's S3Cloud with Red Shift database etc.

- 9) **Search** – A search text box that allows the search of an item from the list of item types in the search combo box. Currently the user can search for a certain mapping by its ID or name. Other search capabilities can be added in future software version releases\updates.