User Manual

v2.5.2

1 Introduction

LogFaces is a standalone solution for collecting, storing, dispatching and filtering log data. LogFaces is a "box" which resides next to your system and lifts the logging burden off you. Built one level above the conventional logging API's such as Apache Logging Services it easily integrates with existing systems, including those built on J2EE, native C++, or Microsoft© .NET Framework. If your system is already using those API's, you can use logFaces for storing log, real-time log monitoring, evaluating operational discrepancies, receiving scheduled email reports with log files and more...

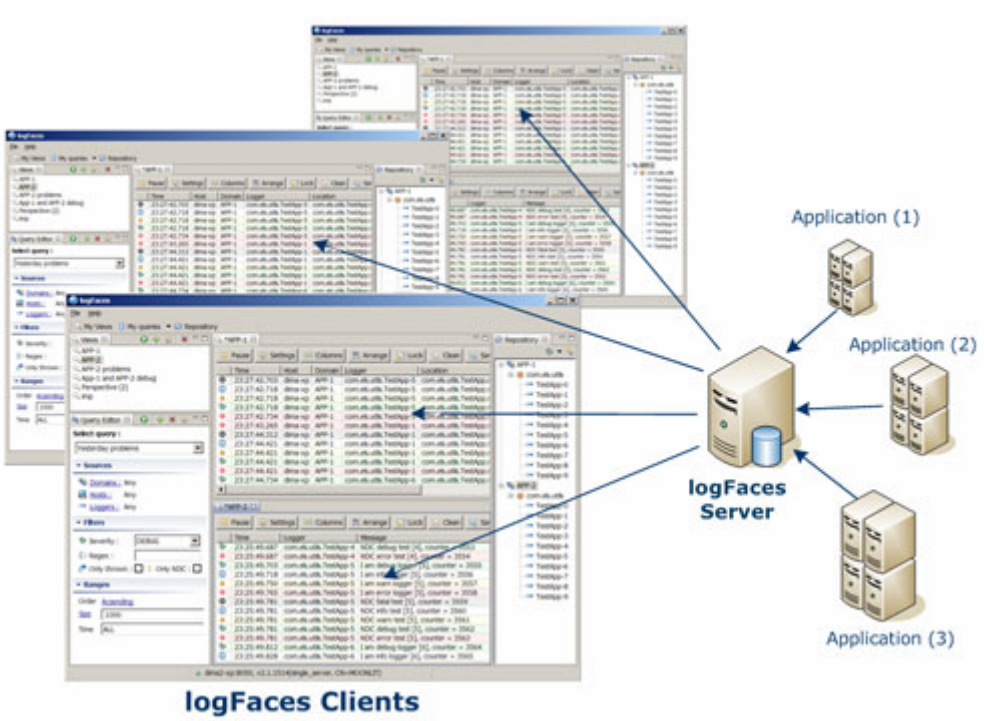


Figure 1-1 LogFaces Architecture

There are three players - your system, logFaces Server and logFaces Client. It works like this: your system/s send log data to the logFaces Server which routes this data to interested clients or **does something with this data on its way**. For example; saving some of it to database or discarding it, splitting and filtering it for various client listeners. The only requirement for any system to be operable with logFaces is to produce log in a format understood by the logFaces Server which collects this log. The current version of logFaces supports Apache log4xxx API's as schema of log statements. But technically logFaces will not be limited only to those formats. Today log4xxx API's are the de-facto standards in software development and most systems already support them. In such cases the integration with logFaces is straightforward and is only a matter of simple configuration.

LogFaces can be used in a compact mode (or **Server Mode** as we call it) where applications send log data directly to the Client. This mode is mostly dedicated for those who want to use logFaces as a real time viewer for single user. It fits better smaller projects due to its simplicity, however it doesn't have persistence capabilities, reporting and data mining features – it's a plain real time viewer of log data.

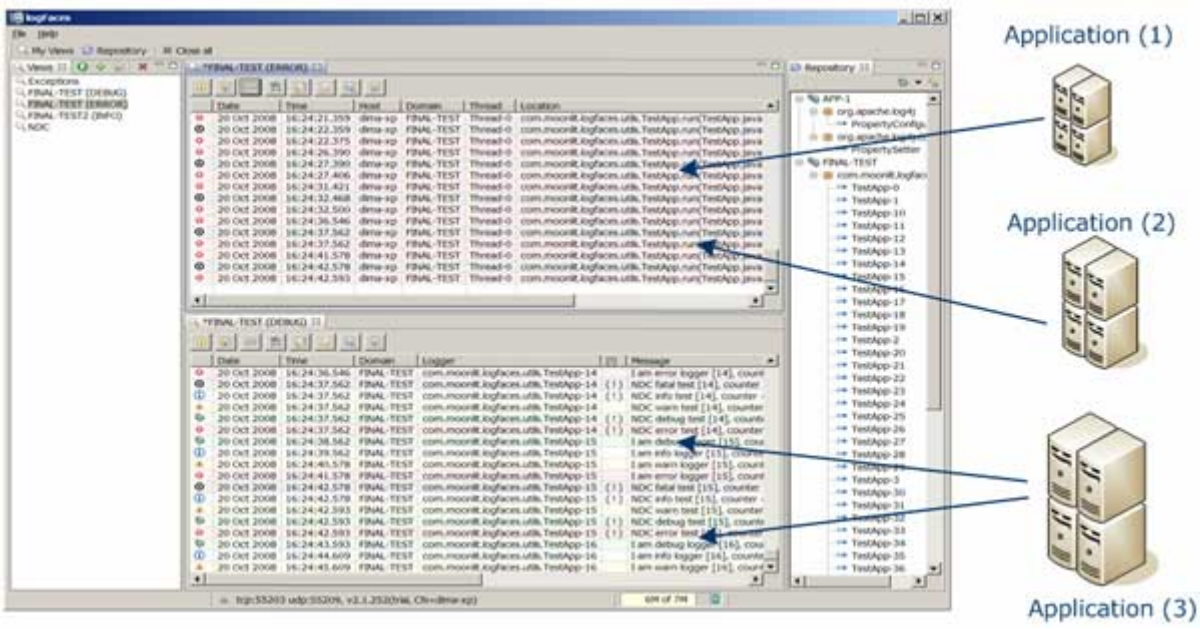


Figure 1-2 logFaces Client in server mode

2 Getting started with logFaces Server

This section will guide you through the quick process of installation and configuration. Java™ Runtime Environment (JRE) 1.5 or later must be installed before proceeding with installation either on Windows or on Linux. LogFaces Server will automatically start after the installation with default settings and trial license for **30 days evaluation**.

2.1 Installing logFaces Server

1. Server distributions come as files named **lfs-os-architecture-bits.zip.** Make sure you select the correct distribution.
2. Unzip downloaded distribution to any location on your local disk.
3. On Windows, if you want to run as a service - it must be registered. Run /bin/installservices.bat with **Administrator** privileges - this will register win32 service called "LFS".
4. On Linux/Unix you will have to give executive permissions to files located in **/bin**.

2.2 Running logFaces Server on Windows

There are two ways to run the server :

1. as a console application - use **/bin/run.bat**
2. as a win32 service - use your service control panel, or type "**net start lfs"** in the command prompt. Make sure you register the service before doing it!

2.3 Running logFaces Server on Linux/Unix

Make sure you give executive permissions to the files located in **/bin** directory.

- In order to start the server in terminal execute command **./bin/lfs console**
- In order to start the server as daemon execute command **./bin/lfs start**
- In order to stop the server process execute command **./bin/lfs stop**
- In order to check the server process status execute command **./bin/lfs status**

2.4 Integrating with applications

To work with logFaces, your application needs to be configured by adding several elements to its logging configuration file. Provided that your system is based on log4xxx API's, you should be having log4xxx configuration file/s, which usually come as property files or XML configuration files. We add an appender to your configuration file which knows how to communicate with logFaces Server. Sections below will show how to setup such appenders in various situations. As of this writing we provide our own appenders for log4j and logback frameworks while .Net and C++ can be freely obtained from Apache and don't require any change to work with logFaces.

2.4.1 Log4j configuration

There are two appenders available to work with log4j applications - simple socket appender and asynchronous socket appender. Depending on your needs you can use either of them but we strongly recommend using the former one as it's more robust and gives better performance. Both appenders can be found in **/lib/lfsappenders.jar** on server installation and also available separately for download from our site.

2.4.1.1 Simple Socket Appender - `com.moonlit.logfaces.appenders.LFXMLSocketAppender`

LFXMLSocketAppender is very basic socket appender which is not much different from the one distributed with log4j itself. The following example shows how to set it up in your configuration :

```
log4j.appender.LFS = com.moonlit.logfaces.appenders.LFXMLSocketAppender
log4j.appender.LFS.application = APP-1
log4j.appender.LFS.remoteHost = localhost
log4j.appender.LFS.port = 55200
log4j.appender.LFS.locationInfo = true
log4j.appender.LFS.reconnectionDelay = 5000

log4j.rootLogger = TRACE, LFS, CONSOLE
```

Property name	Description	Default	Mandatory
application	Identifies application under this name.	none	no
remoteHost	Host name or IP address of logFaces server		yes
port	Port where logFaces server listens.	55200	no
locationInfo	Specifies whether to include location data - class, method and line numbers.	FALSE	no
reconnectionDelay	Rate of reconnection retries in milliseconds.	5000	no

2.4.1.2 Asynchronous Socket Appender - `com.moonlit.logfaces.appenders.AsyncSocketAppender`

AsyncSocketAppender is asynchronous queued appender with built-in fail over mechanism. When your application will do log statements, the events will not be sent to server at the expense of calling thread. They will be queued and sent to logFaces server by the background thread. You can specify several parameters to adjust the appender to optimal performance. In addition to queuing, this appender also knows how to fail over to another host should the current one fail. You can specify how often to retry the connection, how many times to retry and to what host to switch to when all retries are exhausted. This is an example of log4j XML configuration :

```
<configuration configDebug="true">
    <appender name="LOGFACES" class="com.moonlit.logfaces.appenders.AsyncSocketAppender">
            <param name="locationInfo" value="true" />
            <param name="application" value="MY-APPLICATION" />
            <param name="remoteHost" value="host1,host2" />
            <param name="port" value="55200" />
            <param name="reconnectionDelay" value="5000" />
            <param name="nofRetries" value="2" />
            <param name="offerTimeout" value="5000" />
            <param name="queueSize" value="500" />
    </appender>
    <root>
       <priority value ="debug" />
       <appender-ref ref="LOGFACES" />
    </root>
</configuration>
```

This is an example of log4j property file configuration :

```
log4j.appender.LOGFACES = com.moonlit.logfaces.appenders.AsyncSocketAppender
log4j.appender.LOGFACES.locationInfo = true
log4j.appender.LOGFACES.Application = MY-APPLICATION
log4j.appender.LOGFACES.RemoteHost = host1, host2
log4j.appender.LOGFACES.Port = 55200
log4j.appender.LOGFACES.reconnectionDelay = 5000
log4j.appender.LOGFACES.nofRetries = 3
log4j.appender.LOGFACES.offerTimeout = 0
log4j.appender.LOGFACES.queueSize = 500

log4j.rootLogger = TRACE, LOGFACES
```

Table below summarizes all properties available for AsyncSocketAppender. Note that some of the properties are optional and can be omitted, but pay close attention to how you specify fail over parameters and queue options :

Property name	Description	Default	Mandatory
locationInfo	Specifies whether to include location data, e.g. class name, method name and line numbers.	FALSE	no
application	Identifies application under this name. All logs coming through this appender will be stamped with this name, which can later be used on client.		no
remoteHost	Comma separated list of logFaces servers. If more than one host specified, the appender will automatically fail over to the next host when current host becomes unavailable. Switching hosts is done in the loop. If only one host specified, the retries will be done indefinitely with this host.		yes
port	Port where logFaces server will accept the connection from this appender.	55200	no
reconnectionDelay	Rate of reconnection retries in milliseconds.	5000	no
nofRetries	How many times to retry before dropping current host and switching to the next one. If only one host specified in remoteHost attribute, the retries will go indefinitely to the same host.	3	no
queueSize	Size of the event queue. The larger the size, the less likely the data will get lost when connection is lost, because events will be re-transmitted to the server when connection recovers. However, queue size affects JVM heap memory, so be considerate.	500	no
offerTimeout	How long to wait (in msec) while offering event to the appender queue. When server is slower than application and queue gets full, the caller has an option to wait before giving up. Queue can typically get full when server is down or when server can't consume log data in the rate of this appender. **WARNING: Use with care as it will slow down the calling thread when queue fills up.**	0	no

2.4.2 Logback configuration

If you are using logback framework in your applications, you can easily connect with logFaces too. Below is an example of our appender which you should add to your logback configuration.

```
<configuration>

    <appender name="LFS" class="com.moonlit.logfaces.appenders.logback.LogfacesAppender">
        <application>LOGBACK-TESTER</application>
        <remoteHost>localhost</remoteHost>
        <port>55200</port>
        <locationInfo>true</locationInfo>
        <reconnectionDelay>5000</reconnectionDelay>
    </appender>

    <root level="trace">
        <appender-ref ref="LFS" />
    </root>

</configuration>
```

Meaning of the attributes are identical to our log4j appender described above. Note that class name is of course different. Make sure to place **lfsappenders.jar** into the class path of your application, it can be found either in /lib directory of server installation or from our download page. Logback dependency jars must be in the class path as well, make sure you grab them from the authors web site.

2.4.3 Log4cxx configuration

If your system is based on C++ and using Apache Log4cxx API for logging, you should configure it by adding XMLSocketAppender included in log4cxx API itself. Here is a snippet of configuration example:

```
log4j.appender.stdout=org.apache.log4j.ConsoleAppender
log4j.appender.stdout.layout=org.apache.log4j.PatternLayout
log4j.appender.stdout.layout.ConversionPattern= %-5p %d{HH:mm:ss} %-20c{1} %X{stam} | %m%n

log4j.appender.LFS=org.apache.log4j.net.XMLSocketAppender
log4j.appender.LFS.RemoteHost=10.200.1.110
log4j.appender.LFS.Port=55200

log4j.rootLogger=debug, stdout, LFS, FILE
```

The meaning of the attributes is identical to the previous Java™ example. However, note that XMLSocketAppender doesn't (yet?) provide "Application" and "LocationInfo" attributes. This is not a problem for logFaces – those loggers which don't correspond to any logging domain will be automatically grouped in logFaces under name "**Default Domain**". Unfortunately, until those attributes are supported by the underlying API's, we will have to add some code when initializing the

logger in the application. The code snippet is shown below, what we do is simply getting into a root logger, digging out the LFS appender from there and manually set the missing attributes of the layout like this:

```
// this is a workaround for XMLSocketAppender to allow routing of properties
// over the network, we manually setup the layout
LoggerPtr root   = Logger::getRootLogger();
AppenderPtr app = root->getAppender(LOG4CXX_STR("LFS"));
if(app != NULL){
   LayoutPtr layout = app->getLayout();
   if(layout != NULL){
      layout->setOption(LOG4CXX_STR("locationinfo"), LOG4CXX_STR("true"));
      layout->setOption(LOG4CXX_STR("properties"), LOG4CXX_STR("true"));
      MDC::put("application", "WSC");
   }
}
```

In any case, those missing attributes are not a show stoppers, your application can still work with logFaces out of the box with those limitations.

IMPORTANT:

The MDC (message diagnostic context) works only in the context of the current thread. In case you have several threads in your application you should add **MDC::put("application", "xxx")** call in the beginning of every thread. Otherwise, the log statements coming from those threads will be orphaned and server will automatically put them under "Default Domain" which might be a bit confusing.

Future versions of logFaces will include proper appender to avoid those workarounds.

2.4.4 Log4net configuration

If your system is .Net based and using Apache log4net API for logging, you should configure it by adding UdpAppender. Here is a snippet of configuration example:

```
<log4net>
        <appender name="LFS" type="log4net.Appender.UdpAppender">
                <param name="RemoteAddress" value="127.0.0.1" />
                <param name="RemotePort" value="55201" />
                <param name="Encoding" value="UTF-8" />
                <layout type="log4net.Layout.XmlLayoutSchemaLog4j, log4net"></layout>
        </appender>
        <root>
                <level value="DEBUG" />
                <appender-ref ref="LFS" />
        </root>
</log4net>
```

As mentioned earlier, logFaces can listen for TCP and/or UDP. In this example, we use UDP appender and you should make sure that RemotePort attribute in this example corresponds to the one configure in logFaces, which of course can be modified using logFaces Administration Console.

2.5 Administration

Most of logFaces Server configuration is done remotely using conventional browser. To access Administration Console, open your browser and navigate to this URL:

http://your-logfaces-server-host:8050

The shortcut is also created for you during the installation in Windows Start menu. Of course, you can access the console from any other host as long as you network configuration allows the access.

Access to logFaces Administration Console is secured by user name and password, which are both defaulted to "**admin**" during installation:

If you want to modify user name and/or password for the future use, see **Advanced configuration "how to"** for more details.

Administration Console is grouped into several tabs where you can easily modify needed functionality – connectivity, database, context, e-mail, notifications, licensing and status. Let's go through them in details.

2.5.1 Connectivity

Here you specify external and internal ports where Server will be listening :

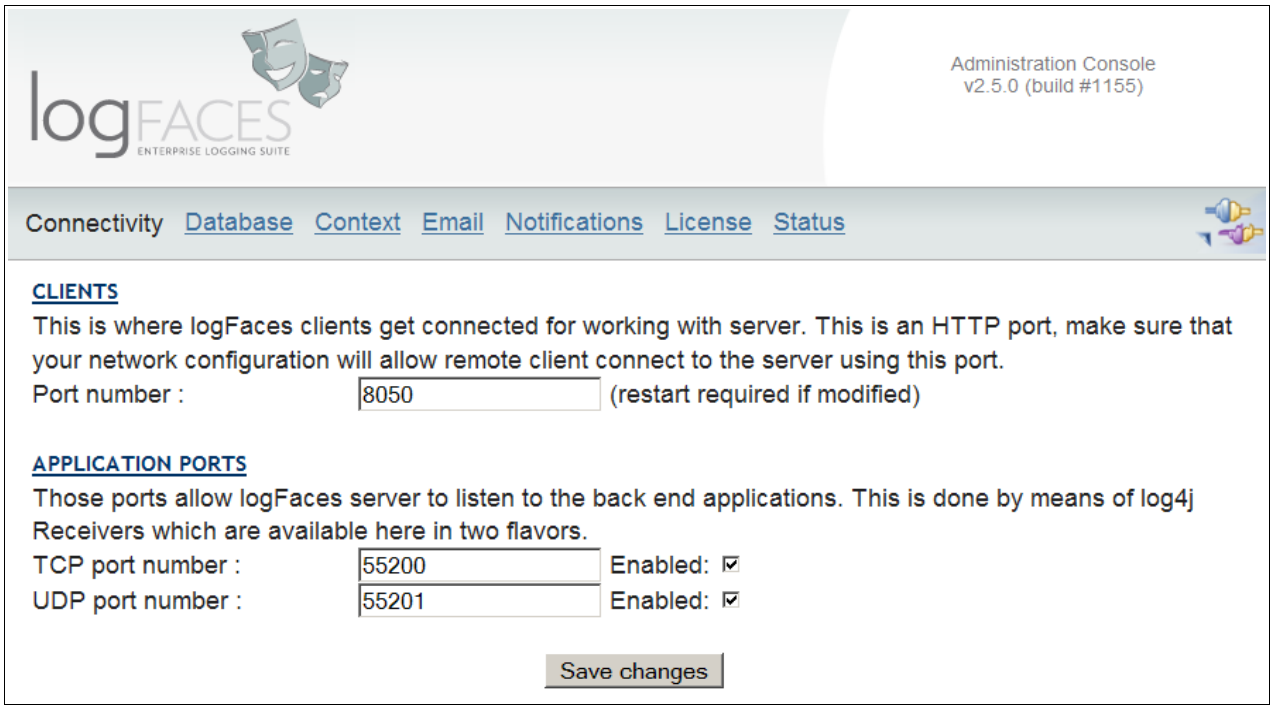


Figure 2-3 Connectivity settings

External port defaulted to 8050 is for Clients and Administration Console connections. This is HTTP connection port which you can modify to fit your environment, but Server will have to be restarted for the changes to take affect.

Internal ports are for the applications to log into logFaces. There are two ports available – TCP and UDP which you can modify or enable/disable. **Changes are effective immediately**. Make sure your applications appenders are adjusted to meet those changes. Socket appender port property named **log4j.appender.LOGFACES.Port** in your configuration should match the ones specified here.

2.5.2 Database

In Database section we specify how logFaces Server database should operate. There are two major things we need to specify – **how much** log to keep and **what** log events should be saved in database (if any).

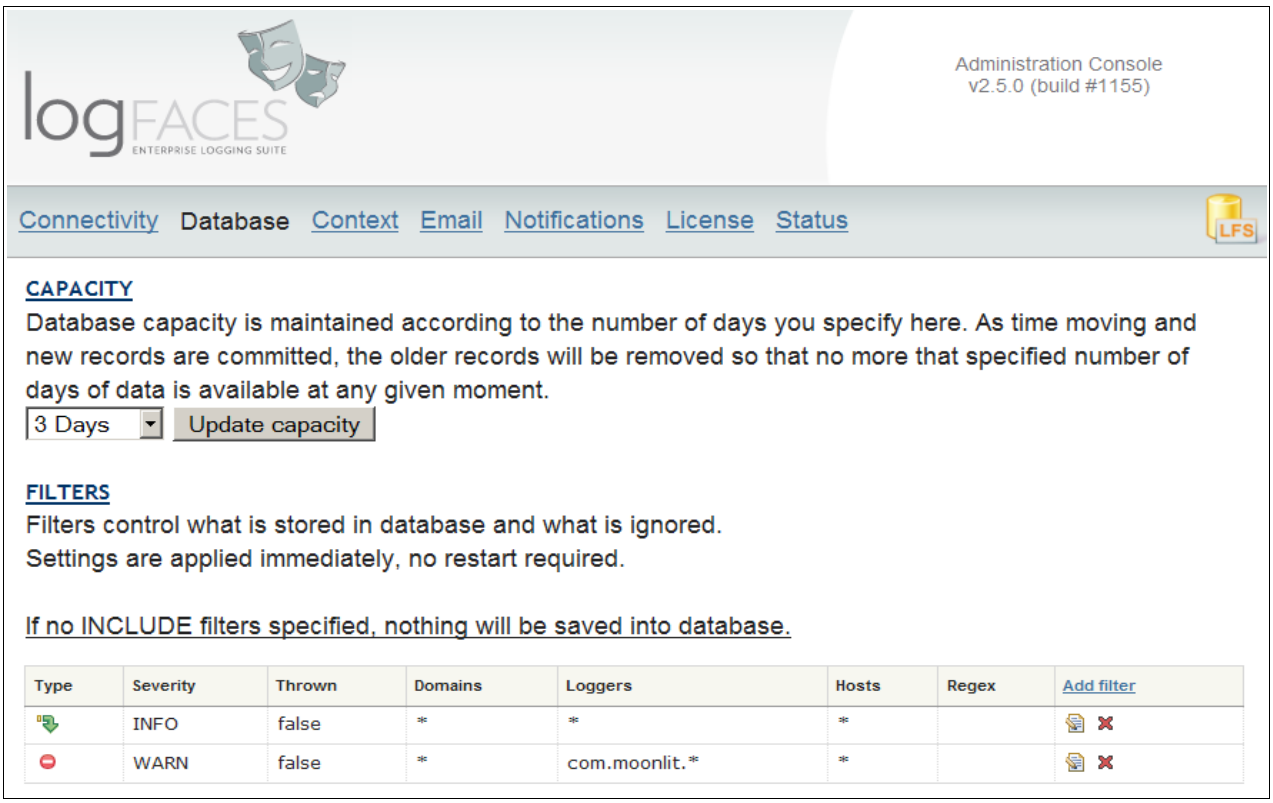


Figure 2-4 Database settings

Database **capacity** is specified in days of log. If you specify "1 week" for example, then at least one week of data will always be available. As time goes, older records are automatically removed while new ones are appended. You should carefully specify this value according your needs; it affects overall performance as well as disk space usage.

Filters let you specify which logging events to be stored in database. Normally we don't need debug information to be floating around and bloating our server, but sometimes we do need it too. There are two types of filters you can specify – **INCLUDE** and **EXCLUDE**. Filtering is done by **severity** of logging events going through logFaces Server, the **domain** (application) of the events and the **logger names**. You can use wild cards to specify names. The example above means that we want to include ALL events whose severity are higher (and including) INFO, but we don't want events which come from com.moonlit.* packages, except those which are higher (and including) ERROR. So, even though we specified the exclude filter, we still want problems from those packages to be saved. You can create fairly complicated filters here; the order of the filters is not important.

If you would like to disable database persistence, simply remove all INCLUDE filters or set capacity to NONE. There is no need to restart the server, those changes are applied automatically.

2.5.3 Context

One of the advanced features in many logging systems is diagnostic context attached by the application to logging events. In log4j and its other flavors there is MDC – Mapped Diagnostic Context. You can read more about it here. To provide convenient integration with MDC's, logFaces lets you map your application context properties in such way so that they could later be used in queries and other displays. You can specify up to 5 different context variables which will be automatically extracted from logging events and stored in database.

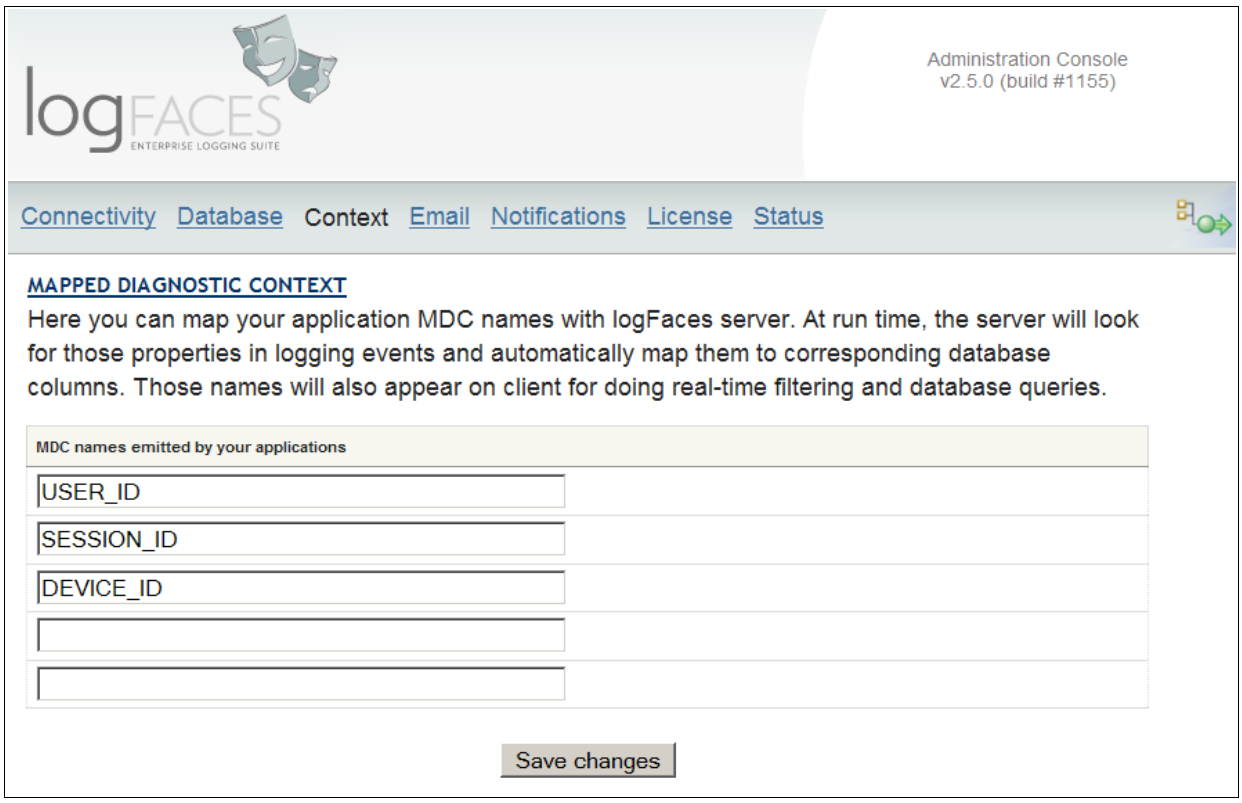


Figure 2-5 Context settings

You can modify those names any times during run time, but it's best to setup the mapping as earlier as possible. For example, if you initially worked with SESSION_ID as first MDC parameter and then decided to rename it to SESSION_NAME, you won't be able to extract previously saved values as SESSION_ID because it will be named differently.

2.5.4 Email

Email settings will allow logFaces server to send e-mails when required by reports and triggers. Here you define outgoing SMTP properties and also can verify that these settings are correct by sending test email to some recipient.

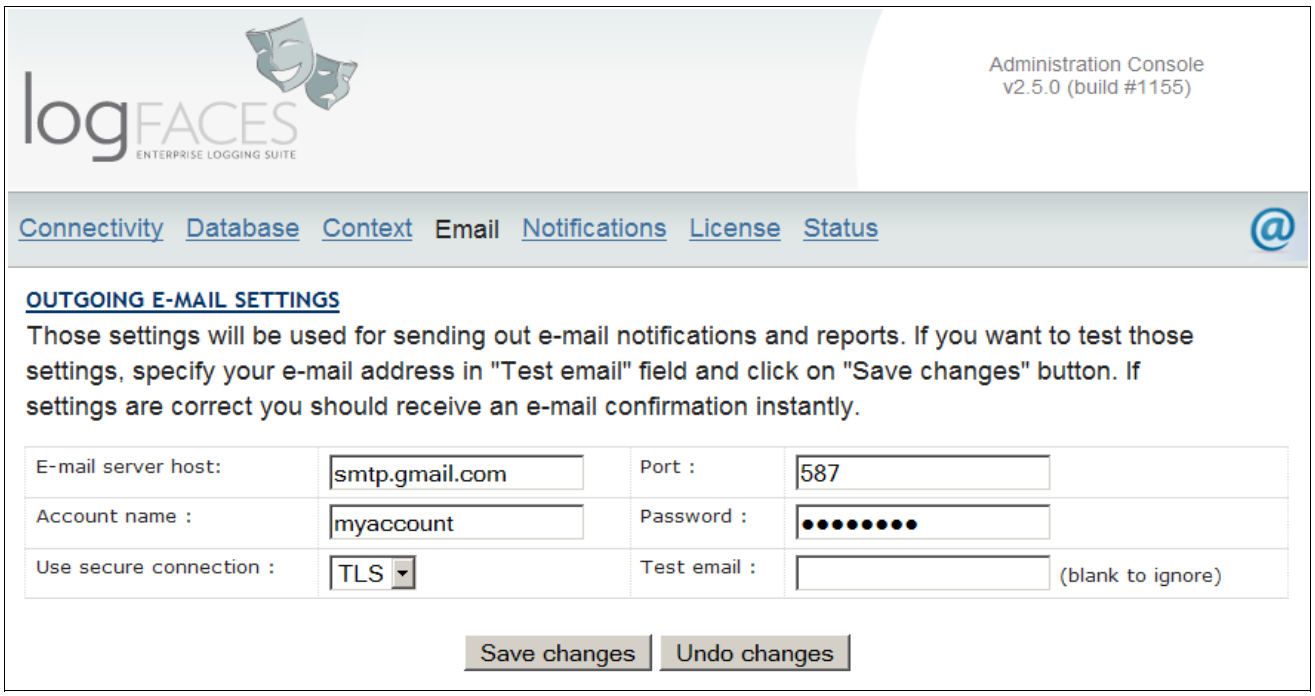


Figure 2-6 Email settings

In "Test email" field you can specify an email address where testing message will be sent to when you save changes. Should anything go wrong, you will be shown an error describing the cause. If everything was correct, you will receive acknowledging email.

2.5.5 Notifications

There are two types of notifications available - reports and triggers. You can manage both of them under this tab :

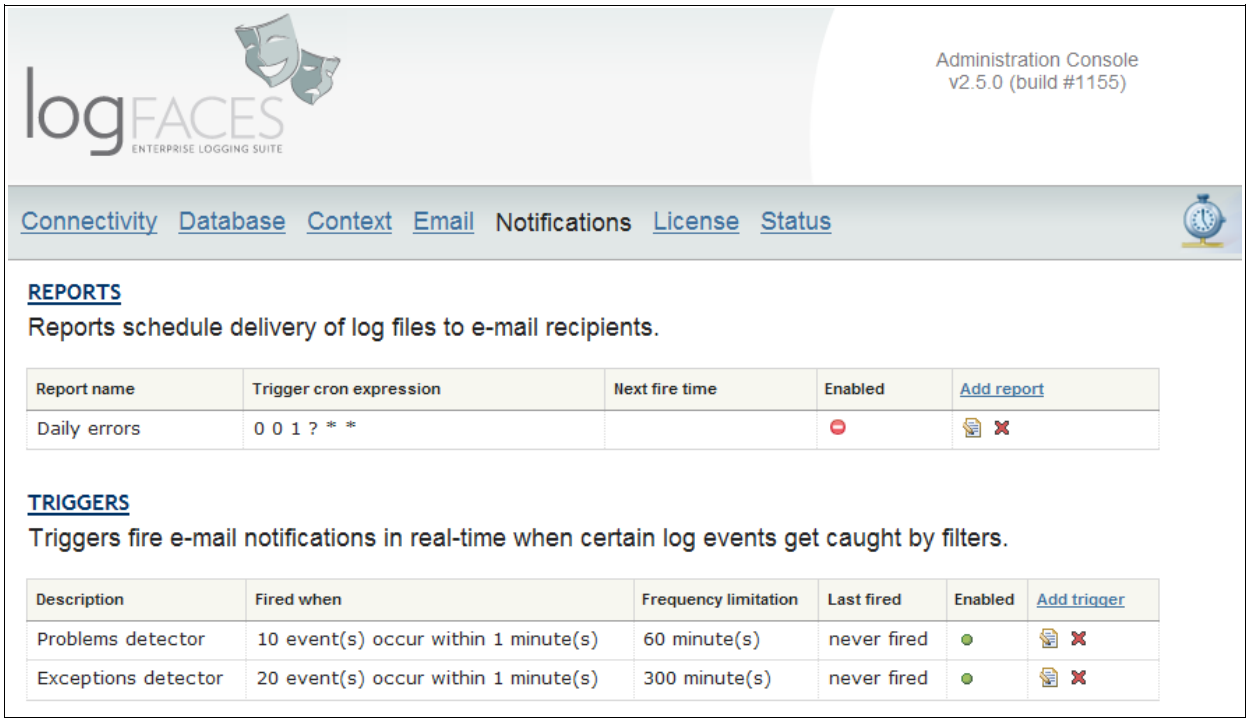


Figure 2-7 Notifications settings

2.5.5.1 Reports

Reports are custom log files that server generates according to schedule and rules you specify. Reports are sent by email to the recipients you specify. When defining report we need to specify what log information we need, the scheduling rule to dispatch report creation and who to dispatch the email to. logFaces Server can handle unlimited amount of reports based on complex scheduling rules and content. As you can see, reports are named, have cron expression with next fire time stamp and an indication if report is enabled. Having report enabled or disabled is handy when you want too keep report definitions but don't want it to fire. When you remove the report from the list, it is permanently deleted and can't be recovered. However, it's not a big deal to create one anew – just click on the "**Add new**" link at the right top header of the list and it will create new dummy records which you can populate and enable by clicking on the report name.

Below is an example of report options screen :

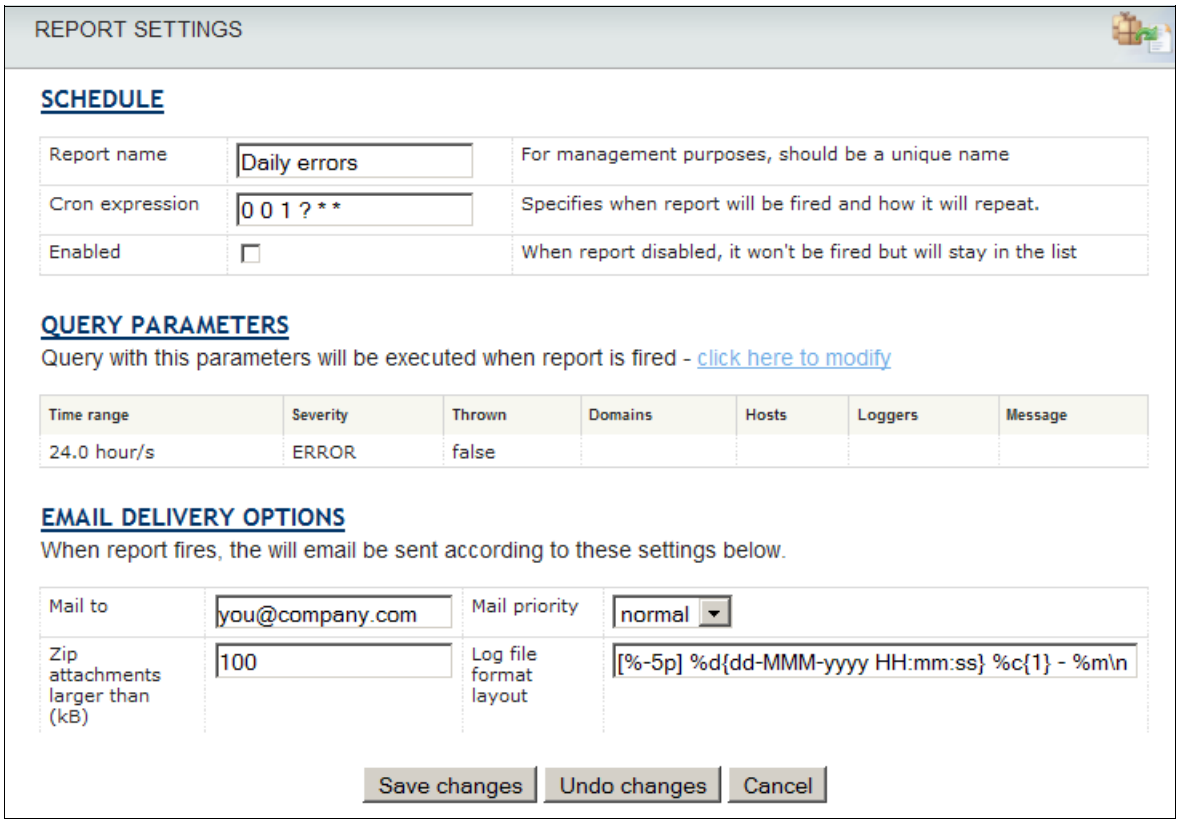


Figure 8: Report settings

- **Report name** – reports are uniquely identified by name, in fact each report is a scheduling job which is referred by the scheduling system by this name. Name of the report will appear in the subject of the email.

- **Cron expression** – is an expression which specifies when and how to fire the report. Cron expressions are very flexible and used to make fairly complex scheduling schema. In the picture example the expression means "fire every day at 1AM", but you can easily specify something like "fire on 10AM on the third Friday of the month". You can get more information about cron expression online, for example here.

- **Enabled** - when unchecked, the report will stay in the system but will never fire. You can enable it any time and it will fire at the next schedule.

- **Query parameters** - when report is triggered, the server will retrieve log statements from database using the following criteria - time range, severity level, domains, host, loggers, text matching and exception.

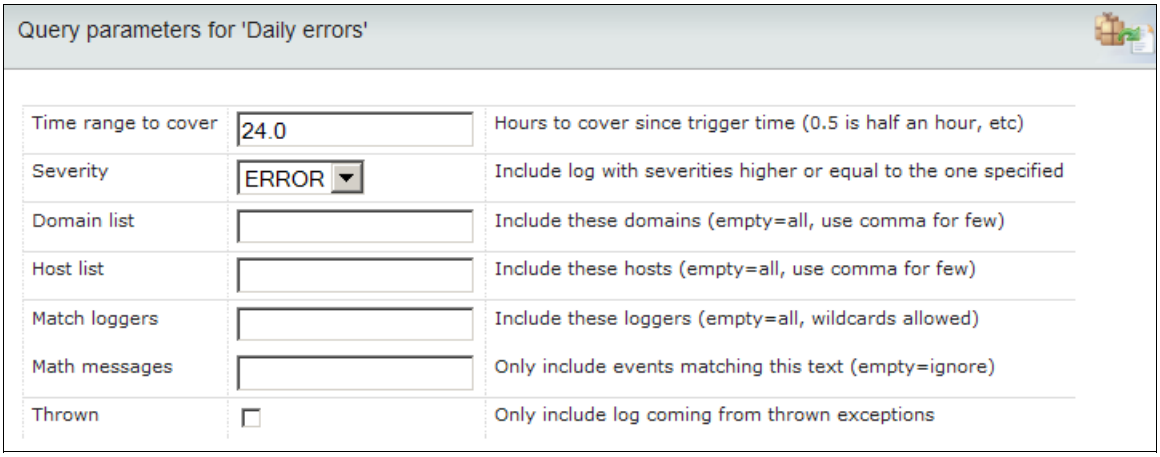


Figure 9: Report query parameters

- **Mail to** - list of recipients to receive the e-mail (use comma to include few recipients)

- **Mail priority** - e-mails can be flagged with standard e-mail priorities (highest, high, normal, low, lowest).

- **Zip attachments** - specify a maximum size of log file in KB; if attachment file will be larger than specified, it will be automatically zipped.

- **Layout** – specifies how to layout the text in the log files. LogFaces is using log4j formatting rules; you can find more details here

2.5.5.2 Triggers

Triggers are similar to reports except that are not scheduled but rather fired immediately when certain conditions met. Conditions are. of course, based on the log data going through the server. By specifying filters you will be able to detect very particular log statements from very particular sources. In addition to this, you can also specify how many of such events to capture and within what time span they should be.

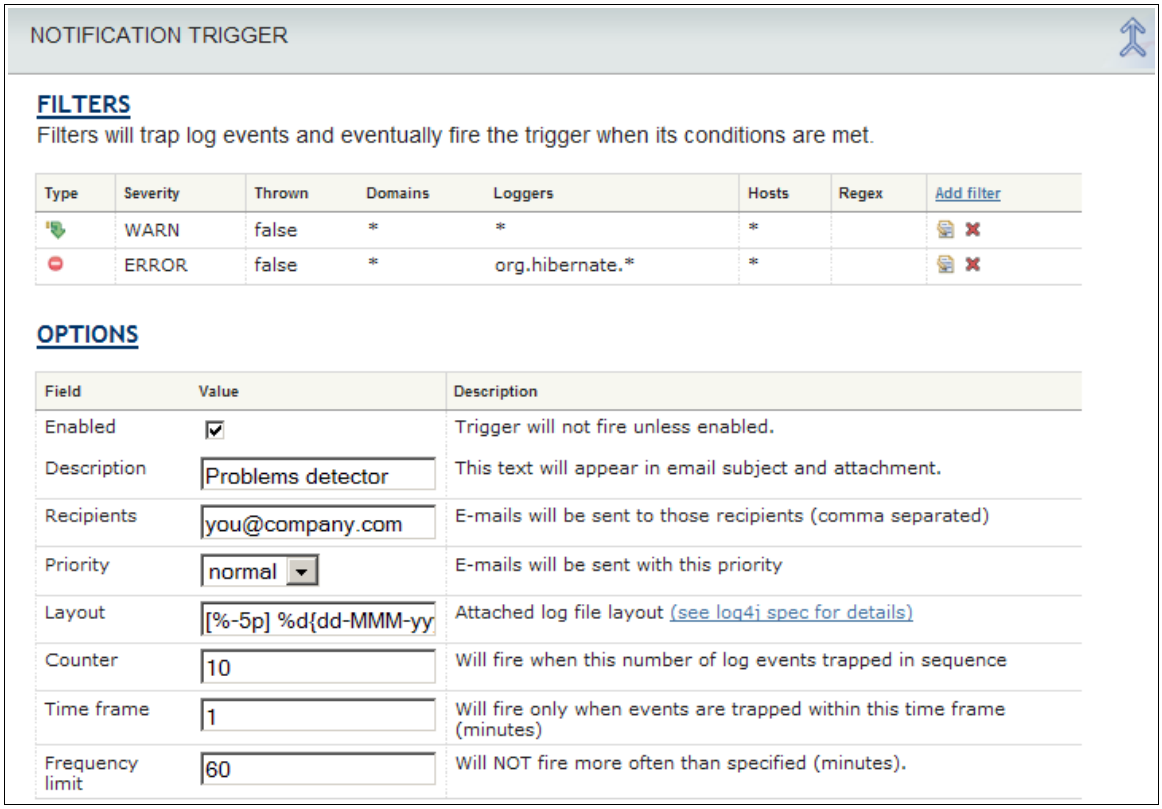


Figure 2-10 Trigger details

By combining **INCLUDE** and **EXCLUDE** filters you setup traps for actual log statements. This is how typical filters looks like :

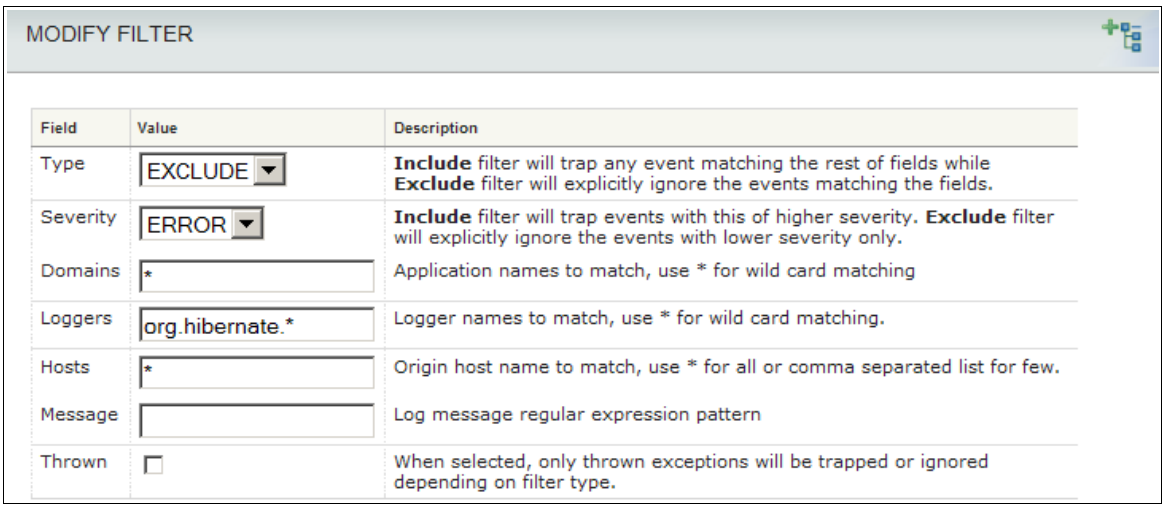


Figure 2-11 Exclude filter

Filter above will explicitly ignore anything coming from org.hibernate packages except those with severity higher than ERROR. But to make filter work we must specify what to INCLUDE, for example :

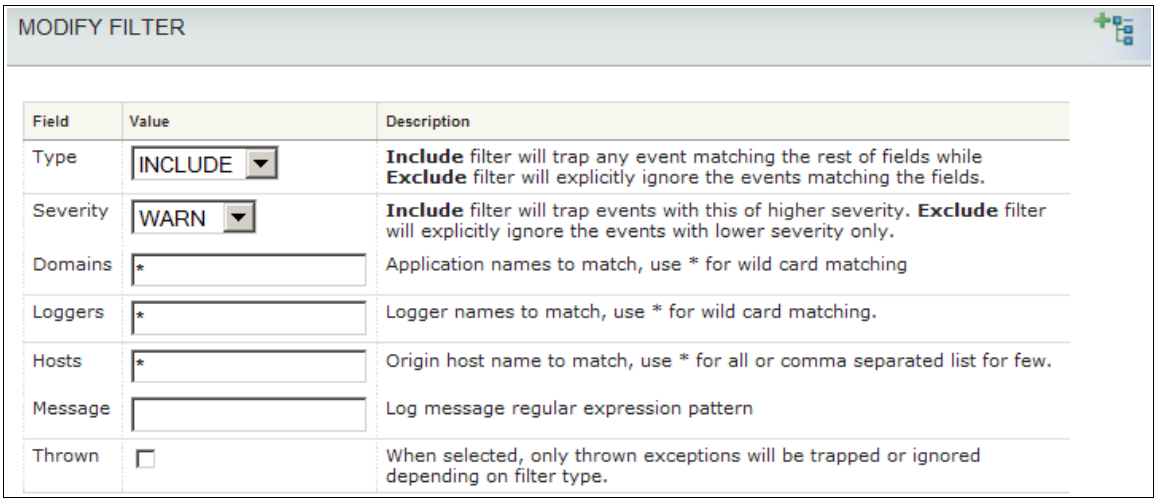


Figure 2-12 Include filter

This filter will catch anything with severity WARN and higher. This way by combining two filters we are going to count all events with WARN+ severity except hibernate, but if there is an ERROR in hibernate, we still want to count it.

Once filters are specified, you can define options as to what actually should be done when events are trapped by the filters. This is done in trigger options :

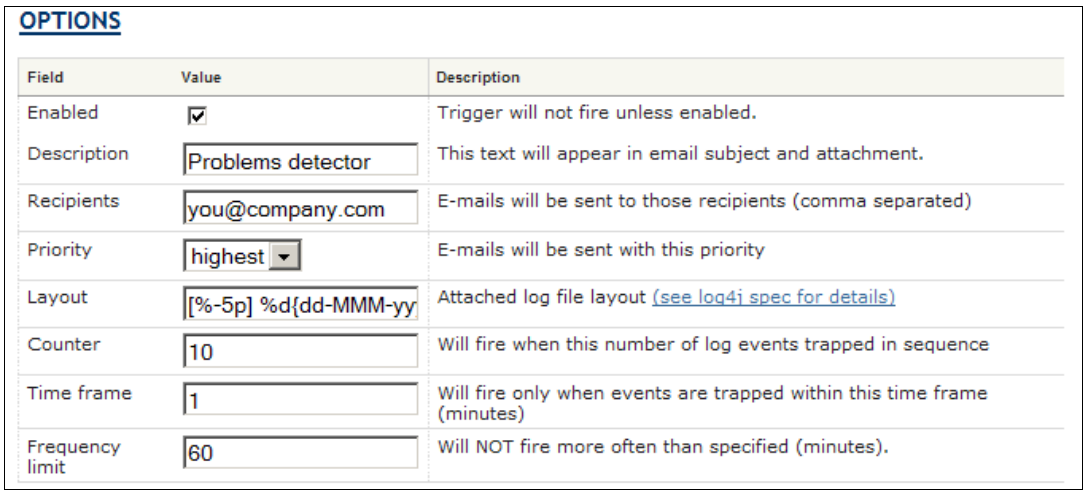


Figure 2-13 Trigger options

- **Enabled** – when not enabled, the trigger will not fire, so you can keep it in the list until you really need it.
- **Description** - Textual information which will be sent out with an email.
- **Recipients** - To whom this trigger may concern - comma separated list of e-mail recipients.
- **Priority** - E-mail priority (highest, high, normal, low, lowest)
- **Layout** - The layout of log file which will be attached to the email.
- **Counter** - The trigger will fire only at least this many events are trapped by the combination of filters available for this trigger.
- **Time frame** - The trigger will fire only when all those events are trapped within this time frame in minutes.
- **Frequency limit** - This is to avoid notification flood, - the trigger will not fire more often than specified by this value in minutes.

2.5.6 License

License tab displays currently installed license information as well as allows you to install new license file. When you install logFaces Server for a first time, it automatically activates one time trial evaluation period for 30 days. If you decide to purchase a license, the license file should be submitted through this form:

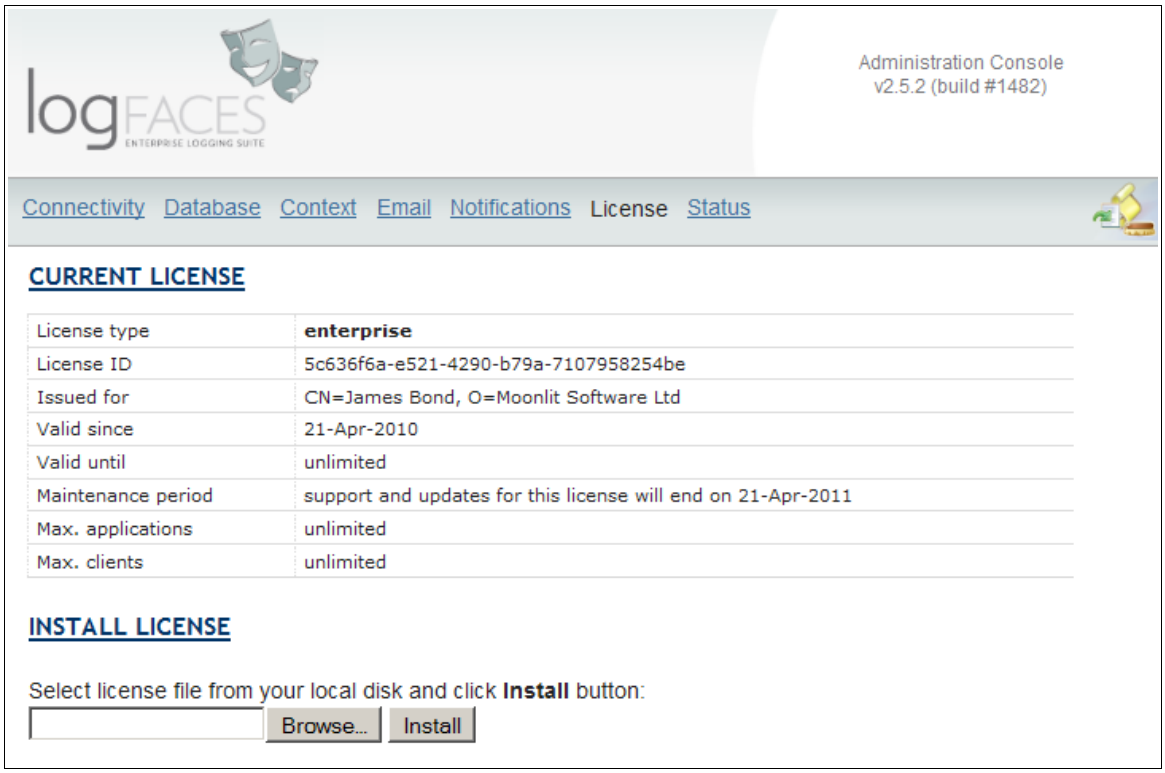


Figure 2-14 License Panel

What happens when evaluation license expires?

LogFaces Server will shutdown its engine and only allow Administration Console access; applications will not be able to append to logFaces and clients won't be able to connect to it.

When you install new license, the engine should be started manually. This can be done by simply restarting the whole service from command prompt or control panel, or from the Status panel link – see the next section.

2.5.7 Status

Status tab contains useful health monitoring information of the Server and allows basic instrumentation tasks.

Status item	Value	Action
Server version	2.5.1.1311	check for software updates
JRE version	1.5.0_16	
Engine status	running	stop engine
Last errors	none	
JVM memory max/free (MB)	508/234	run garbage collection
Number of threads	33	fetch stack traces
Load (events per second)	Moving average: 0, Max: 0	
Number of active clients	0	
Number of active applications	0	
Domains/hosts/loggers	200/1/10	
Connected to database	yes	
Number of records	115717	re-create database
Database name	Apache Derby	
Database version	10.4.2.0 - (689064)	
Database driver	Apache Derby Embedded JDBC Driver	
Database driver version	10.4.2.0 - (689064)	
Database size	652 MB	
Next maintenance time	03-03-2010 00:00:00	do maintenance now
Internal log	-	download

Figure 2-15 Status panel

Most of the information in this tab should be self explanatory for technical people. We will just mention the important instrumentation actions which are available from the links on the right side of the table:

- **Engine start/stop;** sometimes it's required to put the server down without actually shutting the process down. One of the typical uses of this option is when trial license expires. In such case, the logFaces Server will start so that you would be able install proper license, but its engine will be down and no logging will be taken from applications.

- **Run garbage collection**; explicitly call garbage collection now

- **Fetch stack traces**: will download full dump of all threads currently running on server.

- **Last errors** is a list of latest errors encountered by server, you can browse through them to see if anything went wrong lately, or simply reset them.

- **Re-create database** allows to remove all database records; be careful with this operation, it is not recoverable and can't be undone

- **Download internal log** allows fetching the log produced by the logFaces itself. In case you experience problems, this information will be very valuable to provide support. When you click on the **download** link, the server will create internal log file and prompt you to save or open it in text editor. This file can be sent to our support team.

2.6 Advanced configuration options

There are some options which can not be configured through the Administration Console but can be configured manually in several configuration files.

2.6.1 Environment properties - /conf/environment.properties

Environment properties is a collection of properties which fed into server JVM upon start up, it contains system wide parameters; some are mandatory and some are optional - see table below :

Property	Mandatory	Description
com.moonlit.logfaces.config.server	yes	Points to a main configuration file
derby.system.home	no	Derby home only relevant when embedded driver is used
com.moonlit.logfaces.config.hibernate	yes	Points to hibernate configuration file
com.moonlit.logfaces.config.schema	yes	Points to schema file which will be created in database.
com.moonlit.logfaces.config.jobs	yes	Points to jobs configuration file
com.moonlit.logfaces.config.reports	yes	Points to reports configuration file
com.moonlit.logfaces.url.revision	no	URL for checking software updates
com.moonlit.logfaces.url.downloads	no	URL for update downloads
com.moonlit.logfaces.monitoring.highThreadCount	no	Maximum number of threads the server should be able to sustain, if during run time the amount of threads will be higher, the server will issue an internal warning which will appear in Admin. status panel. Default is 300.
com.moonlit.logfaces.monitoring.lowMemoryThreshold	no	Minimum of free heap memory specified as a percentage of maximum heap memory. When free memory will go below this value, the server will issue a warning which will appear in Admin. console. Default is 2.

Table 2.1: Environment properties

2.6.2 Main configuration file - /conf/lfs.xml

Element or attribute	Description	Default
remoting/port	Port number for client connections	8050
remoting/sessionTimeout	Timeout in minutes after which inactive clients are kicked off.	2 minutes
email/emailServer	Outgoing e-mail server name to use when sending e-mail from logFaces	
email/emailAccount	Outgoing e-mail account name to use when sending e-mail from logFaces	
email/emailPassword	Outgoing e-mail account password to use when sending e-mail from logFaces	
hub/receiver/name	Name of log receiver – XMLTCP and/or XMLUDP	
hub/receiver/class	Receiver class, internal use only, don't modify.	
hub/receiver/port	Receiver port number	TCP: 55200 UDP: 55201
hub/receiver/enabled	Enables/disables receiver	
database/commitBuffer	Size of commit buffer used to insert log events into database in a batch operation.	100
database/dayCapacity	Capacity of data storage in days	3
database/recoveryRate	When unable to commit to database, this parameter specifies how often to retry re-connection in minutes.	1
database/nofRecoveryAttempts	How many retries to make in order to re-connect to database after failure before giving up.	5

Table 2.2: Main configuration file

2.6.3 How do I tune the server for the best performance?

There are several things you can do in order to optimize server performance in your environment. If your system transmits heavy load of log data, first thing you might want to do is to adjust JVM heap size - read "How do I increase server JVM memory" paragraph below. To know whether you should modify the default heap size, go to admin. console status tab and verify what is current free memory size. If it frequently goes below 2-5% of total allocated memory, consider increasing the heap size. Server has a built-in mechanism for monitoring its free memory size. When it hits critical values you will be seeing entries in Last Errors - if you see those then you definitely should consider increasing the heap size.

Server uses extensive buffering mechanisms in order to adapt for spikes in data loads. We use **ehcache** to disk overflow the events which pile up in RAM and can't be processed that quickly. There are two disk overflows you can manipulate - router overflow and database overflow. Router is a network component which receives logs from network clients, its purpose is to dispatch events further into the system - to database and to users. In order to avoid loss of data and smoothly process events during heavy load spikes, you can specify how router should cache its data. Another overflow component is the database. Usually databases are the slowest part in most systems and must be taken care of properly. Both, router and database overflows are configured in **/conf/lfs-cache.xml** configuration file which specifies how caching on disk should be done. This file is a typical **ehcache** configuration file - please refer to ehcache user guide for more details.

Another parameter worth noting is **commitBuffer** size defined in **/conf/lfs.xml** configuration file. Data committed to database in bulks, the size of commit buffer specifies the size of this bulk. Typically, in heavily loaded systems you will want to increase this buffer size.

2.6.4 How do I change user name and password for administration login?

- Stop the Server
- Open /conf/realm.properties file in text editor. There is only a single record in this file which has the following format: [userName: password, admin]. Change first two fields as you like
- Start the Server; you should be able to login into the Console with new credentials now.

2.6.5 Can I modify database schema?

Yes, you can do that to a certain extent. Database schema file is referenced in **/conf/environment.properties** through property named **com.moonlit.logfaces.config.schema.**

You can not modify the structure of the tables or column names because they're mapped through hibernate mapping in the code. But you can adjust column size constraints, modify indexes or add some additional statements as long as they don't break the mapping.

After you changed the schema file you need to re-create the database. This can be done either from administration console status tab, or by using some other external tool. Note that when using embedded database, there is no other choice but using the admin. console.

IMPORTANT: If you care for the existing data in the database, make sure to back it up before doing any changes to the schema. One of the options is to use our backup utility described later in this document.

2.6.6 How do I work with external databases?

You have to obtain relevant database driver from your database vendor and place the jars in **/lib/dbdrivers** directory on server. **Our installation only include drivers which permitted by publisher's license**. We do our tests for Oracle, MySQL, SQL Server, DB2 and PostgreSQL, but theoretically there shouldn't be a problem to work with other relational databases as well. It's only a matter of configuration and database driver you wish to use.

Look at **/conf/environment.properties** file – you should see these two properties pointing out to hibernate configuration file and database schema :

com.moonlit.logfaces.config.hibernate=${lfs.home}/conf/hibernate.properties

com.moonlit.logfaces.config.schema=${lfs.home}/conf/lfs.sql

You can modify these references by pointing to different files, but make sure the files are correct. Our distribution contains both hibernate examples and database schema for all databases we support at this moment. The example above will use embedded database driver settings and Derby schema. If, for example, you would like to use PostgreSQL, modify environment.properties as follows :

com.moonlit.logfaces.config.hibernate=${lfs.home}/conf/hibernate-postgree.properties

com.moonlit.logfaces.config.schema=${lfs.home}/conf/lfs-postgree.sql

Then modify hibernate properties file to point to your database. LogFaces Server uses Hibernate ORM framework; it is recommended to have good knowledge of this framework before you decide to make re-configuration.

Once you prepared the configuration, simply restart the logFaces server. During the start up new schema will be published automatically. After the start up, open **Administration Console** and navigate to the **Status** tab. If everything went well, you should see that engine is started, database connection is OK and there are versions of database and driver in the table. If something goes wrong and there are problems with database connection, you will see red marks and last error numbers. To see what happened, click on "show last errors" or "download log file" link; the problem is usually related to a configuration error, typo or perhaps your database is not responding as logFaces expects. If you're unable to figure out the problem yourself, submit this log file or an error code to our support site and we will try to help.

Another, perhaps easier, option is to run the server in console mode and see that there are no exception thrown during it's start up. On Windows you can run server in console mode from **/bin/run.bat** on. On Linux, run the following command: **./lfs console** while in **/bin** directory.

2.6.7 How do I make logFaces win32 service depend on other services?

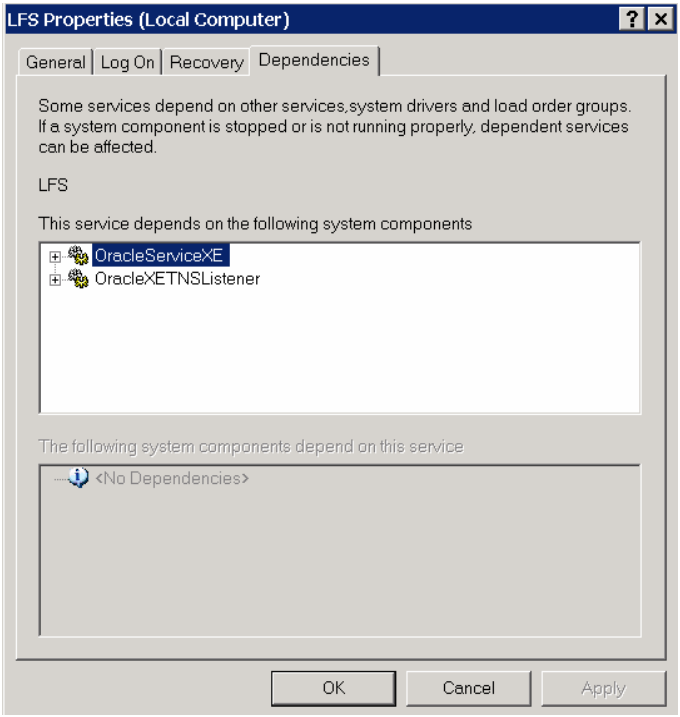
In Windows it's often required to have logFaces service dependent on other services during start up. For example, if you run database server on the same machine as logFaces server, you may want to make sure that it starts only after database successfully starts. To achieve that, you must specify service dependencies.

Open **/bin/lfs.conf** file and modify **wrapper.ntservice.dependency.xxx** properties accordingly. For example:

wrapper.ntservice.dependency.1 = SomeService1

wrapper.ntservice.dependency.2 = SomeService2

Make sure to preserve proper numbering order and specify correct names of dependent services. Then you will need to re-register LFS service. Make sure the service is not running; execute **/bin/uninstallservices.bat** and then **/bin/installservices.bat** – this will re-register service configuration in Windows registry. To verify that everything went OK, check out service properties in your Windows administration tools. For example, in case of Oracle, the dependencies should look similar to what is shown below. If everything looks OK, go ahead and restart your computer to verify that dependencies actually work.



2.6.8 How do I backup my database storage?

You can perform backup of entire database storage into a binary file. This backup file can later be re-imported into another system. We provide a script to fully automate this process - /bin/backup.bat on Windows or /bin/backup.sh on Linuxes. Generally this is a good idea to stop the server during backup process, but this is only required for when you use embedded databases, external databases can be backed up even while server is running.

To import backed up data into another system, go to client Tools menu and select "Import data into database". You will be prompted to select the backup file. Depending on the amount of data, the process may take some time to complete.

2.6.9 How do I increase server JVM memory?

Open **\bin\ lfs.conf** file – this is a bootstrap configuration file.

JVM memory is setup with those attributes:

wrapper.java.initmemory=256

wrapper.java.maxmemory=512

Those values are default, if you experience extensive memory usage, try to increase maxmemory property value.

2.6.10 How do I increase client JVM memory?

Client installation directory should contain a file named **logfaces.ini**. If it doesn't exists – make it manually (it should be in the same folder as **logfaces.exe**). The following parameters define JVM heap sizes, make sure those lines are present as shown :

-vmargs
-Xms64m
-Xmx256m
-XX:MaxPermSize=128m

3 Getting started with logFaces Client

LogFaces Client is a rich GUI application which can be installed anywhere on the network to obtain log data coming either from logFaces Server or directly from your applications.

3.1 Installing logFaces Client

LogFaces Client is distributed for multiple platforms as a zip archive. Simply unzip it and that's all there is to it. On Linux you will have to set the executive permission to the file named "**logfaces**". Before running you may want to adjust **logfaces.ini** for better memory allocations.

3.2 Modes of operation

LogFaces Client can work in two modes - **Client Mode** or **Server Mode**. You select the mode during application start-up. In Client Mode the application connects to and works with remote logFaces server instance. In Server Mode the application acts as actual log server.

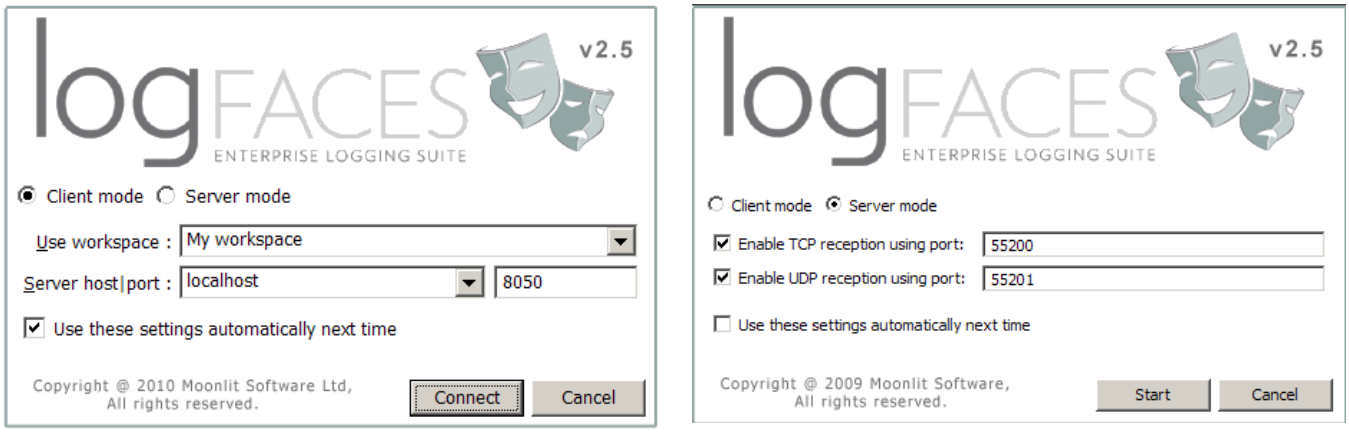


Figure 3-16 Modes of operation

In **Client Mode** we specify remote logFaces server **host** name and **port** number. Application will remember these options for the next time you run it, but you can also indicate to use those settings automatically in the next time and not asking again. It is possible to modify these settings in the File/Preferences menu later. The communication between logFaces Client and Server is one way (from client to server) and is HTTP based, so normally there shouldn't be a problem with firewalls. Of course, the access through the given port should be allowed by your network administrator.

In **Server Mode** the application runs with embedded **compact** version of logFaces Server. This is a limited (and less expensive) version of server and client combined into single application which provides **only real time viewing** of log data. You can use it when you don't need database and other

features available in standalone logFaces Server. In order to run in **Server Mode** we need to specify at least one of the ports which will be used by the application to receive log data from appenders in your applications. You can specify either TCP, or UDP or both, just make sure those ports are available and your application appenders are configured to log into this host and those ports.

Both modes look very much alike from user experience point of view, except that in Server Mode there is no database and querying features.

*Note that in order to run in **Server Mode** you need to install the license on the computer where you run it. As with the logFaces Server, the first time launch will automatically activate 30 days free trial. Note also that this is not the same license type as installed on the logFaces Server. This license needs to be purchased and installed separately unless you hold OEM license.*

3.3 Workspace

When working with several servers, you often need to switch quickly from one system to the other. Workspace is designed just for that, it remembers all the settings you make during your connection - server connection end points, real-time perspectives, queries, counters, etc. You can create as many workspaces as you like and switch between them instantaneously without restarting the application.

File menu contains all workspace related actions :

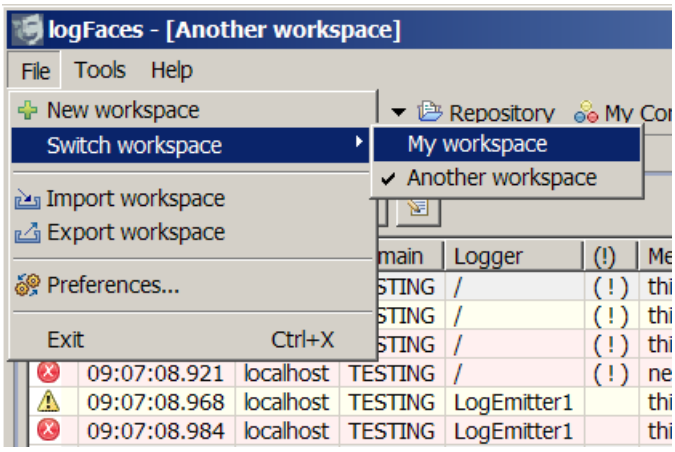


Figure 17: Workspace actions

3.4 Layout

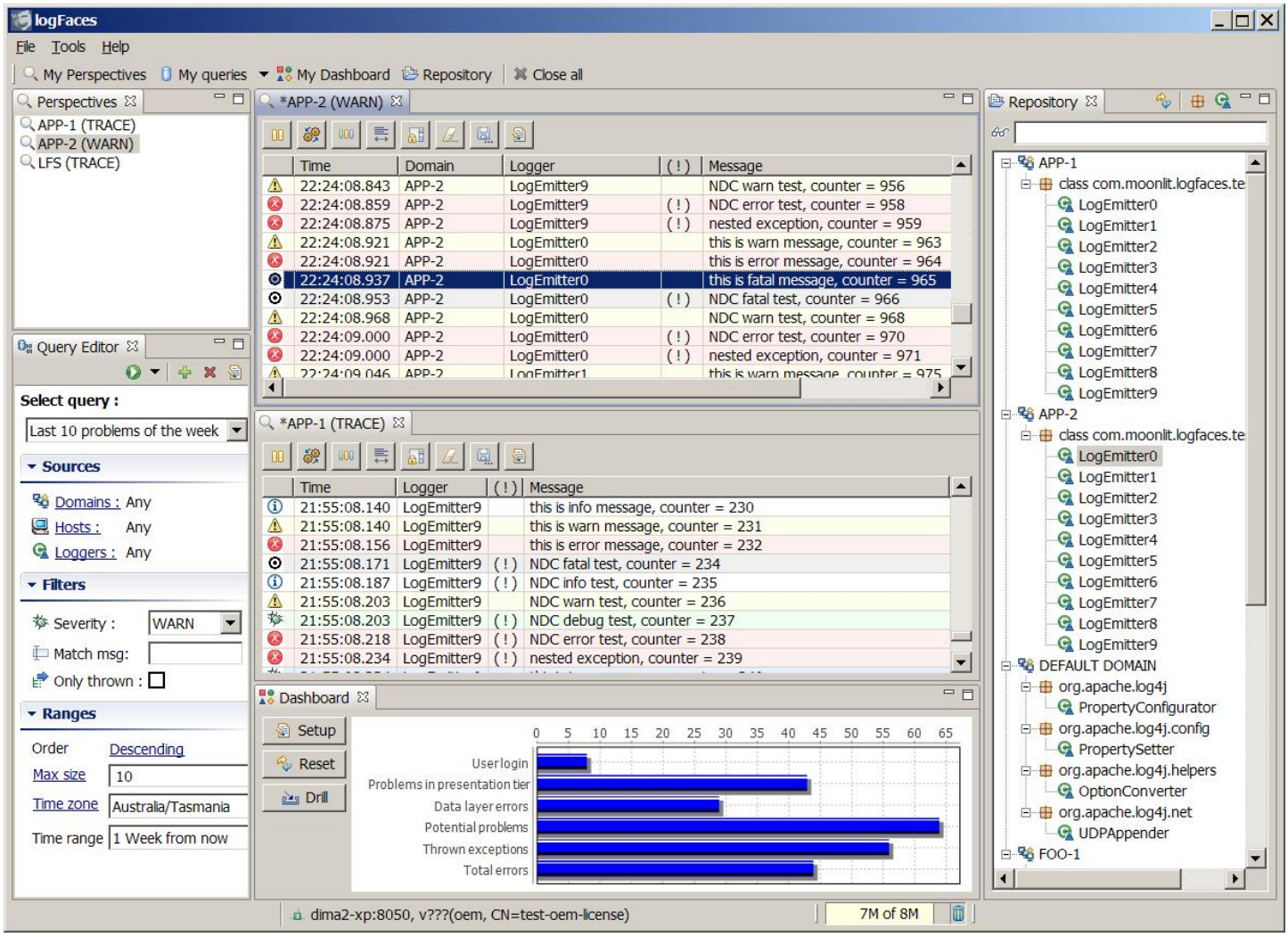


Figure 3-18 Application layout

LogFaces Client is an Eclipse based application; not only it has a look and feel of the Eclipse but is based on the Eclipse. So, if you're familiar with Eclipse, then using the application will be a breeze.

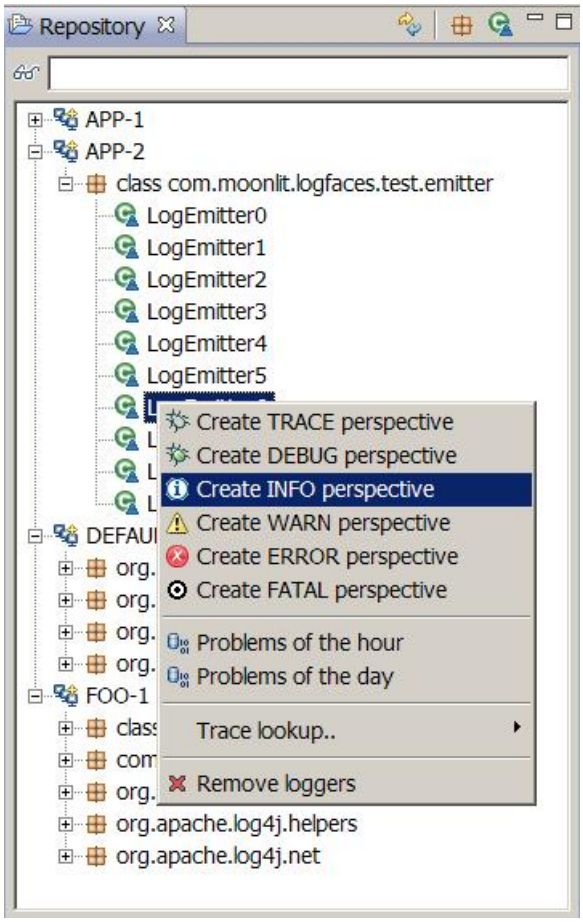
Top left part by default contains Views - for accessing log perspective you create. You can use mouse right click for activating, adding, removing or modifying those perspectives.

Bottom left part by default contains Query Editor for launching and managing database queries.

Rightmost part by default is occupied by Repository which is a list showing which loggers and what applications are known to the logFaces Server at the moment.

The middle part of the screen will be containing active log perspectives and/or database queries you execute. All parts of the screen can be organized to suit your taste; the top toolbar under menu gives you a quick access to the views.

3.5 Repository



Repository is a view containing loggers which are known so far to the logFaces Server. It has two purposes. First of all, it's informative; it shows you what your system looks like log-wise. Secondly, Repository is a quick start for creating perspectives, getting fast access to the errors and deleting unwanted loggers. Right click on it's content to get available operations.

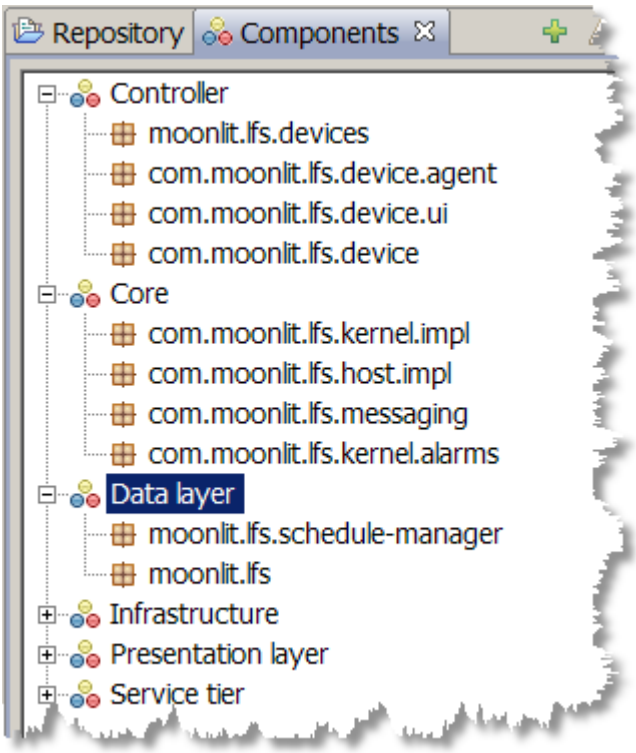
Repository structure is tree-like. The top level is called Domain (or Application). This information is extracted by the server from incoming events by looking into specific property, called "Application". This property is specified when you integrate your application with logFaces. If this property is not specified, the Server will automatically create "Default Domain" and associate all unknown events under this domain. It is highly recommended to setup your application that it provides unique Domain name – it will further help you with quires and give more organized view on your system.

Under Domain resides a Package – leftmost part of the logger name. Some applications might not have any package, it is not a requirement.

Under Package we have loggers - actual sources of log data.

IMPORTANT: When you run your system for the first time, the Repository might not contain all the loggers you actually have in your system – the Server will know your loggers only when they log at least one statement. It is recommended to let your system run for a while before you actually start using the Repository. Also, as your system evolves and there are new loggers (classes?) introduced, you might want to do the refresh to make them appear in Repository. Otherwise, new loggers will not be visible by the Client (unless it gets restarted).

3.6 Components



A component is simply a collection of packages defined on the user level. Why do we want it?

Two points; one - to be able to work with a large number of packages easily, and two - let others (e.g. QA people) to see the system from higher level.

Now, component is only a visual representation which you use to cut your system into a larger blocks. Once you have done that, you will be able to access the log data instantly, or filter it out, or focus on it. Real quick. Just a convenience, but we think it should be very handy particularly for those who swim in an ocean of packages and loggers.

3.7 Creating log perspectives

Log perspective is a real time view on the log stream going through the logFaces Server. Perspectives are organized by name so that you can refer to them later. Each perspective is simply a set of filtering rules telling the Server what logging events should be routed to it. Server has no limitation on amount of perspectives you create. Only when you actually activate the perspective the Server deals with it, other than that Server doesn't know about perspectives. Number of active perspectives is only limited by the physical memory the Server is using. There are two ways the perspectives can be created – "quick" and "advanced"; both ways are actually trivial, but we differentiate them here for the sake of clarity.

The "quick" way is from Repository or Components views - select loggers, right click on selection and choose the severity option you need. Then simply give the name to your perspective and you're done. This will create a perspective whose loggers are all set into the severity you specified. Later you can modify those settings by means of this dialog :

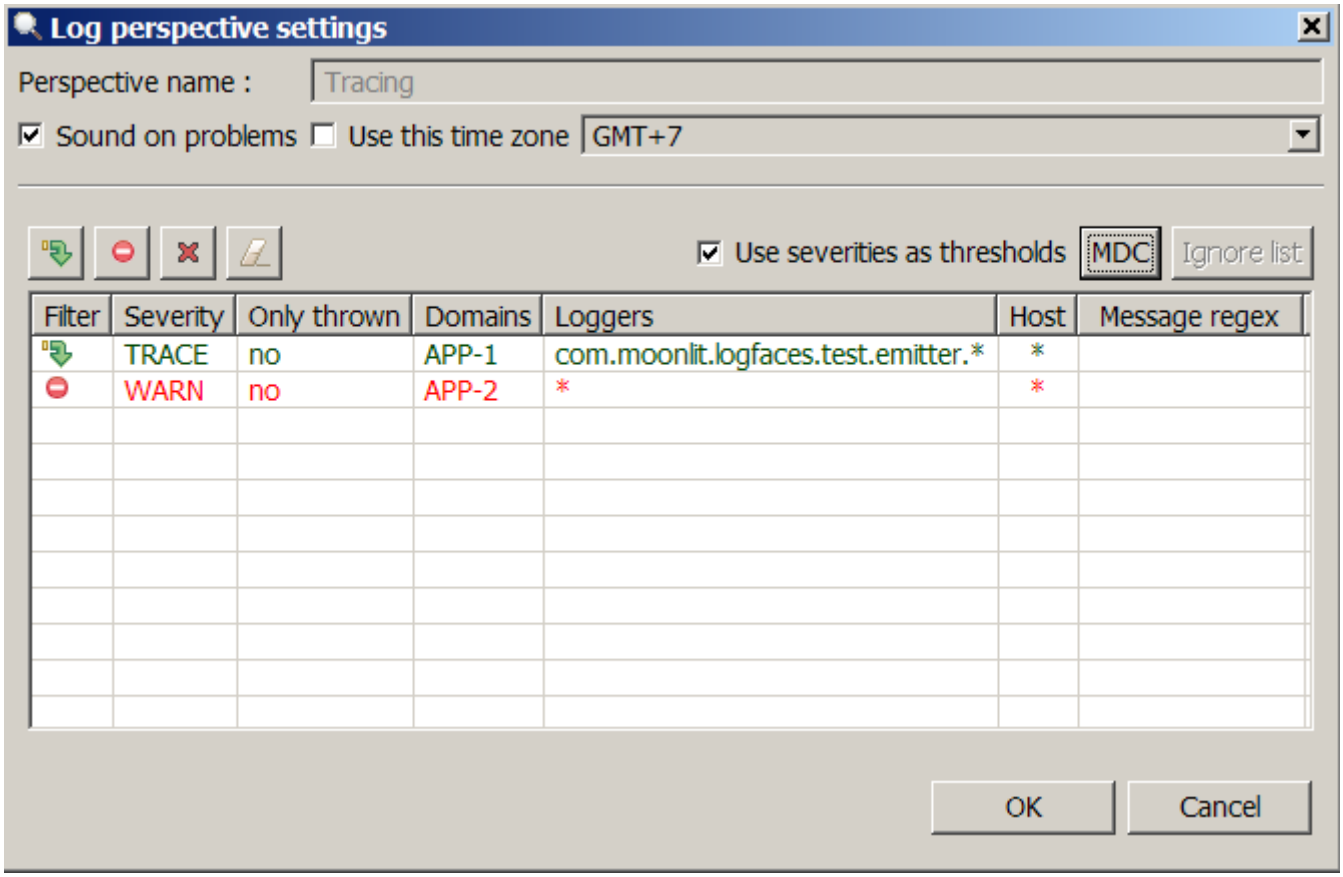


Figure 3-19 Perspective settings dialog

The check box "**Sound on problems**" is very useful if you want to have sound notifications. Your perspective will make a gentle beep when messages with WARNING+ severity will be going through. The frequency of the sound notification is limited to 5 seconds; in any case, if your system is not stable and there are lots of errors, this feature would be annoying. Its best usage is when every error in the system requires your attention and it doesn't happen too often.

You can specify **time zone** which will be applied to time stamps automatically when log events are displayed. Alternatively, default current time zone will be applied.

Ignore Locations is a collection of source locations which will be filtered out even if the logger passes through the filters applied to it. Ignore list is convenient when you want to shut off some monotonous messages from particular location in the code. Each entry in the list is described by file name, class, method and line number from which the log was made.

Mapped Diagnostic Context is a collection of name-value pairs which can be applied to every event going through the filter. When specified at least one MDC, all events will expect to have given name-value pair to pass the filter.

Use severity as threshold is a default option, it indicates that higher severity levels will be automatically included by filters. Alternatively, if you deselect this option, only events with exact severity will pass through filters.

And finally, you can specify combination of filters which will be applied to the stream of logging events. There are two types of filters - includers and excluders. Include filters will explicitly trap the events matching the parameters of the filter, while exclude filters will explicitly ignore events matching the parameters. A combination of 6 parameters define filter.

- **Severity level**. Depending on filter type this parameter works differently. For include filters, it will match all events with higher severity. For exclude filters it will exclude everything except the higher severity levels.

- **Only thrown -** pays attention to whether the event is thrown or not. If this parameter set to true, for include filter it will pass ONLY thrown exceptions while exclude filter will ignore exceptions.

- **Domains, loggers & hosts** - matches the names of domains, loggers and hosts respectively. Use wild cards (*) to match few. Include filter will pass every event matching specified pattern while exclude filter will discard all events matching it.

- **Message regex** - matches actual message text of the events. Same logic of include and exclude filters apply.

3.8 Real-time perspective view

Perspective view is actually the log which goes through the filters you specified.

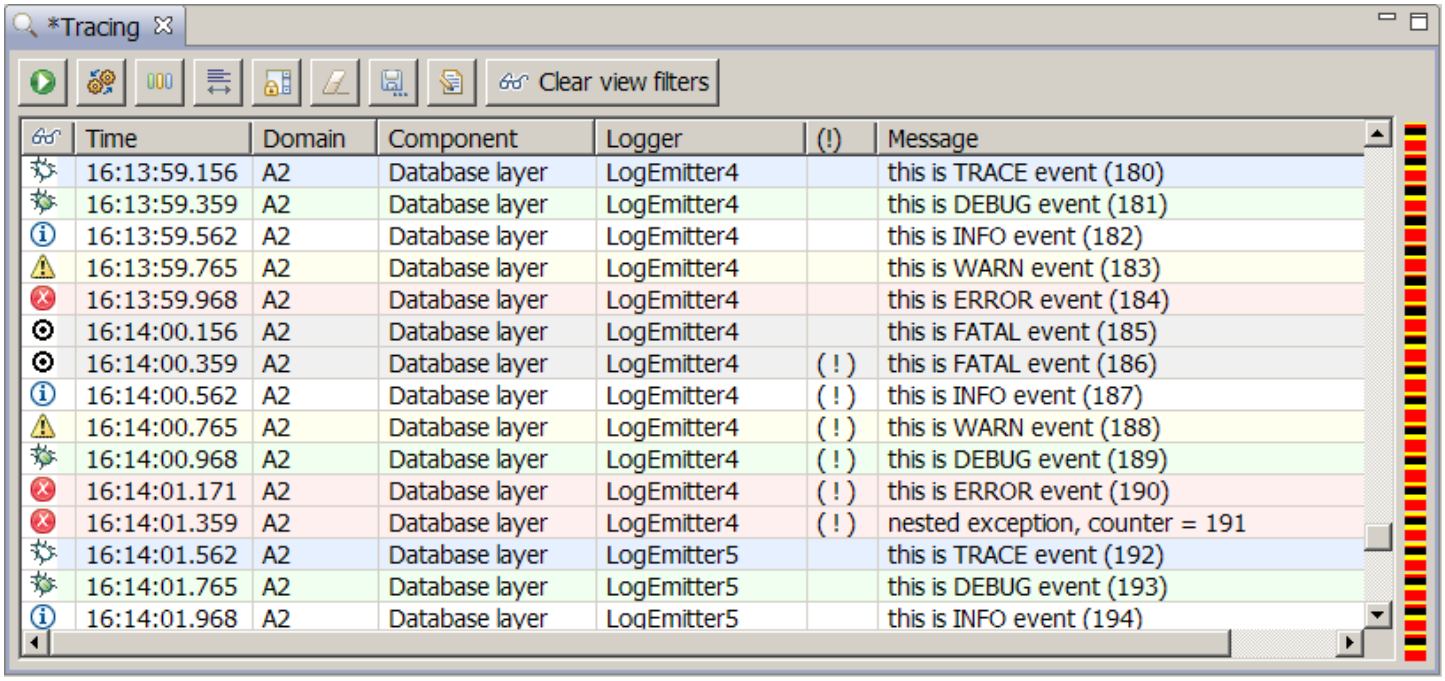


Figure 3-20 Perspective view in real time

Logging events are organized in a table and displayed in real time as events arrive from logFaces Server. Lines are colored by the severity which is also emphasized by the icon on the left. Severity icons and colors denoted as follows:

- TRACE; blue background
- DEBUG; green background
- INFO; white background
- WARNING; yellow background
- ERROR; red background
- FATAL; gray background

Each perspective has its own set of controls on the top of the panel:

- - Pause/resume; when perspective paused, the flow of events will be temporally stopped until perspective is resumed.

-  - Access to the settings where you can modify perspective filters. This can be done any time and applies immediately in real time.

-  – Allows you to specify what columns of logging events you want to be visible in this view.

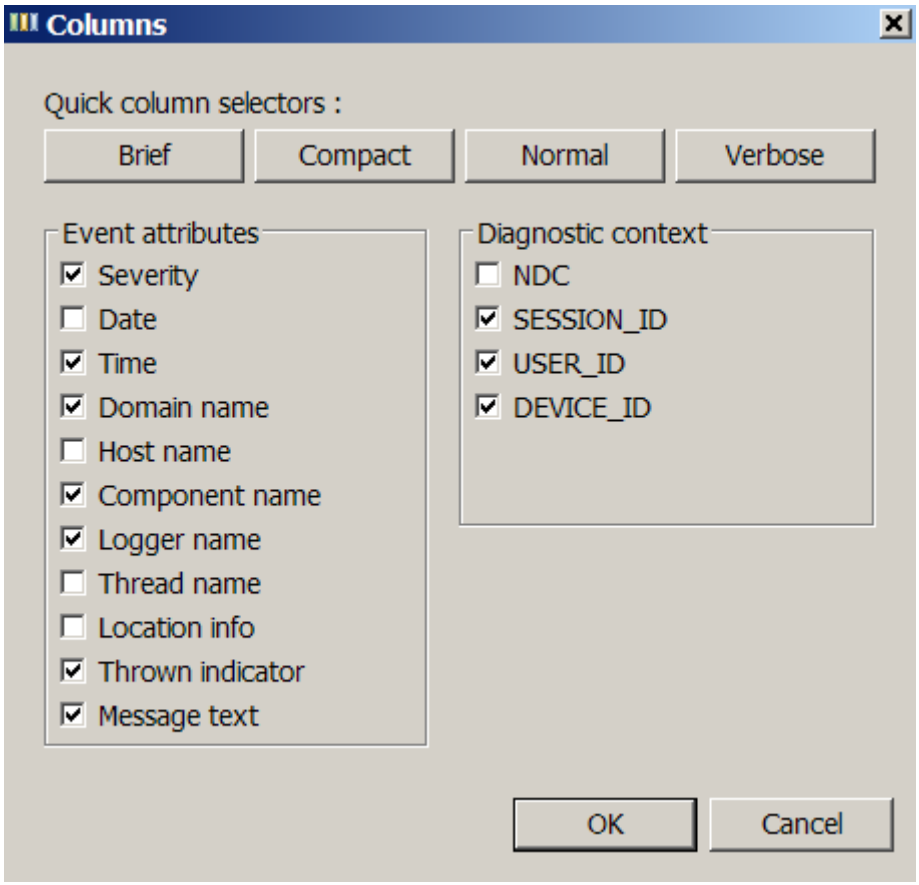


Figure 3-21 Columns selection

-  – Automatically arranges the column width to fit the content.

-  – Toggles the scrolling of the view.

-  – Erases everything from the view

-  – Allows saving the content of the view into a text file – everything you have in the view will be saved into the text file.

-  – Automatically opens view content in the text editor. In order to work with files you first should specify which text editor you would like to use as shown below; this is done only once.

3.9 Details, Message And Exception views

Whenever something is selected in the list of log events, the additional details are also shown in separate views. You may want to organize those views to better fit your workspace.

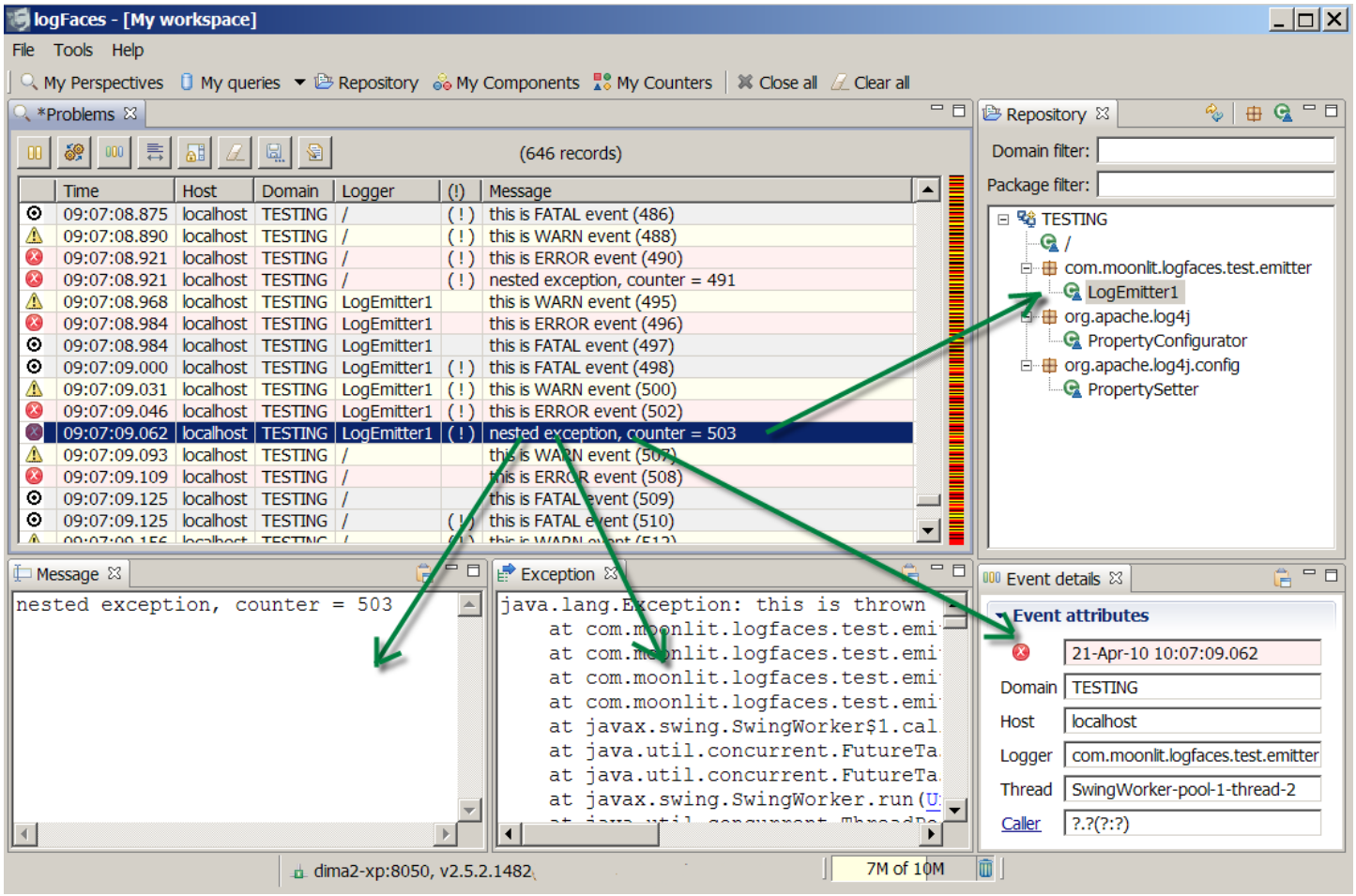


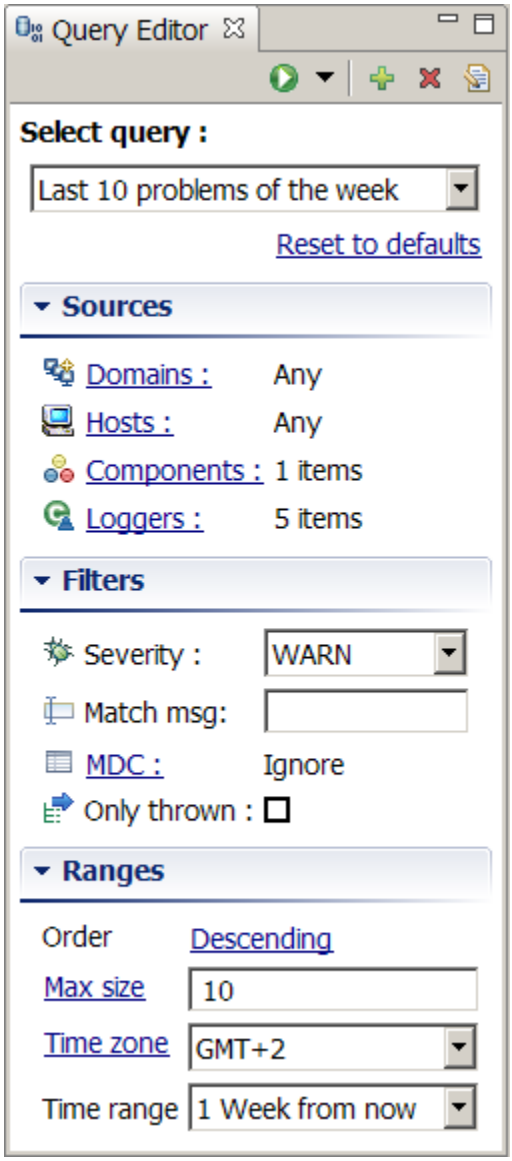
Figure 3-22 Event Details View

Details view shows all possible attributes of selected log event. Message view shows the message text in multiline text viewer and Exception view will display stack trace of the event in case it was thrown. Additionally, the selection is also reflected in Repository view highlighting current logger of the event.

3.10 Using query editor

Query Editor is for creating database queries. The queries you create then can be accessed quickly from the top toolbar, or simply by launching them in place by clicking  .

You create a new query by clicking  button. Queries are named and displayed in the list combo. In the query we specify criteria by which we want the logging events to be retrieved from the database. The options are:



- **Domains** – defines the origin domains of logging event. The query will include only those events which come from the domains you specify or from any domain if nothing is selected.

- **Hosts** – defines the origin hosts of logging event. The query will include only those events which come from the hosts you specify or from any host if nothing is selected.

- **Components -** defines list of components where logging events should belong.

- **Loggers** – defines the origin logger names. The query will include only those events which come from the loggers you specify or from any logger if nothing is selected.

- **Severity** – the threshold which will filter the events by their severity where DEBUG is the lowest and FATAL is the highest.

- **Match msg** – text matching string; will be applied to include only those events whose message text match the specified expression. Works exactly as SQL "like" statement.

- **Only thrown** – when selected will include only those events which originate from exceptions

- **Order** – the order of the results to display. Events are organized in database by time; you have an option to display the result in ascending or descending order.

- **Size** – defines the size of results to display. You never know how much data will get back from the query, so in order to be on the safe side and not to exhaust the memory of the client application, we have a limit.

- **Time zone** - if you are accessing the Server located in another time zone and want to receive queries related to that time zone, then you should specify your Server time zone here. By default the query is set to use client's time zone.

- **Time range** – specifies the time range of the logging events. There are several predefined setups, like "1 Day from now" and the like. But you can also specify **Custom Range** stating the exact start and end time.

If you want to save results of the query directly into the file, select one of the options from the  combo. You can save results to:

- Text log file; the pattern of the file will be as specified in File/Preferences
- XML file; the schema is according to log4j DTD
- Binary file; can be used for later import operations on other logFaces servers.

Once you define the query, it stays in the list and is also available in the top toolbar like this:

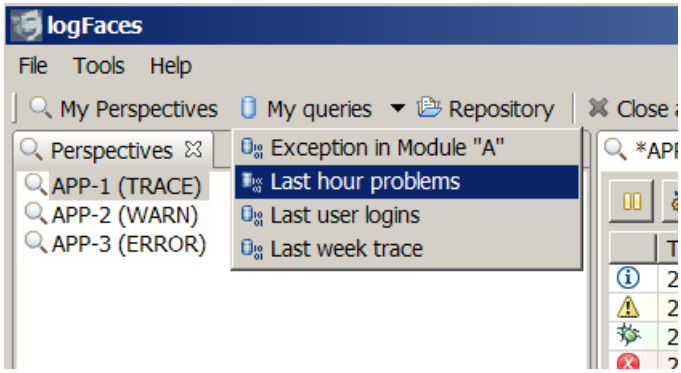


Figure 3-23 Query quick run

All you have to do now is to select it, the results will appear in the view similar to the one of the real time perspectives we have already seen. As with real time perspectives, the query can be saved to

file or opened with external text editor. This way you can create log files dynamically at will instead of having your application to manage them on local disk.

3.11 Drill down

One of the most efficient ways to dig into problems is to first grab them all and then see what happened before or after that point of time. For example, you can define a query which grabs only thrown exceptions occurred yesterday and then use right click on the query results; the drop down menu will let you drill into that particular moment. You can go backwards, forward and both ways while specifying how wide should be the gap:

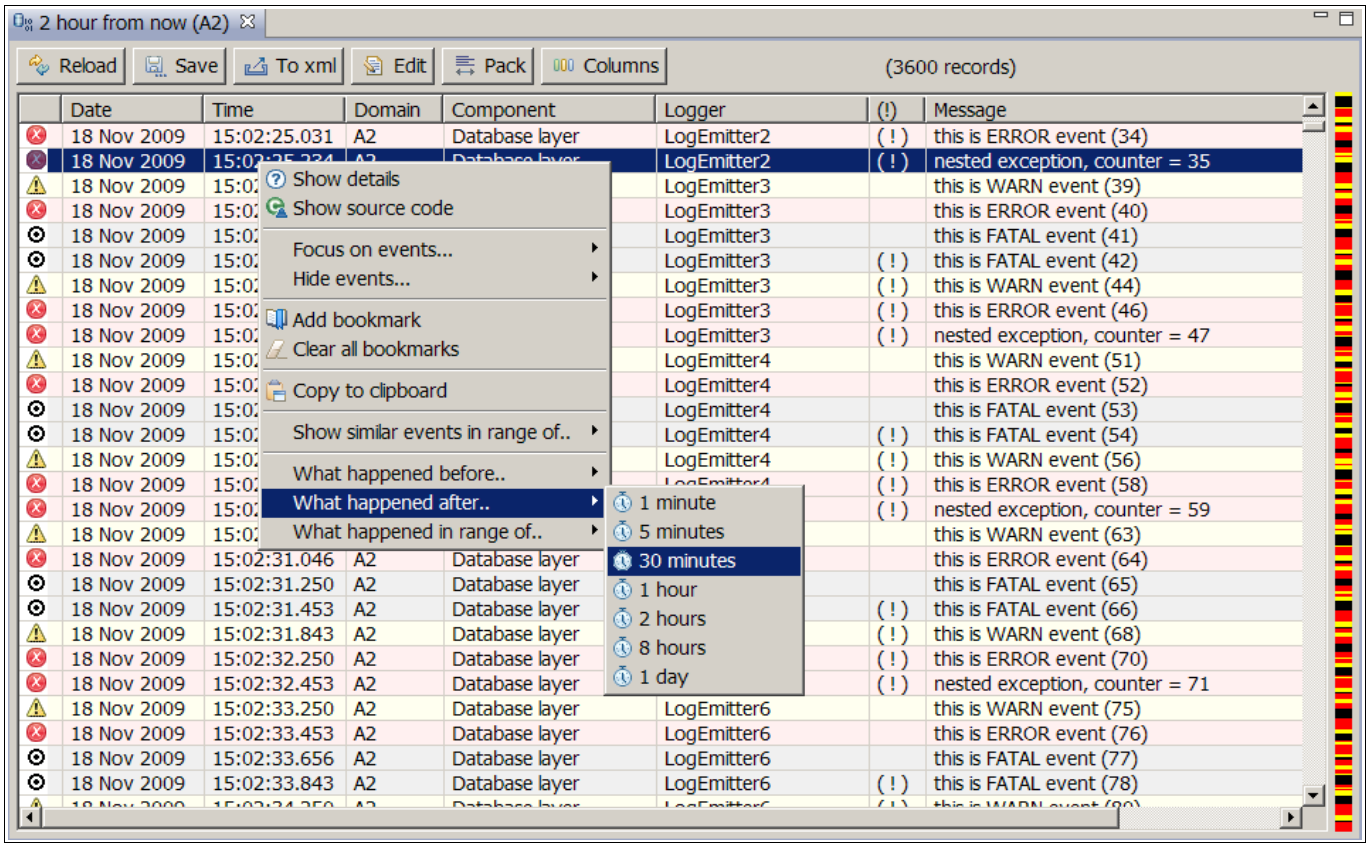


Figure 3-24 Drill down

The drill down query will open up in another view and will contain logging events related to originator's Domain covering all severity and time span you defined – this way you will see what other parts of the system were "saying" when this problem took place.

3.12 Counters

Counters dashboard is another look at your system log-wise. Counters register incrementally certain log events by filtering them out. A dashboard displays collection of counters in the form of a chart. You can create customized snapshot of your application log and see it updated in real-time. Once you have a picture, you can **drill down** and see the events of each particular counter. You can also reset the dashboard and start the counting fresh.

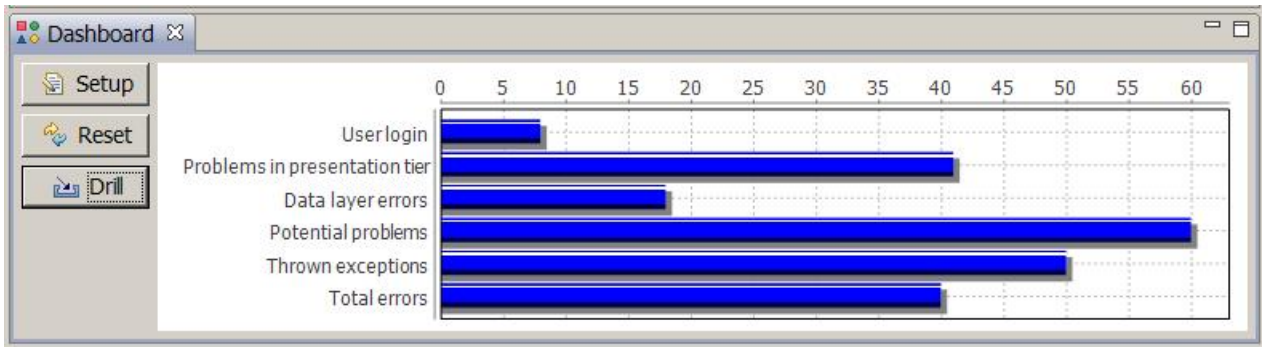


Figure 3-25 Counters

Each counter is a criteria filter by which events are evaluated. If event matches criteria – its counter gets incremented. Counters are evaluated in real-time and each client can have his/her own set of counters. In order to setup your dashboard, click on Setup button and edit the filters. You can add/remove/rename and move around your counters in the list. Right click on the repository part to mark needed loggers and specify criteria.

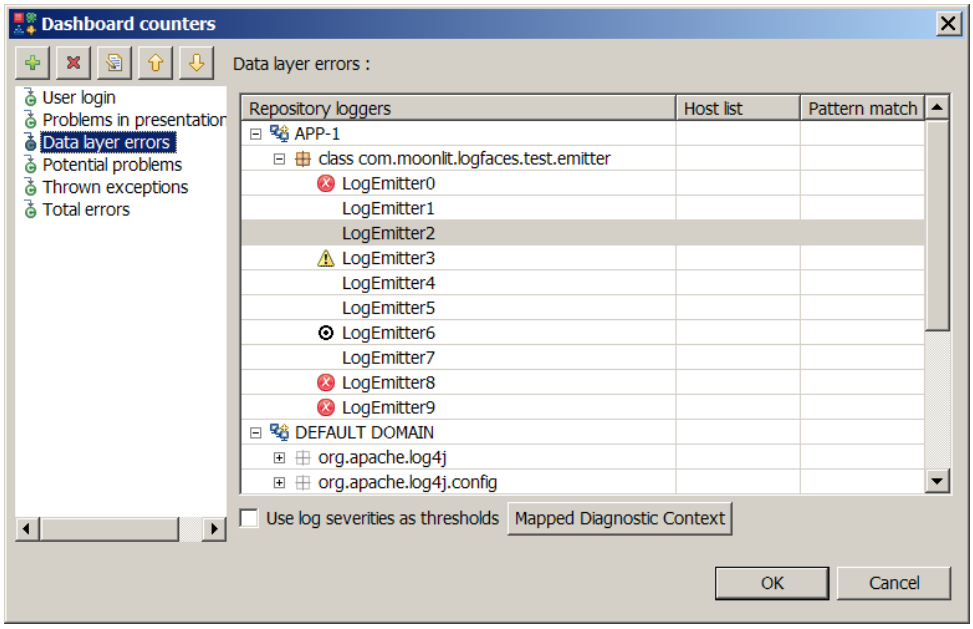


Figure 3-26 Dashboard setup

3.13 Source code correlation

You can correlate log data with the actual source code provided that:

- You application appenders enable location info option.
- You have an up to date source files available for network (or local) access.

Integration with source files is fairly straight forward – you should tell the client where to find source files. Open Preferences from File menu and go to Source Mapping section. There are two types of mapping - directory mapping and URL mapping. Whenever the source files are requested, the directories are looked upon first. Note that in both cases you need to specify the base location - the sources will be searched recursively.

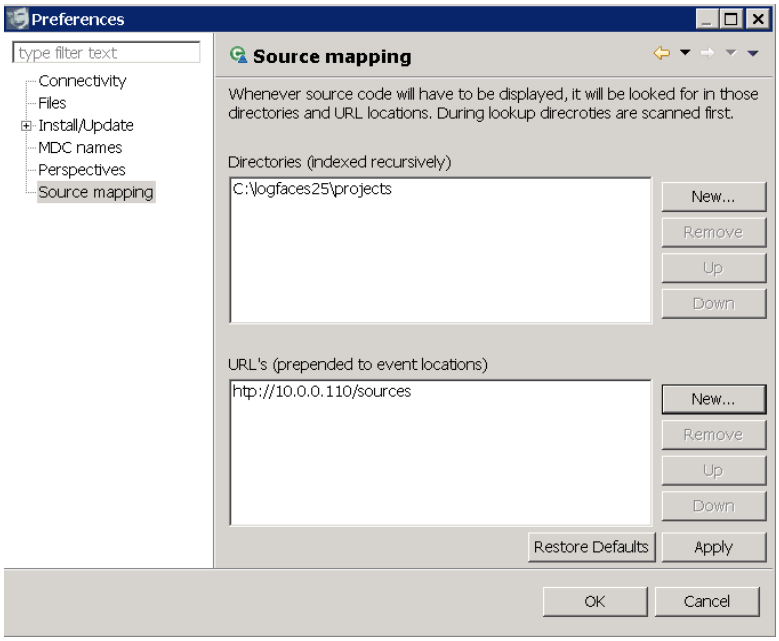


Figure 3-27 Source Mapping

Now, whenever you right click in the log view, there is an option to "Show source code". If source file is found, it will be displayed in separate window pointing to exact location in the code where the log statement is originally coming from.

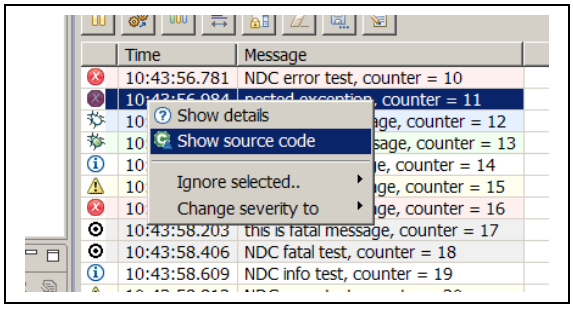


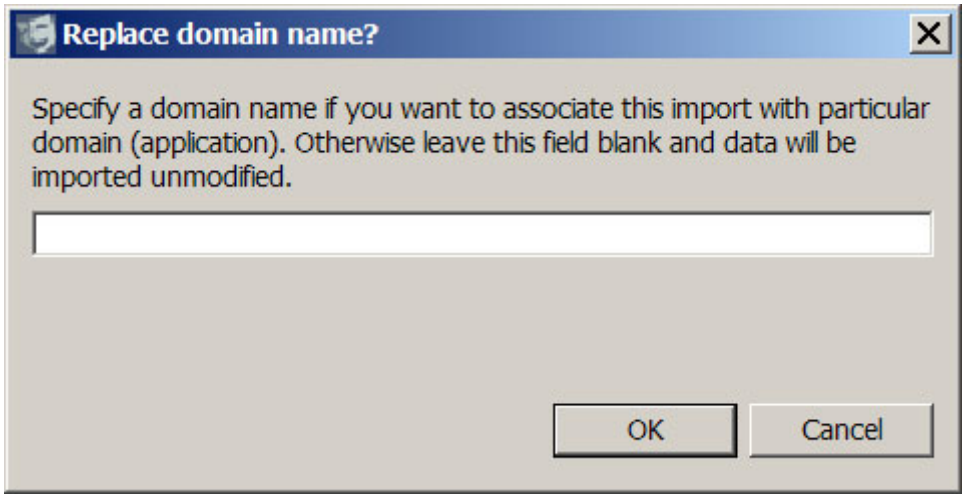
Figure 3-28 Show Source Code

3.14 Import and Export

Data can be exported from one server and imported on the other, we use binary data image for those operations. As mentioned earlier, you can export query results directly into binary file and import it later on somewhere else.

To export all data from the database, go to Tools menu and select "Export data from database". You will be prompted for the file name. This operation is identical to what is done by Backup utility.

To import data into database, go to Tools menu and select "Import data into database". You will be prompted for the **.lfb** file name – binary data exported previously from logFaces. Before the import actually starts, you will be asked to specify a Domain name to which imported data should be associated:



If you leave this field blank, the data will be imported without any modifications. Otherwise, the Domain name of all events will be replaced to the value you provide. This is very convenient when you exchange data between different logFaces servers.

Please take into account, that depending on the dataset size and the speed of the database, those operations may take up to several minutes while taking considerable amount of CPU and database I/O.

3.15 Workspace

All settings you do in the application including visual layout, perspectives and queries are saved locally on your disk. Every time the application is run those settings are restored to the latest.

You can **export** the workspace configuration into a text file and **import** it on another computer. Import/export operations are available through the **File** menu.

3.16 Preferences

There are several global preferences which are also accessible through the File menu – **Connectivity**, **Files MDC names** and **Install/Update**

In **Connectivity** section you can specify the server connection endpoints, support site URL and path to the administration console. Connection endpoints (host and port) will be affective next time you start the application. Support site URL is used for submitting bugs, feature requests or questions – you will be taken to this URL automatically when selecting **Help/Bug report** menu. Admin console parameter is the path of administration console which also can be access from the help menu – it will simply open a browser view to allow you to administer the server without leaving the workbench.

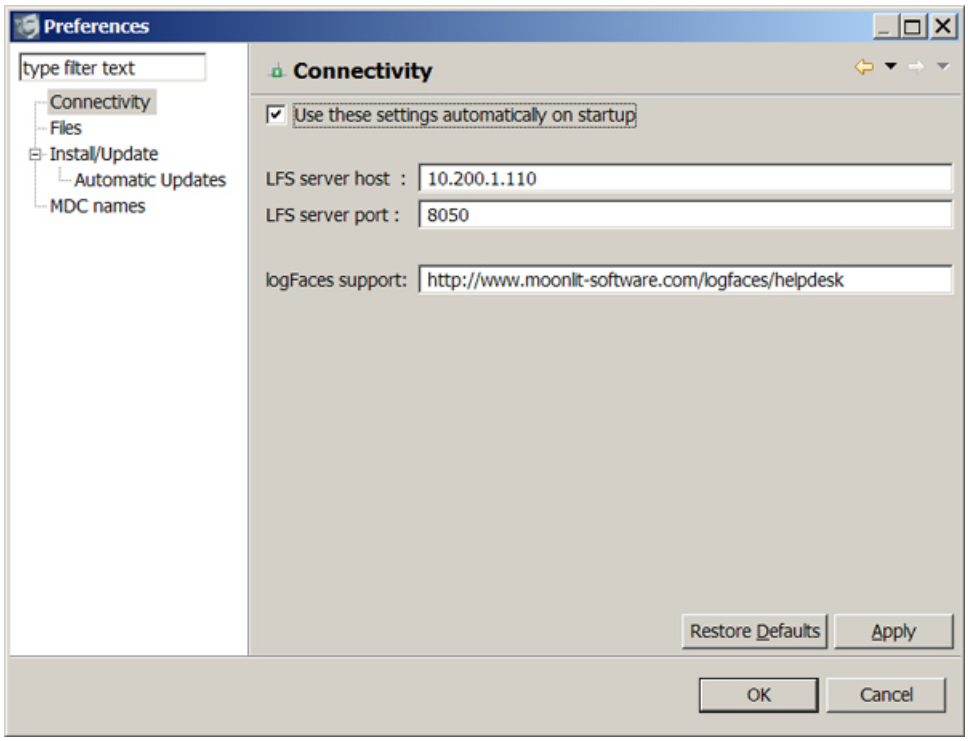


Figure 3-29Connectivity Preferences

In **Files** section you specify the layout format of log files and external editor to be used with the application. The layout format will be applied to all file related operations you do when saving queries to a file or saving the perspective views to a file. The format itself is specified in Apache Log4j documentation, you might want to read about it here.

External text editor is launched every time when you want to display the content in the text editor for doing some text operations. By default after installation the editor is not specified, you will have to manually do it by clicking on a "Browse" button here in preferences:

Source directories mapping will allow us to find source files. Make sure to specify these directories if you will want to correlate log events with source code.

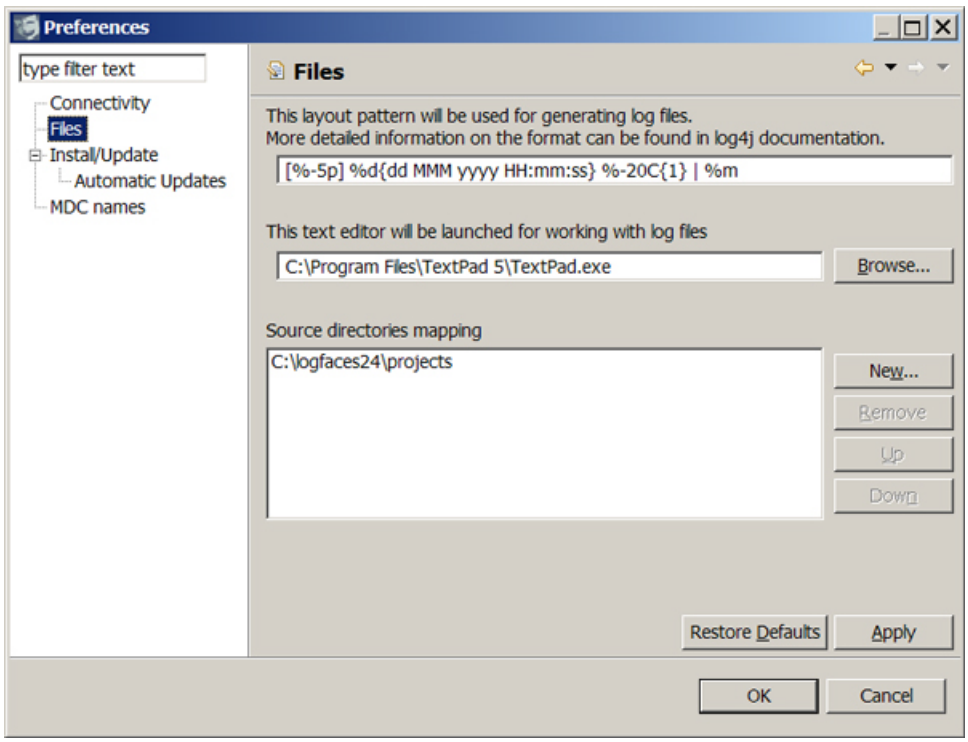


Figure 3-30 Files Preferences

In **MDC names** section you will find current MDC names mapped from applications to the logFaces server and all the way to the client. You can not modify them, but you can synchronize the names in case they were modified after client got started.

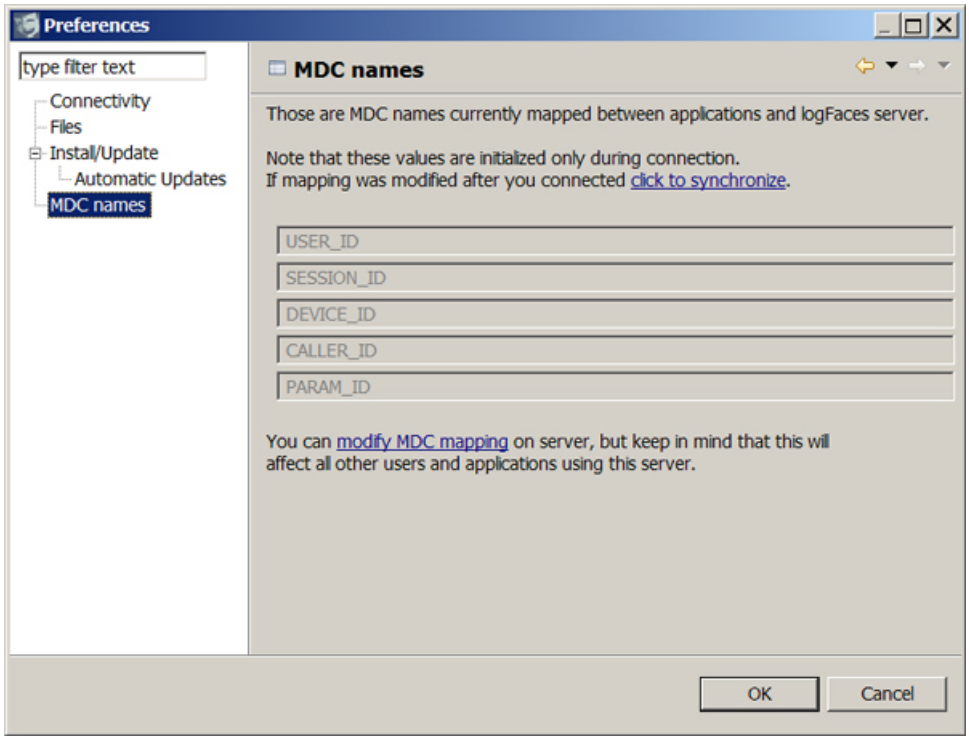


Figure 3-31: MDC names

In **Install/Update** section you specify how you would like to obtain software updates. There are several options which specify the update policy, for example, you can request to check for the updates every time the application is run and notify when they're available for installation:

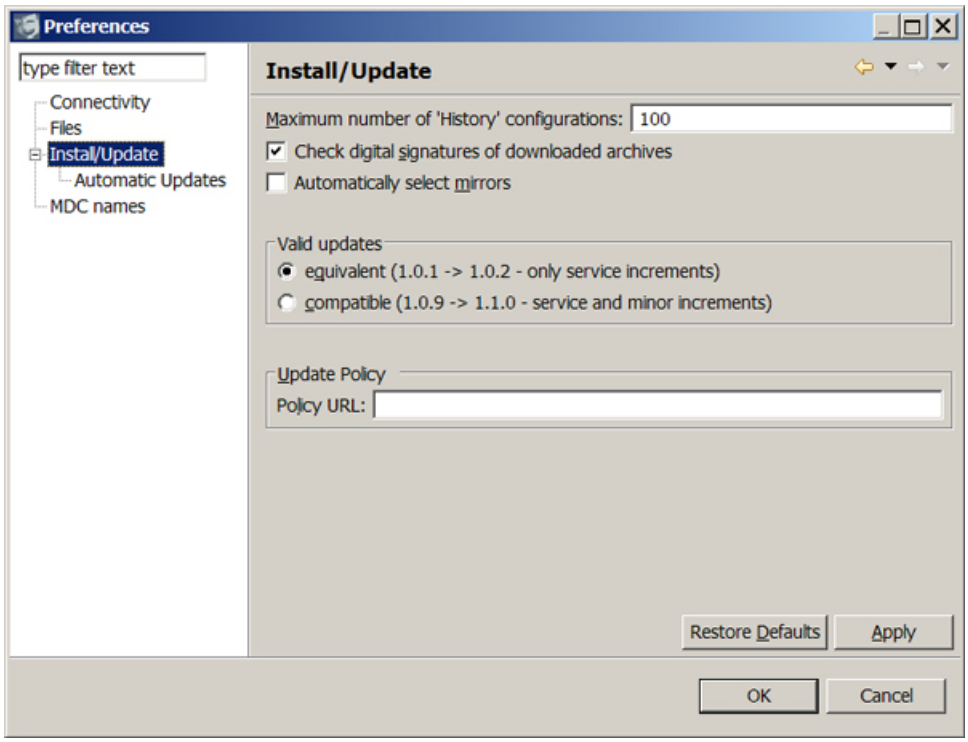


Figure 3-32 Update Preferences

You may as well disable the automatic updates and do it manually from the Help menu on the main menu bar. When new version of the software will be released, the application will display notification dialog and ask your permission to install the updates. Normally this will require consequent restart of the application, but in many cases it won't be necessary.

3.17 Status bar

Status bar displays the following information:

- current connection state
- current connection end point
- current version of logFaces server
- current license
- number of database records (click on the icon to refresh the counter)
- current RAM used on client versus maximum RAM allocated plus manual garbage collector.

Figure 3-33 Status Bar