MediaWatcher

Reference Manual

About	Description
Revision	1.6;
History	09/05/2014; Updated from 25/11/2020
Authors	Sean Kirwan

rtsoftware

Overview

Media Watcher takes sources of external data - such as RSS feeds and XLS files - and converts them to a form which tOG can use.

Media Watcher runs continuously in the background, checking for new data in the data sources, processing them according to the source type and outputting the result to the selected destination. It supports a variety of source and destination types and a single instance can handle multiple sources.

If the data source required is not covered by the list of existing source types, the applications functionality can be extended using c++ dll plugins or scriptable.

Installation

The Media Watcher installer can be downloaded from the RTSW ftp site. Running this this application setups a basic folder structure and populates it with executables, basic configuration files and a launcher for the application.

Location

Depending on your requirements, Media Watcher can be installed on either a server machine or renderers directly, or both. Important criteria such as network speeds for database write and retrieval play critical roles in determining the optimum location for the physical Media Watcher installation.

Folder Structure

The configuration files are set up through the interface. On Windows, they are stored at *C:\Users\tog\Desktop\RTSW\conf* by default - but this is selectable.

The following sub-folders are created:

- bin - contains the application executable as well as some utility executables (for example, wget.exe is used for downloading files),
- conf - contains one sample configuration file (TMW_Basic.conf),
- icons - icons used by the application launcher,
- logs - all logs are written here,
- TOGMediaWatcher/backups - some plugins will backup all files processed here,
- TOGMediaWatcher/fails - some plugins will save any file that could not be successfully imported will be copied here,
- TOGMediaWatcher/inbox - files to be imported should be copied here,
- TOGMediaWatcher/outbox - some plugins will write output files here,
- TOGMediaWatcher/plugins - dynamic plugins should be placed in this folder,
- TOGMediaWatcher/tmp -
- Uninstallers - contains the uninstaller for the application.

Application Launcher

A sample application launcher is written to the RTSW folder by the installer. It is not fully configured. Right click on the launcher and select properties. Add the following to the Target property:

```
<RTSW>\bin\TOGMediaWatcher_r<revision>.exe --conf
<RTSW>/conf/TMW_basic.conf
```

Where:

- RTSW - the installation folder (by default, C:/Users/tog/Desktop/RTSW),
- Revision - the revision number for the application (check in the bin folder for the correct name of the executable which will include the revision number).

Copy the configured launcher to the desktop. Double click on the launcher to run Media Watcher. The application is immediately minised to the system tray.

Maintaining Application Execution

Media Watcher is not a service and does not by itself maintain its over execution after it crashes or after the host system restarts.

An alternative way to launch Media Watcher is to use the batch file in the bin folder `loopMediaWatcher.bat`. This runs the application (it will be necessary to alter the batch file if the install folder has been changed from the default) and then loops. If the application fails, it is started again after a timeout period has expired. The batch file can also be copied to the Windows Startup folder so Media Watcher is started if the system is restarted.

Workflow

The tOG MediaWatcher workflow proceeds as follows:

1. Setup the details of what data to acquire and process,
2. Acquire the data and watch the specified folders for new files,
3. Process the data appropriately and save the result as directed,
4. Repeat steps 2 and 3 indefinitely.

InBoxes

Media Watcher maintains a list of inBoxes. An InBox specifies data sources and other parameters for processing the data acquired from these sources. As Media Watcher runs it checks each inBox in the order specified in the inBox list on the interface, acquires and processes data as specified by the inBox.

Plugins

The processing performed in step 3 above is done by plugins. Some are built into Media Watcher and others are loaded when the program starts from the plugins folder. All implement the following API:

```
bool    <plugin_name>_pAccept(char *fileName);
bool    <plugin_name>_pIsSerial();
bool    <plugin_name>_pImport(char *fileName,TMWInBox *inBox);
```

These methods have to have unique names - so the plugin name is included in the name

of the API methods.

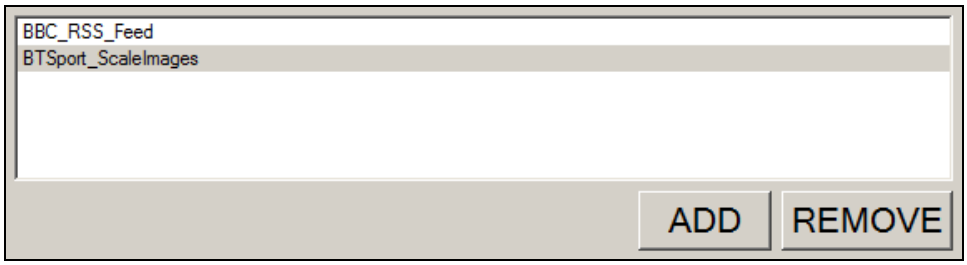
A plugin type is not explicitly matched with a particular file type. When a file is to be processed the pAccept method of each plugin is called with that file. This method returns true if the plugin can process the file. If it can then the pImport of the plugin is called on that file. pIsSerial determines whether the processing of a file can be threaded out. Only one plugin can process a file. The file is removed after the file has been accepted and processed by a plugin.

The order in which the plugins are tested with the new file is as follows:

1. Command plugin If a command is specified,
2. MOS output plugin If a MOS output is specified,
3. Non-built in plugins,
4. Customer-specific plugins (Opta, Scout7 etc.),
5. General plugins (XML, RSS, XLS etc.).

Users can of course write their own plugins - which just have to implement the extern C interface specified above. These non-builtin plugins are stored in the plugins folder.

InBox Management



An inBox specifies what data is to be acquired and what folder to watch and the parameters for the plugin selected to process the data. Any number of inBoxes can be created using the ADD button. They can be removed using the REMOVE button.

The data on the Input, Output and Parameters tabs is specific to each inBox.

InBox Data Acquisition

This proceeds as follows:

1. If there is a feed specified make a http request using the specified url and save the downloaded data to a file in the specified watch folder. *NB. the application wget.exe in the bin folder is used to do the download.*
2. If there is a serial port specified read data from the port between STX and ETX characters and save the data to a file in the specified watch folder (with a .dat extension).
3. If there is a webserver specified then Media Watcher runs a simple webserver at the port supplied through the interface. External applications can post messages to this web server. These messages are saved to the watch folder with name of the form `httpMessage_<messageId>.xml` - where messageId is an internally generated serial number.

InBox Data Processing and Output

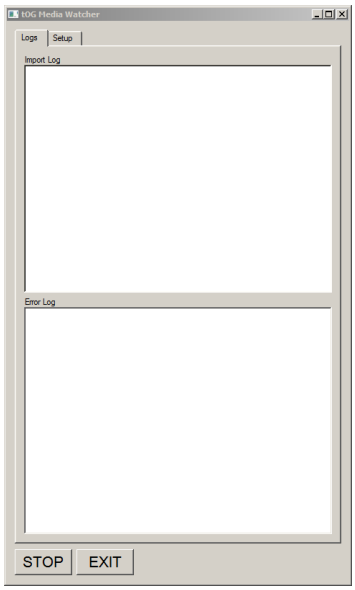
1. Media Watcher checks the watch folder periodically (every second).
2. A file is copied to the folder with an extension which is not recognized by Media Watcher and then once it is copied over completely it is renamed to a form which Media Watcher will recognize.
3. When tOG MediaWatcher finds a file it creates a temporary file with a .size extension which contains just the current size of the file.
4. Periodically the size of the file is checked and if it's the same as the size stored in the .size file, it is considered to be complete. The file is then processed and deleted.
5. If there is a file present it is backed up the the backups folder (if it exists) and moved to the tmp folder (this MUST exist).
6. It is then presented to all the registered plugins and if any plugin accepts it then that plugin is used to process the file.
7. If no plugin accepts it then it is copied to the fails folder (if it exists).
8. It is then deleted from the tmp folder.

Combining InBoxes

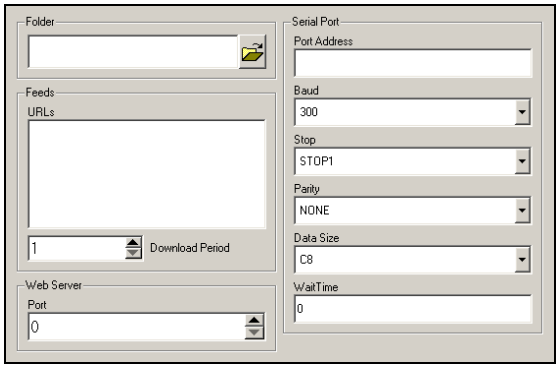
It is possible to combine inBoxes. For instance, a tOG renderer could be controlled usings messages on a serial port as follows:

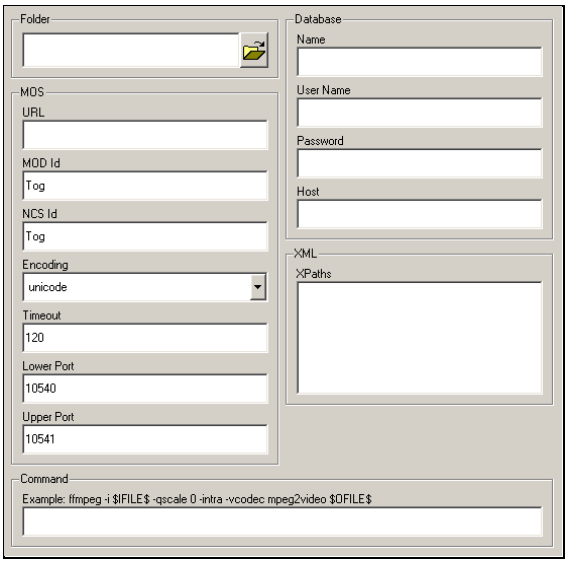
1. an inBox with a serial port and out folder specified will save messages written on the serial port (between STX and ETX) to files in the out folder.
2. an inBox whose watch folder is the output folder of the last inBox will read that file. A command specified on this inBox could extract data from the file and write a file (with a .mos extension) to a different output folder.
3. an inBox whose watch folder is the output folder of the last inBox will read that file and with the details of a MOS output specified on it will send the command to the tOG Renderer.

InBox Interface Details



Every successfully processed file is listed on the **Import Log** list and any errors on the **Error Log** list. This information is also written to the log file in the log folder.





Folder

(input) - this is the watch folder. It is the default but not necessary behaviour to set it to RTSW/TOGMediaWatcher/inBox.

Feeds URLs

this is a list of URLs. Periodically a http request is made to these urls and the resulting file is saved to the Folder.

Feeds Download Period

the period in seconds between downloading from urls.

Web Server Port

a simple web server is run at this port which accepts POST requests.

Serial Port Parameters

the parameters of the serial port from which to read data. The user must specify port address, baud rate, stop, parity, data size and wait time.

Folder

(output) - this is the output folder. It is the default but not necessary behaviour to set it to

RTSW/TOGMediaWatcher/outBox.

MOS

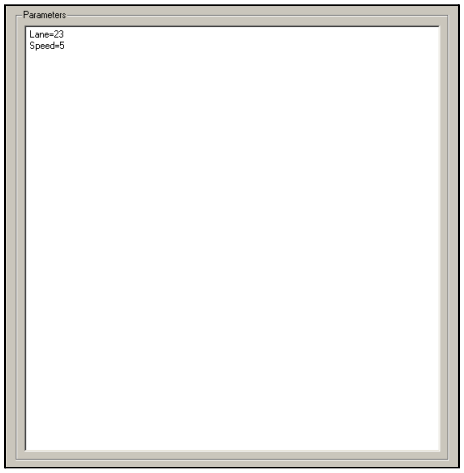
these parameters should match the MOS protocol parameters on the tOG Live instance running at url.

Database

the user can specify the name of the database, the user name and password with which to login and the url of the machine hosting the database.

XML XPaths

used by the general XML plugin to extract data from the file. One xpath should be listed per line.



Parameters

the user can enter name/value pairs into the text area, one per line in the form name=value. These are accessible by any plugin and is a means to pass configuration information to the plugin.

Plugins

There are many types of plugins. Some are general and handle well known file types. Some handle formats that are specific to certain data providers (for example, Opta provide live and seasonal sports data).

General

RSS feed files

This plugin looks for a .xml extension and then inside for standard RSS tags. It parses the file and inserts the header and item data into a database.

XML files

This plugin looks for a .xml extension. It parses the file and inserts the data into a database. It uses the xpaths specified on the interface. This are not simple xpaths. Each one maps onto a column in a table. It specifies the table name (TNAME), column name (CNAME) , column type (CTYPE), column modifiers (CMODIFIERS) and the actual xpaths (CXPATH). The following is an example:

```
TNAME=party_results;;CNAME=region_id;;CTYPE=varchar(65);;CMODIFIERS=PRIMARY
KEY;;CXPATH=foreignkey/region_results/id
```

All the xpath descriptions are used to create new tables in the database (if they don't already exist) with the specified columns. These are then populated using the xpaths.

XLS files

The plugin looks for a .xls extension (spreadsheet). Each sheet is written to the database as a separate table. The table column names and types come directly from the spreadsheet.

CSV files

The plugin looks for a .csv extension. All the data goes into a single table in the database and the name of the file is used as the table name. The first line should contain the action to perform with the data (INSERT/DELETE/UPDATE), the second line specifies column names and the third specifies column types. The rest of the lines should contain the data.

Stack files

The plugin looks for a .stk extension. The file is moved to the out folder of the InBox e.g. the Stacks folder in a project.

Image files

The plugin looks for a recognised image or movie extension. The file is moved without alteration to the out folder of the InBox e.g. the Images folder in a project.

Zip files

The plugin looks for a .zip extension. The file is unzipped first and is expected to contain standard tOG asset files (graphic, material, sound files etc). These are then copied to the appropriate folder (e.g. into a project) - the output folder specified in the inBox.

JSON files

The plugin looks for a .json extension.

General files using command

A command must be specified. If it is blank then this plugin does not accept any files. The specified command must contains these two substrings:

- $IFILE$ - this is the name of the input file (in the tmp folder)
- $OFILE$ - this is the name of the output file (in the output folder).

This allows for a general conversion of any file. Previously ImageMagick has been used to scale image files and ffmpeg to convert movie formats.

General files to MOS

The details of the MOS protocol must be specified. MOS protocol is used to control tOG renderers over a network. Refer to the tOG Live manual for details of the MOS protocol. The file is assumed to contain an XML packet that conforms to the MOS protocol and is send directly to the tOG renderer without alteration.

Sports

Opta, Scout7 and **STATS** are companies that supply sports information. tOG Sports and tOG Analyst use this data. The following plugins can store this disparate data in a single database - the schema covers .

Opta feed files

This plugin reads the F9,F24, F27, F30 and F42 feeds. It looks for a .xml extension and a specific format for the file name. The file is parsed and the data inserted into a database. These feeds contains data for teams and players for individual matches and across whole seasons. Opta uploads files continuously via ftp to the watch folder.

Scout7 feed files

This plugin makes a series of http requests using the Scout7 restful API to retrieve a full set of Scout7 data. This covers a wide range of actual player, team and competition data for a wide variety of leagues as well as specific scouting information for a Scout7 login. All this information is stored in a database.

STATS feed files

This plugin reads STATS datafiles. It looks for a .xml extension and a specific format for the file name. The file is parsed and the data inserted into a database.

tOG Analyst (formerly BlackBox)

This set of plugins support the operation of the tOG Analyst product. Media Watcher is run every day to retrieve and process raw football data and populate a local database. It is then run to calculate KPIs from the raw data and a third time to ingest customer specific data.

- ImportBB_CalculateMatchKPIs
- ImportBB_CalculateSeasonKPIs

These plugins convert basic match and seasonal data (captured by other plugins in a database) into KPIs using formulae specified through tOG Analyst. They act on empty files named kpi_match.xml and kpi_season.xml which are simply copied into the watch folder.

- ImportBB_SFCPlayerPerformanceCSV
- ImportBB_SFCScoutsCSV
- ImportBB_SFCVideoLibraryCSV
- ImportBB_CPCandidatesCSV
- ImportBB_CPRiskAssessmentCSV

These are customer specific plugins to ingest data that only that customer uses. They read data from spreadsheets which are used internally by the customer.

Customer-specific

RT Software can be contracted to implement plugins to meet unique customer needs not handled by already existing plugins. The following are some examples:

ImportSW6086

The ingests data from a parliamentary system which posts information to the Media Watcher web server about the attendance and voting of parliamentary members. This was implemented for the Scottish parliament.

ImportXIgnite*

This ingests financial from the XIgnite web server using http requests and uses it to populate a database.

ImportTRACAB

This ingests player tracking data for a match from the ChyronHego TRACAB system and stores the tracks in a database.

- ImportSwissTimingSwimming
- ImportBayernMunichClip
- ImportBBCApprovalJob
- ImportERRCandidates
- ImportERRResullts

A miscellaneous set of plugins that handle swimming results from Swiss Timing, convert movie clips for Bayern Munich FC and handle elections data for the Estonian elections.

Configuration File

This is found by default in the conf folder. It contains the details of all the inBoxes specified through the interface. With each inBox is stored a name/value pair list of all the parameters specified for the inBox.

```
<tOGMediaWatcher>
        <home>C:\Users\tog\Desktop\RTSW</home>
        <inBox>
                <name>OptaStatistics</name>
                <inputDirectory>RTSW\TOGMediaWatcher\inbox</inputDirectory>
                <outputDirectory>RTSW\TOGMediaWatcher\outbox</outputDirectory>
                <databaseHost>localhost</databaseHost>
                <databaseUserName>root</databaseUserName>
                <databasePassword></databasePassword>
                <databaseName>togsports_db</databaseName>
                <downloadPeriod>10</downloadPeriod>
                <baud>4</baud>
                <stop>1</stop>
                <parity>0</parity>
                <dataSize>3</dataSize>
                <portAddress></portAddress>
                <waitTime>120</waitTime>
                <mosId>Tog</mosId>
                <ncsId>Tog</ncsId>
                <encoding>unicode</encoding>
                <timeout>120</timeout>
                <lowerPort>10540</lowerPort>
                <upperPort>10541</upperPort>
                <host>localhost</host>
                <command></command>
                <webserverPort>0</webserverPort>
        </inBox>
</tOGMediaWatcher>
```

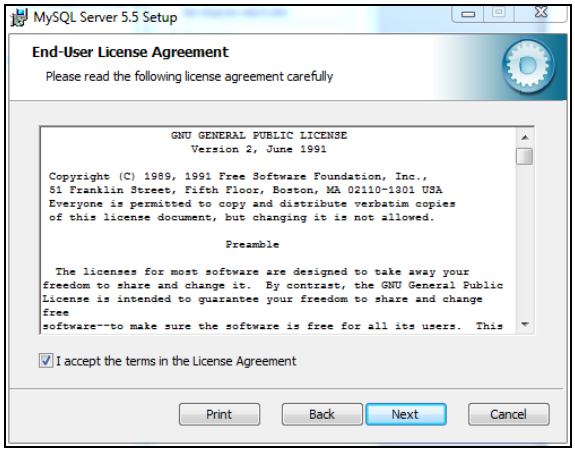
Appendix A

Windows Install for Mysql

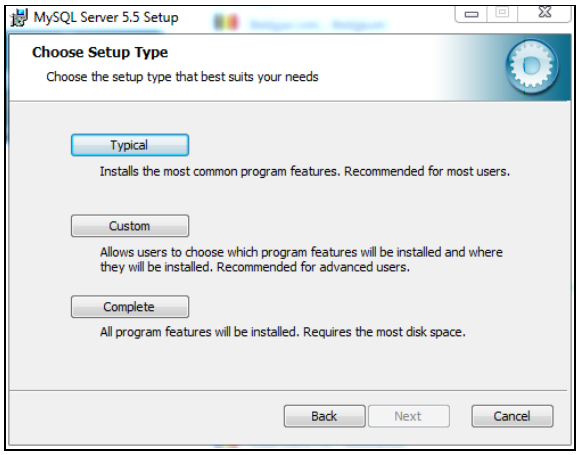
Locate the correct version of MySQL: 5.5.35 or 5.1. This can be downloaded from the RTSW ftp site onto the machine desktop.

- Double click on the installer which opens the install application.

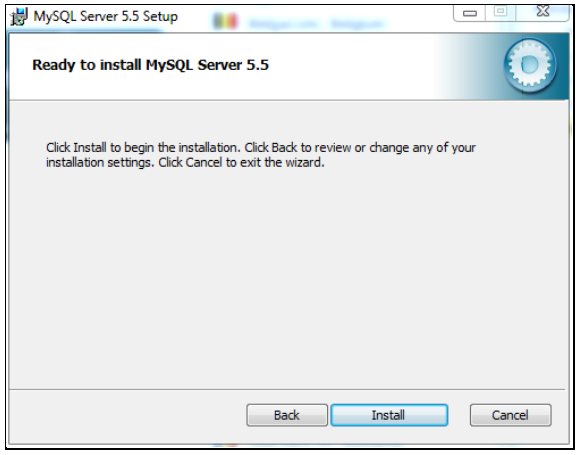
- Click Next.



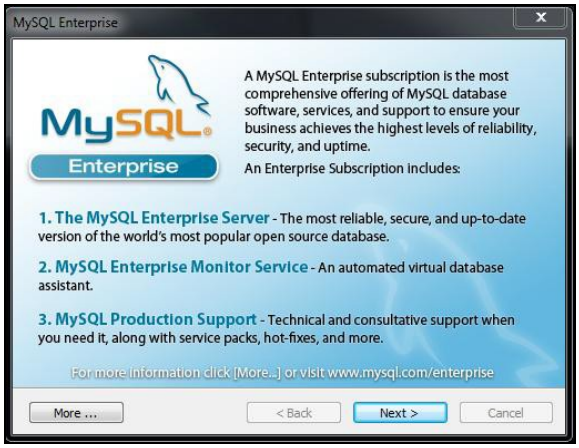
- Click Accept then Next.



- Click Typical.

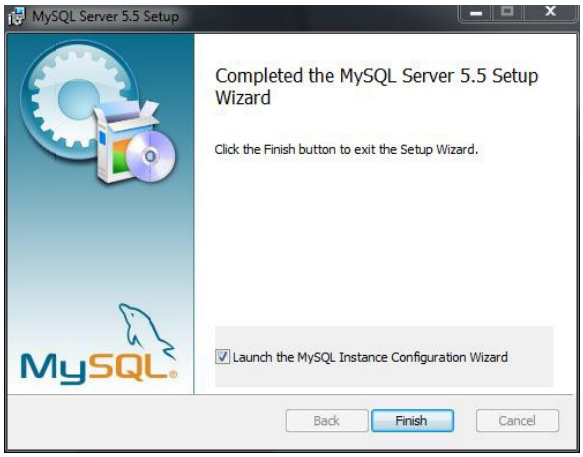


- Click Install. Mysql will now install.

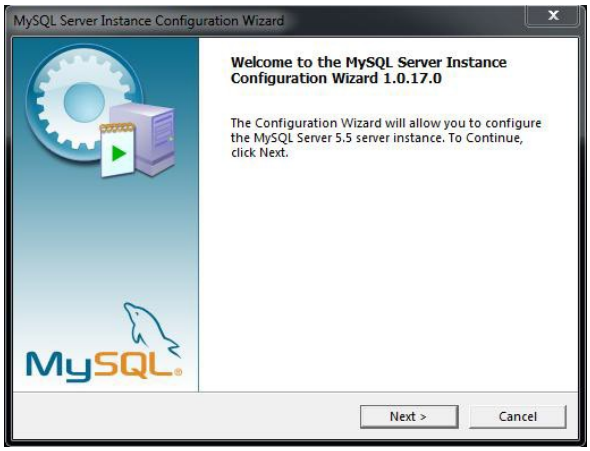


- Click Next.

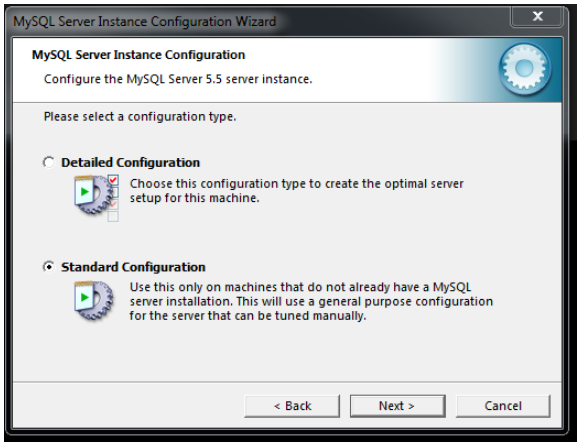
- Click Next.



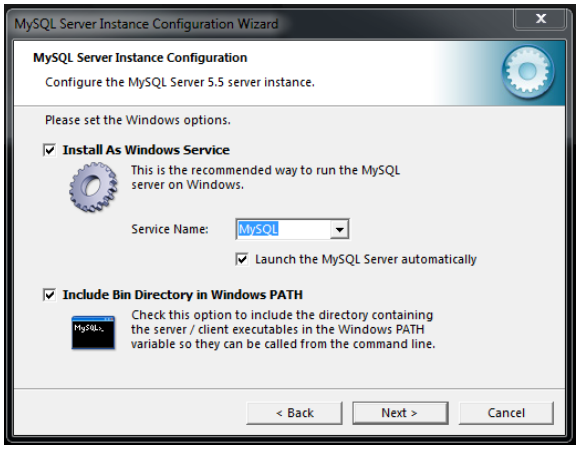
- Mysql will now launch the Mysql Instance Configuration.



- Click Next.



- Select Standard Configuration and then Next.

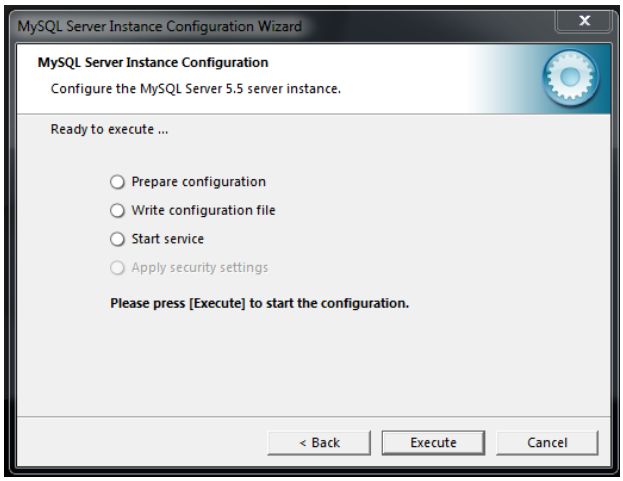


IMPORTANT NOTE: : Ensure you select "***Include Bin Directory in Windows PATH***" option

- Click Next

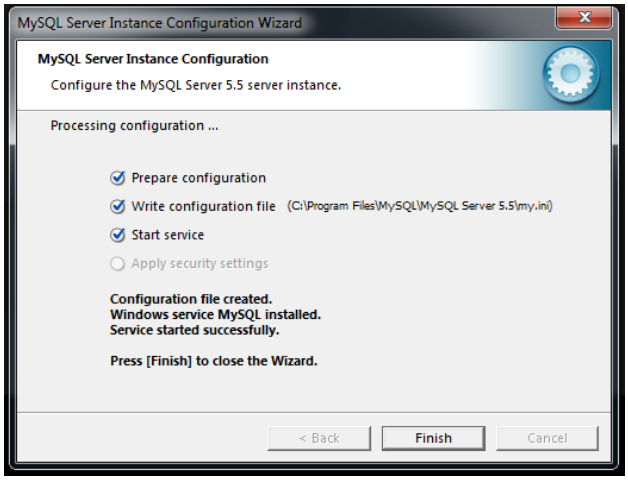
IMPORTANT NOTE : Deselect the *"Modify Security Settings"*

- Click Next - this remove the need for a password when logging into the database.



- Now click Execute.

The Mysql service will now be started and install should complete.



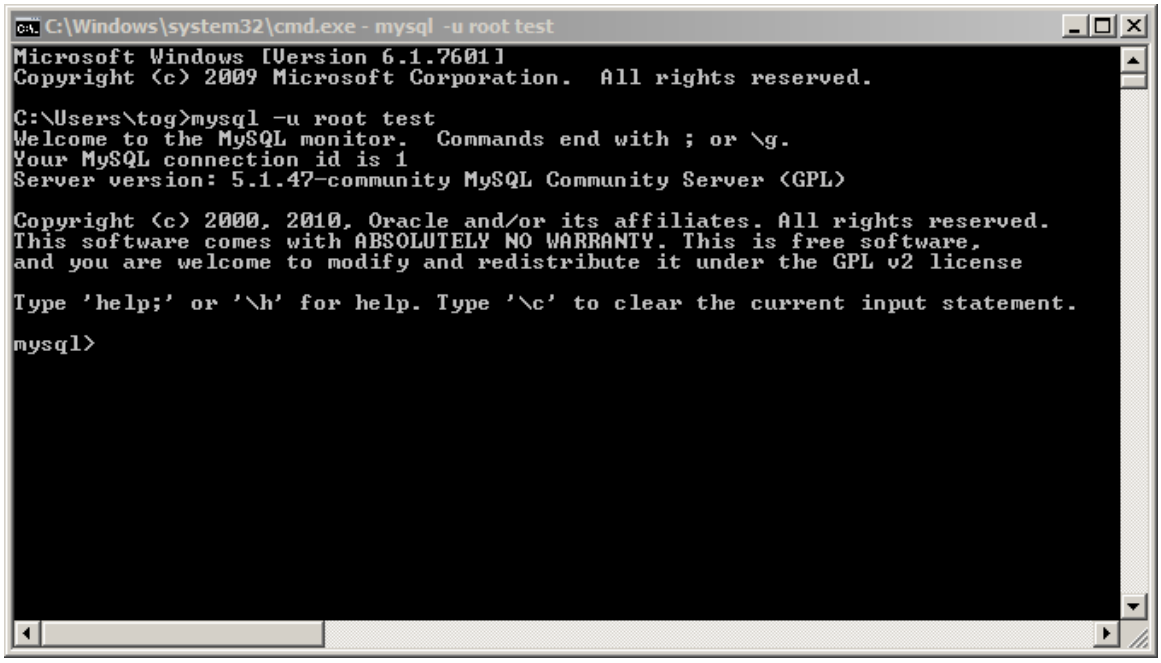
- Click Finish to complete the Mysql install.

Once MySQL has been installed, we have the framework ready to create specific databases into which our data can be written to and retrieved from.

Creating a database

Now MySQL is installed a new database is required into which data can be stored and retrieved.

- Click the Windows Start button.
- Enter into the search string: `cmd`



- Enter: `mysql -u root`
- Press `<return>` on the keyboard.
- Enter: `create database xxxx;` where xxxx is the name of your database.
- Press `<return>` on the keyboard.

IMPORTANT: ***Do not forget to input the semi-colon.***

You should see the line:

```
Query OK, 1 row affected <0.01 sec>
```

- Enter: `exit`
- Close the command window.