

SqlMyTunes for MC Instructions
A Solution for Making the Most of your Media Database

SqlMyTunes for MC is an application which exports your media library to a chosen SQL server and allows you to write SELECT statements that create playlists and smartlists, as well as perform updates on your MC library. If you have more than one MC library, all can be exported, and subsequently utilized. That is, you are able to create playlists and smartlists or perform updates based on entries in other libraries, if you wish to. Currently, SqlMyTunes supports Microsoft SQL Server and SQL Express.

Installation

Just unzip the contents of the file wherever you like and run it!

Quick Startup

Install it and explore the options! I recommend reading the instructions at least a little, but hovering the mouse pointer over an option will give you a quick run-down of that option. By default on the first run nothing is selected, and SqlMyTunes will do nothing until you press the Start button. The default options allow you to export your library and experiment with a few SQL queries.

Using SqlMyTunes

Selecting a library

Select a library you wish to configure for export. Checking *Export this library* will include the currently selected library in the export.

Setting SQL configuration

Click the *SQL* button to set SQL configuration. Each library has a separate SQL configuration. It is recommended for simplicity that you create a different database for each library, though you may export all libraries to the same database if you wish – this, however, requires some more advanced setting-up in the *Actions* list. For more details see the section headed **Multiple Libraries into a Single Database**.

Selecting Fields

Select the fields you wish to include in the export – right-click the field to set individual options for that field, i.e. the field type and whether to allow NULL values. The more fields you select, the longer the export will take.

Selecting Playlists

Check the playlists you wish to export to SQL. The more playlists you export, the longer the export will take. Checking *Export all library entries* will export all entries, regardless of whether an entry is in the checked playlists. Unchecking *Export all library entries* will only export files in the checked playlists.

Actions

Actions are stored procedures that you write yourself in SQL, inside the database you have attributed to the library. These are executed according to their type;

Setup
Expected input parameters:
- `None`

Expected returns:
- `Nothing`

A stored procedure of this type performs an initial setting up of the database, creation of Tables, etc and is called automatically by SqlMyTunes on clicking *Start*. There is a default stored procedure created automatically on clicking *Start* named **DefaultSetup**. You may choose to execute this or write your own and execute that instead. You may specify as many or as few stored procedures of this type as you like.

Insert file
Expected input parameters:
- `@libraryName NVARCHAR(255)`
  The name of the current library
- `@fileKey INT`
  The MC File Key value
- *`Dependent on fields checked for export`*
  The value of each checked field

Expected returns:
- `Nothing`

A stored procedure of this type inserts a library file entry into the database tables and is called automatically by SqlMyTunes when iterating through your MC library. There is a default stored procedure created automatically on clicking *Start* named **DefaultInsertFile**. You may choose to execute this or write your own and execute that instead. You may specify as many or as few stored procedures of this type as you like.

Insert playlist
Expected input parameters:
- `@libraryName NVARCHAR(255)`
  The name of the current library
- `@playlistID INT`
  The MC Playlist ID

- `@name NVARCHAR(max)`
  The name of the playlist
- `@path NVARCHAR(max)`
  The playlist path
- `@smart BIT`
  Whether smartlist or playlist
- `@expression NVARCHAR(max)`
  The expression (if a smartlist)
- `@notes NVARCHAR(max)`
  Any notes associated with the playlist

Returns:

- `Nothing`

A stored procedure of this type inserts a library playlist entry into the database tables and is called automatically by SqlMyTunes when iterating through your MC library. There is a default stored procedure created automatically on clicking *Start* named **DefaultInsertPlaylist**. You may choose to execute this or write your own and execute that instead. You may specify as many or as few stored procedures of this type as you like.

Insert playlist file

Expected parameters:

- `@libraryName NVARCHAR(255)`
  The name of the current library
- `@playlistID INT`
  The MC Playlist ID
- `@fileKey INT`
  The MC File Key
- `@sequence INT`
  The file's numbered sequence in the playlist

Returns:

`Nothing`

A stored procedure of this type inserts a library playlist file entry into the database tables and is called automatically by SqlMyTunes when iterating through your MC library. There is a default stored procedure created automatically on clicking *Start* named **DefaultInsertPlaylistFile**. You may choose to execute this or write your own and execute that instead. You may specify as many or as few stored procedures of this type as you like.

Finalize

Expected input parameters:

- `None`

Expected returns:

- `Nothing`

A stored procedure of this type performs a finalizing of the database and is called automatically by SqlMyTunes when it has finished iterating through your MC library. There is a default stored procedure created automatically during export named **DefaultFinalize**. You may choose to execute this or write your own and execute that

instead. You may specify as many or as few stored procedures of this type as you like.

Updates

Expected parameters:

- `@libraryName NVARCHAR(255)`
  The name of the current library

Returns:

One or many query datasets, each containing the following columns;

- `FileKey INT`
  The MC File Key of the file to be updated
- `Other field name(s)`
  Each of these field names will be updated with its associated value

A stored procedure of this type performs an update of your MC library according to the values in the returned data set, and is executed **once all selected libraries have been exported successfully**. For each row in the data set the file in your MC library with the MC File Key is retrieved and then updated with any further column/value pairs passed in the dataset. You may specify as many or as few stored procedures of this type as you like. Obviously you must take care to **only include field names in the returned data set that you wish to be updated in MC** to avoid altering your MC library spuriously or unnecessarily.

Playlists

Expected parameters:

- `@libraryName NVARCHAR(255)`
  The name of the current library

Returns:

One or many query datasets, each containing the following columns;

To create/maintain smartlists;

- `PlaylistPath NVARCHAR`
  The playlist path, use backslash to separate playlist groups e.g. This\Is\A\Test will create four playlist groups; This, Is, A, Test
- `PlaylistName NVARCHAR`
  The playlist name
- `Expression NVARCHAR`
  The smartlist expression to assign to the playlist
- `Notes NVARCHAR`
  Any notes you want to be assigned to the playlist

Or to create/maintain playlists;

- `PlaylistPath NVARCHAR`
  The playlist path, use backslash "\" to separate playlist groups, much like a Windows folder structure e.g. This\Is\A\Test will create four playlist groups; This, Is, A, Test
- `PlaylistName NVARCHAR`
  The playlist name
- `FileKey INT`
  The MC File Key of the file to add to the playlist

- `Notes NVARCHAR`
  Any notes you want to be assigned to the playlist

Or to simply delete a playlist or smartlist;

- `PlaylistPath NVARCHAR`
  The playlist path, use backslash "\" to separate playlist groups, much like a Windows folder structure
- `PlaylistName NVARCHAR`
  The playlist name
- `Delete BIT`
  Should be **1** if the playlist or smartlist is to be deleted

A stored procedure of this type creates (or edits) a playlist or smartlist in your MC library, and is executed **once all selected libraries have been exported successfully and any *Updates to library* actions have finished**. Playlists are initially cleared before they are added to. You may specify as many or as few stored procedures of this type as you like. In order to create smartlists you should make sure the settings are correct in the **Settings** form (select the Settings menu from the main screen). The name and folder of the stock smartlist to use in creation of smartlists should match those that are created by MC. The default setting is the English language setting (i.e. folder "*Smartlists*" and name "*4 or 5 Stars*").
Explanation: MC does not allow the creation of smartlists automatically, so to "trick" it, SqlMyTunes tells MC to create the stock smartlists and then edits the resulting smartlist that is named in your SqlMyTunes settings.

Multiple Libraries into a Single Database

It is recommended you use a single database for a single library for simplicity. However, if you want to put all your libraries into a single database you may do so. There are two ways to do this;

- Remove the default actions and write your own *Setup database*, *Insert* and *Finalize database* stored procedures – note that removing these actions means they will still be **created** but not **executed**.

Or a simpler way;

- Remove the default *Setup database* actions from **all but the first** library settings (using the order they appear in the dropdown list)
- You **must** have exactly the same field selections and settings in each of your library settings so the default *Insert* actions will execute successfully
- Remove the default *Finalize database* action from **all** of your library settings
- Make use of the passed `@libraryName` argument in all of your *Updates to library* and *Playlist management* actions

Command Line Arguments

The following command line arguments can be used:

- **/StartExport** - starts export immediately on startup
- **/ExitAfterExport** - exits once export is complete
- **/RunMinimized** - runs minimized
- **/ExecAfterExport** – executes the program and arguments specified in the .settings file after export
- **/Settings** - use an alternative .settings file, if not specified uses the default *SqlMyTunesMC*.

Argument usage:

SqlMyTunesMC.exe [/StartExport] [/ExitAfterExport] [/RunMinimized] [/ExecAfterExport] [/Settings:"*xxx*"]

Where *xxx* is the name of the settings file.

And Finally...

SqlMyTunes for MC works with MC12, MC13 & MC14.