

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

SqlMyTunes for iTunes is an application which exports your media library to a chosen SQL server and allows you to write SELECT statements that create playlists as well as perform updates on your iTunes library. 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 source

Select a source you wish to configure for export. “Library” is your main library, your iPods/iPhone will also be available here if you plug them in. Checking *Export this source* will include the currently selected source in the export.

Setting SQL configuration

Click the *SQL* button to set SQL configuration. Each source has a separate SQL configuration. It is recommended for simplicity that you create a different database for each source, though you may export all sources 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 Sources into a Single Database**.

Selecting Fields

Select the fields you wish to include in the export. 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 source 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:
- `@sourceName NVARCHAR(255)`
  The name of the current source
- `@fileID NVARCHAR(16)`
  The iTunes File ID 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 iTunes 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:
- `@sourceName NVARCHAR(255)`
  The name of the current source
- `@playlistID NVARCHAR(16)`
  The iTunes Playlist ID
- `@name NVARCHAR(max)`
  The iTunes Playlist ID
- `@path NVARCHAR(max)`
  The path of the playlist
- `@smart BIT`
  Whether the playlist is a smart playlist

- `@shared BIT`
  Whether the playlist is shared
- `@duration INT`
  The duration of the playlist
- `@kind NVARCHAR(255),`
  The playlist kind
- `@specialKind NVARCHAR(255),`
  The playlist "special" kind
- `@time NVARCHAR(255),`
  The total time of the playlist
- `@visible BIT,`
  Whether the playlist is visible
- `@size FLOAT(53)@name NVARCHAR(max)`
  The size of 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 iTunes 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:

- `@sourceName NVARCHAR(255)`
  The name of the current library
- `@playlistID NVARCHAR(16)`
  The iTunes Playlist ID
- `@fileID NVARCHAR(16)`
  The iTunes File ID
- `@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 iTunes 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 iTunes 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:

- `@sourceName NVARCHAR(255)`
  The name of the current source

Returns:

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

- `FileID NVARCHAR(16)`
  The iTunes File ID of the file to be updated
- `Other field name(s)`
  Each of these field names will be updated with its associated value, as they are displayed in the Fields List in the main screen

A stored procedure of this type performs an update of your iTunes library according to the values in the returned data set, and is executed **once all selected sources have been exported successfully**. For each row in the data set the file in your iTunes library with the iTunes File ID 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 iTunes** to avoid altering your iTunes library spuriously or unnecessarily. The following iTunes fields are available for editing – if others are specified they are ignored;

Album	Lyrics[2]
Album Artist	Name
Album Rating	Part Of Gapless Album
Artist	Played Count
Artwork[1,2,4]	Played Date
Bookmark Time	Rating
BPM	Remember Bookmark
Category	Season Number
Comment	Show
Compilation	Skipped Count
Composer	Skipped Date
Description	Sort Album
Disc Count	Sort Album Artist
Disc Number	Sort Artist
Enabled	Sort Composer
Episode ID	Sort Name
Episode Number	Sort Show
EQ	Start
Exclude From Shuffle	Track Count
Finish	Track Number
Genre	Unplayed
Grouping	Video Kind
Location[3]	Volume Adjustment
Long Description	Year

[1] If Artwork is specified in an update, it is assumed the string supplied is a valid filename you wish to apply as artwork to the file

[2] These fields can be slow to retrieve, and will therefore slow the export process down

[3] If Location is specified in an update, the file is moved to that location and iTunes is pointed to that location. Any empty folders left over as a result of the move are deleted.

[4] If column `Move Artwork` is also specified, `BIT` value 1, the artwork is moved to the same folder as the file that is being set

Or to delete a file from your library;

- `FileID NVARCHAR(16)`
  The iTunes File ID of the file to be deleted
- `Delete BIT`
  Should be **1** if the file is to be deleted from the library

Playlists

Expected parameters:

- `@sourceName NVARCHAR(255)`
  The name of the current source

Returns:

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

To create/maintain playlists and playlist folders;

- `PlaylistPath NVARCHAR(max)`
  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. If you wish a playlist or playlist folder name to have a backslash in it, use two backslashes to designate this i.e. "\\"
- `PlaylistName NVARCHAR(max)`
  The playlist name
- `FileKey INT`
  The iTunes File Key of the file to add to the playlist
- OPTIONAL: `Shuffle BIT`
  Should be **1** if the playlist is to be shuffled
- OPTIONAL: `Rebuild BIT`
  Should be **1** if the playlist is to be rebuilt, i.e. all contents cleared and rebuilt

Or to simply delete a playlist, smart playlist or playlist folder;

- `PlaylistID NVARCHAR(16)`
  The playlist's ID as held in the `Playlists` table
- `Delete BIT`
  Should be **1** if the playlist or smart playlist is to be deleted

Or to simply shuffle a playlist;

- `PlaylistPath NVARCHAR(max)`
  The playlist path, which is a similar to a Windows folder structure. If a playlist, smart playlist or playlist folder name has a backslash in it, use two backslashes to designate this i.e. "\\"
- `PlaylistName NVARCHAR(max)`
  The playlist name
- `Shuffle BIT`
  Should be **1** if the playlist is to be shuffled

A stored procedure of this type creates (or edits) a playlist with the specified files in the selected iTunes source, and is executed **once all selected source entries have been exported successfully and any *Updates to source* actions have finished**. Playlists are initially cleared before they are added to if `Rebuild` is specified as *true*. You may specify as many or as few stored procedures of this type as you like.

Multiple Sources into a Single Database

It is recommended you use a single database for a single source for simplicity. However, if you want to put all your sources 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 source settings so the default *Insert* actions will execute successfully
- Remove the default *Finalize database* action from **all** of your source settings
- Make use of the passed `@sourceName` argument in all of your *Updates to source* 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 *SqlMyTunesiTunes*.

Argument usage:

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

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

And Finally...

SqlMyTunes for iTunes works with iTunes 8 & 9.