# MAVERICKAPPLICATIONS

www.maverick-apps.de

Altis Life – Modular Talent Tree – Welcome

Welcome to the installation of “Altis Life Talent Tree Modular”. Please follow the tutorial closely to avoid errors. Further below you can find the full documentation for this product.  
This manual requires the “Altis Life Maverick Framework” to be already installed.

Altis Life – Modular Talent Tree – Installation

Step 1  
Open your Altis Life missionfile. Your missionfile folder will be represented as <missionfile> in this manual. Your “life\_server” folder will be represented as <server>.

Step 2  
Copy the enclosed folder “Scripts\talent-tree-modular” into “<missionfile>\maverick\”. The folder structure should now be “<missionfile>\maverick\talent-tree-modular”.

Step 3  
Open the file “<missionfile>\maverick\maverick\_functions\_master.cpp” and the following line at the very top

#include "talent-tree-modular\functions.cpp"

You can now save and close this file.

Step 4  
Open the file “<missionfile>\maverick\maverick\_master.cpp” and the following line at the very top

#include "talent-tree-modular\config.cpp"

You can now save and close this file.

Step 5  
Open the file “<missionfile>\maverick\maverick\_remoteExec\_master.cpp” and the following line at the very top

#include "talent-tree-modular\remoteExec.cpp"

You can now save and close this file.

Step 6  
Open the file “<missionfile>\core\actions\fn\_processAction.sqf” and replace the following line

\_cP = \_cP + 0.01;

with

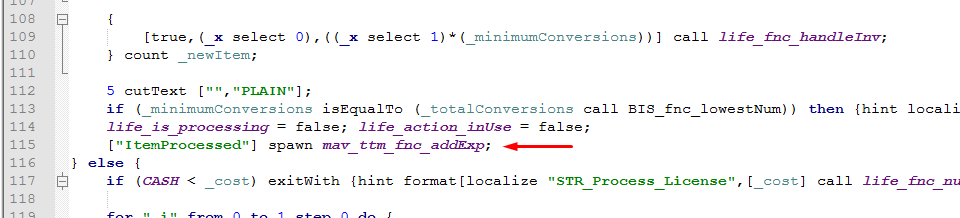
\_cP = \_cP + (0.01 \* (missionNamespace getVariable [**"mav\_ttm\_var\_processMultiplier"**, 1]));

Find the following code excerpt

**life\_is\_processing** = false; **life\_action\_inUse** = false;

and add the following code snippet below it

[**"ItemProcessed"**] spawn **mav\_ttm\_fnc\_addExp**;



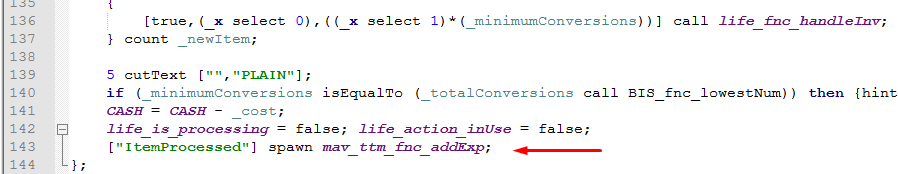
(Image might contain slightly different code)

Repeat the same for the following at the following code excerpt (near very end of the file)

[0] call **SOCK\_fnc\_updatePartial**;

**life\_is\_processing** = false;

**life\_action\_inUse** = false;



(Image might contain slightly different code)

You can now save and close this file.

Step 7  
Open the file “<missionfile>\core\items\fn\_lockpick.sqf” and replace the following line

\_cP = \_cP + 0.01;

with

\_cP = \_cP + (0.01 \* (missionNamespace getVariable [**"mav\_ttm\_var\_lockpickMultiplier"**, 1]));

At the very bottom of the file, add:

[**"VehicleLockpicked"**] spawn **mav\_ttm\_fnc\_addExp**;

You can now save and close this file.

Step 8  
Open the file “<missionfile> \core\shops\fn\_vehicleShopBuy.sqf” and above the last the following lines

closeDialog 0; //Exit the menu.

true;

add the following code snippet

[**"VehiclePurchased"**] spawn **mav\_ttm\_fnc\_addExp**;

You can now save and close this file.

Step 9  
Open the file “<missionfile>\Altis\_Life.Altis\core\init.sqf” and add the following lines

waitUntil {(missionNamespace getVariable [**"life\_perksInitialized"**, true])};

**life\_paycheck** = **life\_paycheck** \* (missionNamespace getVariable [**"mav\_ttm\_var\_paycheckMultiplier"**, 1]);

below the line

[] spawn **life\_fnc\_survival**;

You can now save and close this file.

Step 10  
Open the file “<missionfile>\description.ext” and add the following line

#include "maverick\talent-tree-modular\gui\\_masterTitles.cpp"

below the line

class RscTitles **{**

Step 11  
Execute the following statement in your database

**ALTER** **TABLE** `altislife`**.**`players`

**ADD** **COLUMN** `exp\_level` **INT** **NOT** **NULL** **DEFAULT** 0**,**

**ADD** **COLUMN** `exp\_total` **INT** **NOT** **NULL** **DEFAULT** 0 **AFTER** `exp\_level`**,**

**ADD** **COLUMN** `exp\_perkPoints` **INT** **NOT** **NULL** **DEFAULT** 0 **AFTER** `exp\_total`**,**

**ADD** **COLUMN** `exp\_perks` TEXT **AFTER** `exp\_perkPoints`**;**

Step 12

If you are using 5.0 or higher, open the files

<missionfile>\maverick\talent-tree-modular\fn\_loadFromDatabase.sqf  
 <missionfile>\maverick\talent-tree-modular\fn\_updateDatabaseEntry.sqf

and replace all instaces of **playerid** with **pid**.

The installation is now complete.

Altis Life – Modular Talent Tree – Documentation

Configuration of levels

Open the file “<missionfile>\maverick\talent-tree-modular\configuration\levels.cpp” to view all preconfigured levels.

Adding perks

To add perks, view the file “<missionfile>\maverick\talent-tree-modular\modules\maverick\_perkset\_1\perks.cpp”

To create a new perk, simply copy from a template given in the file.  
A perk is made of the following entries:

class perk\_gunsspecialist\_lessRecoil\_1 **{**

displayName **=** "Recoil Compensation"**;**

requiredPerkPoints **=** 5**;**

requiredLevel **=** 5**;**

requiredPerk **=** ""**;**

subtitle **=** "Level 5 Required, 5 Perk Points"**;**

description **=** "Learn to control weapons better and lower the noticable recoil<br/><br/><t color='#10FF45'>-5% less recoil</t>"**;**

initScript **=** "maverick\talent-tree-modular\modules\maverick\_perkset\_1\functions\functions\_recoilCompensation\_1.sqf"**;**

limitToSides**[]** **=** **{};**

color**[]** **=** **{**1**,**1**,**1**,**1**};**

**};**

The text behind the keyword “class” is the perks unique classname. Perks with the same classname will crash your server.

displayName is the text that will be displayed when viewing all available perks in the overview menu.

requiredPerkPoints is the number of perk points a player will have to spend to unlock this perk.

requiredLevel is the number of levels the player must reach before being able to purchase this perk.

requiredPerk is a string which may either be empty (the perk does not have a dependency) or hold the classname of another perk which it depends on. Dependencies will be visually represented in the overview menu through sub-levels and connecting lines.

subtitle is a string which may be anything you want. In our example perks we chose to display the number of perk points and the required level.

description is a string which holds the text the overview menu will display when viewing that perk. This string may contain structured text styling.

initScript is a string which is either empty (no initialization script) or a path to a script file in your missionfile. This script will be executed every time the perk initializes. This is either on log-on (if the perk is already owned by the player) or when the perk is purchased through the overview menu.

limitToSides[] is an array of strings which limits a perk to specific sides. The available sides are “CIV”, “WEST”, “EAST” and “INDEPENDENT”.

color[] is an array of doubles each ranging from 0 to 1 (RGBA divided by 255). This color will be picked to display the perks name in the overview menu. Leave this entry empty to display the perks name in white.

Checking if a player has a perk

The following code snippet will return a BOOLEAN whether the specified perk is owned by the player or not:

[**life\_currentExpPerks**, **"PerkClassname"**] call **mav\_ttm\_fnc\_hasPerk**;

How players gain experience points

Players can gain experience points through anything you like. In the example installation, players gain experience points by purchasing vehicles, lock picking vehicles and processing materials. You can give players experience points with this code snippet

[**"ActionName"**] spawn **mav\_ttm\_fnc\_addExp**;

What are actions?

Rather than giving the mav\_ttm\_fnc\_addExp function a specific number of experience points to add, we have decided to hard-code these values into configuration files. View the file “<missionfile>\maverick\talent-tree-modular\modules\maverick\_perkset\_1\actions.cpp” to see all existing actions. Each class in this file looks something like this

class VehiclePurchased **{**

expToAdd **=** 5**;**

message **=** "Vehicle Purchased"**;**

**};**

expToAdd is a number that describes how many experience points will be added when this action is called.

message is a text that will be displayed above the progress bar when this action is called.

Simply copy paste an existing action and adjust the values including the classes name to create your own actions.

Other variables

life\_currentPerkPoints holds the current number of available perk points.

life\_currentExpLevel holds the current level number.

life\_currentExp holds the current number of experience points.

Accessing the overview dialog

Use the following code snippet to create the perks-overview dialog

createDialog **"TTM\_GUI\_RscDisplayTalentOverview"**;

Alternatively, the dialog can also be opened through a key-combination. View the “config.cpp” to edit this combination, the default is:

*CTRL + T*