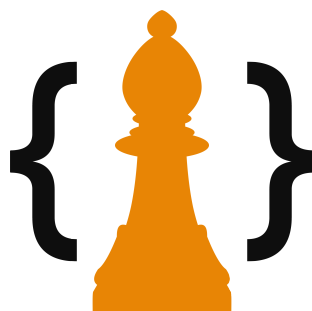


Includes for San Andreas Multiplayer Plus Version 1.8, Documentation

Ivan Kmeřo

January 2022



Contents

1	Changes in ISAMPP 1.8	4
2	Introduction	5
3	Installation	5
4	Includes	6
5	Body Parts	7
6	Bone Identifiers	7
7	Camera Modes	8
8	Color List	8
9	Crime Reports	9
10	Explosions	9
11	GameText Styles	10
12	Location Identifiers	10
13	Map Icon Identifiers	10
14	Objects	10
15	Pickup Identifiers	11
16	Pickup Types	11
17	Skin/Playermodel Identifiers	11
18	Sound Identifiers	11
19	Vehicle Components	11
20	Vehicle Identifiers	11
21	Vehicle Health	12
22	Vehicle Colors	12
23	Vehicle Paintjobs	12
24	Weapon Identifiers	12
25	Weather Identifiers	12
26	Custom Functions	13
26.1	isampp_console_printversion()	13
26.2	pawncc_console_printversion()	13
26.3	MppTeleport(playerid, locationid)	13
26.4	MppTeleportEx(playerid, locationid, pstringcolor)	13
26.5	MppTeleportToCoords(playerid, x, y, z, interiorid, facingangle)	13
26.6	MppShowPlayerPosition(playerid, pstringcolor)	14

26.7	MppShowVehicleInfo(playerid, vehicleid, pstringcolor)	14
26.8	MppGetPlayerName(playerid)	14
27	Legacy Functions	15
27.1	isnull(string)	15
27.2	rot13(string[])	15
27.3	strcpy(dest[], const source[], len = sizeof(dest))	15
27.4	strclr(string[])	15
27.5	strisempty(const string[])	15
27.6	strrest(const string[], &index)	15
27.7	strtok(const string[], &index)	15
27.8	strtolower(string[])	15
27.9	strtoupper(string[])	15
28	SA-MP Scripting Basics	16
28.1	Glossary	16
28.2	Escape Codes	16
28.3	Limits	17
29	SA-MP Macros	18
29.1	Animations List	18
29.2	Bullet Hit Types	18
29.3	Camera Cut Styles	18
29.4	Component Slots	18
29.5	Connection Status	19
29.6	Constants List	19
29.7	Dialog Styles	19
29.8	Fighting Styles	19
29.9	Keys	20
29.10	Map Icon Styles	20
29.11	Marker Modes	20
29.12	Material Text Alignments	21
29.13	Material Text Sizes	21
29.14	Object Edition Response Types	21
29.15	Player States	22
29.16	Pvar Types	22
29.17	Record Types	22
29.18	Select Object Types	22
29.19	Shop Names	23
29.20	Special Actions	23
29.21	Spectate Modes	23

1 Changes in ISAMPP 1.8

ISAMPP Library:

- New legacy functions rot13, strcpy, strchr, strisempty, strtolower, strtoupper (*i_funcl.inc*)
- New function pawncc_console_printversion() (*i_sampp.inc*)
- New definitions for sounds (*i_soundids.inc*)
- Defined 1124 new game objects (*i_objects.inc*)
- Game objects are now in separate files according to their IDE files
- Game objects now use prefix *OBJ_*
- Updated documentation

ISAMPP Sandbox Game Mode:

- Phone pickup and 2 of 3 bottle pickups now work
- New command /cancelaction for disabling special actions
- Compiler upgraded to Pawn compiler 3.10.10 (pawncc)

Changelogs for previous versions of ISAMPP are located in folder "docs/changelogs"

2 Introduction

San Andreas Multiplayer Includes Library+ or ISAMPP (*Includes for San Andreas Multiplayer Plus*) is library of include files for San Andreas Multiplayer Mod SA-MP. Purpose of ISAMPP is to make development of game modes for SA-MP much easier. SA-MP by default uses in most cases numeric identifiers which are hard to remember and without enough experience you have to use SA-MP wiki a lot. Word identifiers are much more easier to remember, they make much more sense (especially if you are familiar with modding the game Grand Theft Auto: San Andreas) and may also have positive impact on your workflow. Except re-definitions of numeric identifiers, ISAMPP also contains other useful libraries - such as library of location names with coordinates, vehicle names list, legacy scripting functions, or few related custom stock functions. ISAMPP is available with sandbox-styled game mode in which you can see everything implemented.

ISAMPP is not part of San Andreas Multiplayer mod (SA-MP) and it is not affiliated with Rockstar Games, Rockstar North or Take-Two Interactive Software Inc.

Grand Theft Auto and Grand Theft Auto: San Andreas are registered trademarks of Take-Two Interactive Software Inc.

ISAMPP versions 1.3 or newer should be considered public domain.

<https://creativecommons.org/publicdomain/zero/1.0/>

Older versions were published under MIT License.

<http://opensource.org/licenses/mit-license>

3 Installation

Copy contents of *include* folder to "[SA-MP Server folder]/include" and also to "include" folder for Pawno (by default "[SA-MP Server folder]/pawno/include").

In your game mode file you can include ISAMPP header for all its contents:

```
#include <i_sampp>
```

or include header files as you wish separately:

```
#include <i_sampp/i_bodyparts>
#include <i_sampp/i_boneids>
#include <i_sampp/i_cammode>
#include <i_sampp/i_carcols>
#include <i_sampp/i_carmods>
#include <i_sampp/i_colorlist>
#include <i_sampp/i_crimes>
#include <i_sampp/i_explosions>
#include <i_sampp/i_funcl>
#include <i_sampp/i_iconids>
#include <i_sampp/i_locationids>
#include <i_sampp/i_objects>
#include <i_sampp/i_paintjob>
#include <i_sampp/i_pickupids>
#include <i_sampp/i_pickuptypes>
#include <i_sampp/i_skinids>
#include <i_sampp/i_soundids>
#include <i_sampp/i_textstyle>
#include <i_sampp/i_vehhealth>
#include <i_sampp/i_vehids>
#include <i_sampp/i_weaponids>
#include <i_sampp/i_weatherids>
```

If you wish to run included sandbox game mode, you have to add file testmpp.pwn to your "gamemodes" folder and compile it. Alternatively, you can use pre-compiled file testmpp.amx.

4 Includes

Every ISAMPP include file starts with prefix *i*.. Please keep in mind that ISAMPP may not be the only set of include files using this prefix.

bodyparts - List of available player/npc body part identifiers
boneids - List of player bone identifiers
cammode - List of known camera modes
carcols - List of definitions for all available vehicle colors
carmods - List of all available components for vehicle customization
colorlist - List of color definitions
crimes - List of crime reports
explosions - List of available types of explosions
funcl - Legacy functions
iconids - List of map icon identifiers
locationids - List of location definitions with coordinates, interior identifiers and names
objects - List of game objects/models
paintjob - List of definitions for all available vehicle paintjobs
pickupids - List of definitions for pickup identifiers
pickuptypes - List of definitions for pickup types
skinids - List of definitions for character skin/model identifiers
soundids - List of definitions for sound identifiers
textstyle - List of definitions for GameText styles
vehhealth - Vehicle Health configurations
vehids - List of definitions for all available vehicles
weaponids - List of definitions for weapon identifiers, sorted by weapon slot IDs
weatherids - List of definitions for weather identifiers

5 Body Parts

Body part identifiers are included as macro defines with prefix *BODYPART_*. Body parts can be used in functions *OnPlayerGiveDamage*, *OnPlayerTakeDamage* and *OnPlayerGiveDamageActor*. It is unknown if IDs 0, 1 and 2 have any use. You can test this feature in sandbox game mode by shooting Actor NPC.

Identifier	Value	Body Part
BODYPART_TORSO	3	Torso
BODYPART_GROIN	4	Groin
BODYPART_LEFTARM	5	Left Arm
BODYPART_RIGHTARM	6	Right Arm
BODYPART_LEFTLEG	7	Left Leg
BODYPART_RIGHTLEG	8	Right Leg
BODYPART_HEAD	9	Head

Resources: <https://open.mp/docs/scripting/resources/bodyparts>

6 Bone Identifiers

Bone identifiers with prefix *BONE_* can be used for attaching objects to specific parts of player model/skin with function *SetPlayerAttachedObject*.

Identifier	Value	Bone
BONE.SPINE	1	Spine
BONE.HEAD	2	Head
BONE.LEFTUPPERARM	3	Left Upper Arm
BONE.RIGHTUPPERARM	4	Right Upper Arm
BONE.LEFTHAND	5	Left Hand
BONE.RIGHTHAND	6	Right Hand
BONE.LEFTTHIGH	7	Left Thigh
BONE.RIGHTTHIGH	8	Right Thigh
BONE.LEFTFOOT	9	Left Foot
BONE.RIGHTFOOT	10	Right Foot
BONE.RIGHTCALF	11	Right Calf
BONE.LEFTCALF	12	Left Calf
BONE.LEFTFOREARM	13	Left Forearm
BONE.RIGHTFOREARM	14	Right Forearm
BONE.LEFTSHOULDER	15	Left Clavicle (shoulder)
BONE.RIGHTSHOULDER	16	Right Clavicle (shoulder)
BONE.NECK	17	Neck
BONE.JAW	18	Jaw

Example of implementation:

```
public OnPlayerSpawn(playerid)
{
    //Give player green hat on spawn
    SetPlayerAttachedObject(playerid, BONE_LEFTUPPERARM, 19487, 2, 0.101,
        -0.0, 0.0, 5.50, 84.60, 83.7, 1.0, 1.0, 1.0, 0xFF00FF00);

    return 1;
}
```

Resources: <https://open.mp/docs/scripting/resources/boneid>

7 Camera Modes

List of known camera modes to be used as returned value of function *GetPlayerCameraMode(playerid)*. In ISAMPP sandbox game mode you can output more information about active camera mode with command `/cameramode`

Identifier	Value	Description
CAMMODE_TRAIN	3	Train/tram camera
CAMMODE_FOLLOWPED	4	Follow player camera
CAMMODE_SNIPER	7	Sniper rifle aiming
CAMMODE_ROCKETAIM	8	Rocket launcher aiming
CAMMODE_FIXED	15	Fixed camera (non-moving)
CAMMODE_VEHICLEFRONT	16	Vehicle front camera, bike side camera
CAMMODE_VEHICLEDEFAULT	18	Normal vehicle camera, several variable distances
CAMMODE_BOATDEFAULT	22	Normal boat camera
CAMMODE_WEAPONAIM	46	Weapon aiming camera
CAMMODE_ROCKETAIM2	51	Heat-seeking Rocket Launcher aiming camera
CAMMODE_WEAPONAIM2	53	Aiming any other weapon
CAMMODE_VEHICLEPASSENGER	55	Vehicle passenger drive-by camera
CAMMODE_HELICHASE	56	Chase camera: helicopter/bird view
CAMMODE_GROUNDCHASE	57	Chase camera: ground camera, zooms in very quickly
CAMMODE_FLYBYCHASE	58	Chase camera: horizontal flyby past vehicle
CAMMODE_GROUNDCHASE2	59	Chase camera (air vehicles): looking up
CAMMODE_FLYBYCHASE2	62	Chase camera (air vehicles): vertical flyby
CAMMODE_FLYBYCHASE3	63	Chase camera (air vehicles): horizontal flyby
CAMMODE_PILOTCHASE	64	Chase camera (air vehicles): camera focused on pilot

Resources: <https://open.mp/docs/scripting/resources/cameramodes>

8 Color List

Color List contains definitions of some few common colors in two formats. Primary or Main color definitions have prefix *COLOR_* and are defined in 0xRRGGBBAA format. Secondary or String color definitions with prefix *SCOL_* are defined in more common hex format "{RRGGBB}" without the alpha color transparency values. The difference of these two formats can be explained by this example:

```
SendClientMessage(playerid, COLOR_RED, "Hello "SCOL_BLUE"World");
```

where the output will print in game client message box string of text in this format: **Hello World**

Secondary or String colors can be used directly in strings while Primary/Main colors would be used mainly as separate parameters in SA-MP functions. To see how every color looks in game, use in included sandbox mode for ISAMPP command `/maincols [0-14]` or `/stringcols`. Screenshot of color strings printed out in client message box is located in `"docs/images/i-colorlist.png"`.

Resources: <https://open.mp/docs/scripting/resources/colorslist>

9 Crime Reports

List of crime report identifiers starting with prefix *CRIME_* followed by ten-code. These macros can be used with function *PlayCrimeReportForPlayer(playerid, suspectid, crimeid)*;

Identifier	Value	Description
CRIME_10_71	3	10-71 Advise nature of fire
CRIME_10_37_1	4	10-37 Investigate suspicious vehicle
CRIME_10_81_1	5	10-81 Breathalyzer Report
CRIME_10_24	6	10-24 Assignment Completed
CRIME_10_21_1	7	10-21 Call () by phone
CRIME_10_21_2	8	10-21 Call () by phone
CRIME_10_21_3	9	10-21 Call () by phone
CRIME_10_17	10	10-17 Meet Complainant
CRIME_10_81_2	11	10-81 Breathalyzer Report
CRIME_10_91_1	12	10-91 Pick up prisoner/subject
CRIME_10_28_1	13	10-28 Vehicle registration information
CRIME_10_81_3	14	10-81 Breathalyzer Report
CRIME_10_28_2	15	10-28 Vehicle registration information
CRIME_10_91_2	16	10-91 Pick up prisoner/subject
CRIME_10_34	17	10-34 Riot
CRIME_10_37_2	18	10-37 Investigate suspicious vehicle
CRIME_10_81_4	19	10-81 Breathalyzer Report
CRIME_10_7_1	21	10-7 Out of service
CRIME_10_7_2	22	10-7 Out of service

Example of implementation:

```
public OnPlayerCommandText(playerid, cmdtext[])
{
    if (strcmp("/crime", cmdtext, true, 20) == 0) {
        PlayCrimeReportForPlayer(playerid, playerid, CRIME_10_71);
        return 1;
    }

    return 0;
}
```

Resources: <https://open.mp/docs/scripting/resources/crimelist>

10 Explosions

There are 14 types of available explosions defined as macros with prefix *EXPLOSION_* and you can test every explosion with in-game command **/explosion [ID]**. These identifiers can be used with both functions *CreateExplosion* or *CreateExplosionForPlayer*.

Resources: <https://open.mp/docs/scripting/resources/explosionlist>

11 GameText Styles

List of definitions for SA-MP function *GameTextForPlayer(playerid, const string[], time, **style**)*. Some styles might not work properly or could crash the game.

Identifier	Value	Notes
GMTEXT_STYLE.PRICEDOWN	0	Appears for 9 seconds
GMTEXT_STYLE.RPRICEDOWN	1	Fades out after 8 seconds
GMTEXT_STYLE.SA	2	Does not disappear until player respawns
GMTEXT_STYLE.SLIM1	3	San Andreas specific font
GMTEXT_STYLE.SLIM2	4	San Andreas specific font
GMTEXT_STYLE.SLIMW	5	Displays for 3 seconds
GMTEXT_STYLE.BPRICEDOWN	6	Blue Pricedown font in middle of screen
GMTEXT_STYLE.VEHNAME	7	SA vehicle names (fixes.inc)
GMTEXT_STYLE.LOCATION	8	SA location names (fixes.inc)
GMTEXT_STYLE.RADIO	9	SA selected radio station names (fixes.inc)
GMTEXT_STYLE.RADIOW	10	SA switching radio station names (fixes.inc)
GMTEXT_STYLE.PMONEY	11	SA positive money (fixes.inc)
GMTEXT_STYLE.NMONEY	12	SA negative money (fixes.inc)
GMTEXT_STYLE.STUNT	13	SA stunt bonuses (fixes.inc)
GMTEXT_STYLE.CLOCK	14	SA in-game clock (fixes.inc)
GMTEXT_STYLE.NOTIFICATION	15	SA notification popup (fixes.inc)

fixes.inc: <https://github.com/pawn-lang/sa-mp-fixes>

Resources: <https://open.mp/docs/scripting/resources/gametextstyles>

12 Location Identifiers

Locations, in-game interiors and exteriors, are stored in file *i.locationids.inc* as set of macro definitions for identifiers and three arrays. Array *locCoords* is storing float values of XYZ position on the map and float value of PlayerFacingAngle. By default, your PlayerInterior value is set to 0 whenever you are "outside" on the map. When player enters the interior, in order to load the interior you have to change PlayerInterior value accordingly. This is what array *locID* includes. Third array, *locName*, has defined strings of names of particular locations so this information can be displayed to player if needed.

Note: If you wish to add new location to this list, you have to do it in correct order. Not every location has proper collision boxes because it was used only in cutscene, so in certain interiors you may fall through ground.

Resources: <https://open.mp/docs/scripting/resources/interiorids>

13 Map Icon Identifiers

There are in total 64 map icons included as macro defines with prefix *ICON_*.

Resources: <https://open.mp/docs/scripting/resources/mapicons>

14 Objects

Available game objects are in order by IDE file and ID number. Every game object is listed with prefix *OBJ_* followed by its name of DFF file.

Resources: https://open.mp/docs/scripting/resources/samp_objects

15 Pickup Identifiers

Pickup identifiers are included as macro defines with prefix *PICKUP_*. Every pickup defined in ISAMPP is placed on map in ISAMPP sandbox game mode. Weapon pickups use combined prefix *PICKUP_WEAP_* for easier use.

Resources: <https://open.mp/docs/scripting/resources/pickupids>

16 Pickup Types

Pickup types specify behavior of created pickups in game modes. Every pickup type available in SA-MP is listed with prefix *PICKUP_TYPE_*

Resources: <https://open.mp/docs/scripting/resources/pickuptypes>

17 Skin/Playermodel Identifiers

Skins are included as macro defines with prefix *SKIN_*, followed by name of model in game. With exception of few character models (*mainly main story characters and other special characters*) where the name of model is obvious (*f.e. emmet = Emmet, etc.*), character models are named in this or similar format: prefix (*f.e. if is located only in certain area*), race, gender, age, suffix. In this context BMYRI = **B**lack, **M**ale, **Y**oung, **R**ich. SBFYS = (**S**an **F**ransisco) **B**lack, **F**emale, **Y**oung, **S**treet, etc. This format is very helpful in contrast with plain number identifiers SA-MP uses by default.

Resources: <https://open.mp/docs/scripting/resources/skins>

18 Sound Identifiers

Sounds meant to be played with function *PlayerPlaySound* are defined with prefix *SND_* in file *i_soundids.inc*. Please note that this list of available sounds may be incomplete and some sound loops might cause the game client to crash.

Resources: <https://open.mp/docs/scripting/resources/sound-ids>

19 Vehicle Components

List of all available components for vehicle customization, starting with prefix *CARMOD_*. Can be used with function *AddVehicleComponent(vehicleid, **componentid**)*;

Resources: <https://open.mp/docs/scripting/resources/carcomponentid>

20 Vehicle Identifiers

Vehicle identifiers are included as macro defines with prefix *VEH_* followed by the name of vehicle in game with capital letters (*motorcycles, bicycles, boats, planes, helicopters, trailers, trains, etc. are included in this format as well*). For better orientation in list, defines are sorted by vehicle type.

Resources: <https://open.mp/docs/scripting/resources/vehicleid>

21 Vehicle Health

Vehicle Health configurations are included as macro defines with prefix *VEH_HEALTH_* followed by the desired pre-defined identifier. These values are only related to engine condition and do not change visual damage of vehicle model.

Identifier	Exact Value	Description
VEH_HEALTH_FULL	1000	Full vehicle health
VEH_HEALTH_FULL_LOW	650	Lowest value for undamaged vehicle
VEH_HEALTH_WHITESMOKE	649	White smoke from engine
VEH_HEALTH_WHITESMOKE_LOW	550	Lowest value for white engine smoke
VEH_HEALTH_GREYSMOKE	549	Grey smoke from engine
VEH_HEALTH_GREYSMOKE_LOW	390	Lowest value for grey engine smoke
VEH_HEALTH_BLACKSMOKE	389	Black smoke from engine
VEH_HEALTH_BLACKSMOKE_LOW	250	Lowest value for black engine smoke
VEH_HEALTH_ONFIRE	249	Sets car on fire

You can test this in ISAMPP sandbox game mode with command `/setvehiclehealth [ID]` while being in any vehicle.

Resources: <https://open.mp/docs/scripting/resources/vehiclehealth>

22 Vehicle Colors

List of all available colors for vehicles in game. Color names using prefix *CARCOL_SAMP_* are supported only in SA-MP version 0.3x.

Resources: <https://open.mp/docs/scripting/resources/vehiclecolorid>

23 Vehicle Paintjobs

List of all available paintjobs for vehicles in game, starting with prefix *PAINTJOB_* followed by vehicle name and name of paintjob. Use *PAINTJOB_REMOVE* to remove applied paintjob.

Example of implementation:

```
AddStaticVehicle(VEH_CAMPER, 425.7991, 2493.3472, 16.5794,  
                180, CARCOL_WHITE, CARCOL_WHITE);  
ChangeVehiclePaintjob(vehicleid, PAINTJOB_CAMPER_TRUTH);
```

Resources: <https://open.mp/docs/scripting/resources/paintjobs>

24 Weapon Identifiers

Weapon identifiers are included as macro defines with prefix *WEAP_*. In ISAMPP sandbox game mode weapon pickups are fully functional. Every weapon in game can be used only for certain weapon slot and weapon identifiers are ordered in file *i_weaponids.inc* by weapon slot number.

Resources: <https://open.mp/docs/scripting/resources/weaponids>

25 Weather Identifiers

Weather identifiers are included as macro defines with prefix *WEATHER_*. In ISAMPP sandbox game mode you can test weather settings with command `/w [ID]`

Resources: <https://open.mp/docs/scripting/resources/weatherid>

26 Custom Functions

ISAMPP uses various stock functions which may be useful in creating your own game modes for SA-MP or simply for debugging purposes. These stock functions are defined in *i_sampp.inc* file.

26.1 isampp_console_printversion()

Outputs ISAMPP version to server console.

Example of implementation:

```
main() {
    isampp_console_printversion();
}
```

26.2 pawncc_console_printversion()

Outputs version of compiler to server console if Pawncc is used.

Example of implementation:

```
main() {
    pawncc_console_printversion();
}
```

26.3 MppTeleport(playerid, locationid)

Teleports player to desired location passed as parameter 'locationid'.

Example of implementation:

```
if (strcmp("/tp barbershop", cmdtext, true, 20) == 0) {
    MppTeleport(playerid, LOC_BARBERSHOP);
    return 1;
}
```

26.4 MppTeleportEx(playerid, locationid, pstringcolor)

Same as MppTeleport, plus outputs location name to in-game chat window. Parameter 'pstringcolor' must be in hexadecimal format 0xRRGGBBAA.

Example of implementation:

```
if (strcmp("/tp barbershop", cmdtext, true, 20) == 0) {
    MppTeleportEx(playerid, LOC_BARBERSHOP, COLOR_LIMEGREEN);
    return 1;
}
```

26.5 MppTeleportToCoords(playerid, x, y, z, interiorid, facingangle)

Teleports player to specified xyz coordinates, supplied with interior identifier and player facing angle.

Example of implementation:

```
if (strcmp("/tpcoord", cmdtext, true, 20) == 0) {
    MppTeleportToCoords(playerid, 49.4172, 2512.4282, 16.4844, 0, 272);
    return 1;
}
```

26.6 MppShowPlayerPosition(playerid, pstringcolor)

Outputs current player location coordinates, interior identifier, facing angle and player camera position coordinates to in-game chat window.

Parameter 'pstringcolor' must be in hexadecimal format 0xRRGGBBAA.

This function might not work properly if player is in a vehicle.

Example of implementation:

```
if (strcmp("/showplayerpos", cmdtext, true, 15) == 0) {
    MppShowPlayerPosition(playerid, COLOR_LIGHTRED);
    return 1;
}
```

26.7 MppShowVehicleInfo(playerid, vehicleid, pstringcolor)

Outputs ID, model, health, position and rotation of vehicle in which is player currently sitting to in-game chat window.

Parameter 'pstringcolor' must be in hexadecimal format 0xRRGGBBAA.

Note: You must pass vehicleid parameter if you want to get model name else function returns '0 Unknown'.

Example of implementation:

```
new VehicleModelID = 0;

public OnPlayerCommandText(playerid, cmdtext[])
{
    if (strcmp("/showvehicleinfo", cmdtext, true, 15) == 0) {
        MppShowVehicleInfo(playerid, VehicleModelID, COLOR_LIGHTBLUE);
        return 1;
    }
    return 0;
}

public OnPlayerEnterVehicle(playerid, vehicleid, ispassenger)
{
    VehicleModelID = GetVehicleModel(vehicleid);
    return 1;
}

public OnPlayerExitVehicle(playerid, vehicleid)
{
    VehicleModelID = 0;
    return 1;
}
```

26.8 MppGetPlayerName(playerid)

Returns player nick/name from given playerid.

Example of implementation:

```
if (strcmp(cmdtext, "/myname", true) == 0) {
    SendClientMessage(playerid, COLOR_LIGHTBLUE, MppGetPlayerName(playerid));
    return 1;
}
```

27 Legacy Functions

Legacy functions are implemented in file *i_func1.inc* for backwards compatibility with really old SA-MP gamemodes which often use them.

27.1 `isnull(string)`

Checks whether a string is equal to null (empty). More efficient than checking if `strlen` is equal to 0.

27.2 `rot13(string[])`

Rotates the alphabet in string by half of its length - 13 characters. It is a symmetric operation: applying it twice on the same string reveals the original.

27.3 `strcpy(dest[], const source[], len = sizeof(dest))`

Copies the source string to the destination string.

27.4 `strclr(string[])`

Empties and clears given string.

27.5 `strisempty(const string[])`

Returns true if the given string is empty, otherwise returns false.

27.6 `strrest(const string[], &index)`

Splits string and gives back remaining part of the string divided by space ' ' character as default delimiter.

27.7 `strtok(const string[], &index)`

`Strtok` is used for splitting strings and was used as one of the methods for creating game commands with arguments. Strings are divided by space ' ' character as default delimiter.

27.8 `strtolower(string[])`

Changes all characters in the string to lowercase.

27.9 `strtoupper(string[])`

Changes all characters in the string to uppercase.

28 SA-MP Scripting Basics

28.1 Glossary

Word	Meaning
PAWN	The scripting language used to make SA-MP scripts
Gamemode	The main game script that runs on a server
Filterscripts	Scripts that run alongside gamemodes
Plugins	Extra functions and other features added through .DLL or .SO libraries
Include	Pieces of script in .INC files to be included in Filterscripts/Gamemodes
Pawno	The script editor most people use for PAWN programming
Pawncc	The compiler that compiles .pwn to .amx files
Masterlist	The server SA-MP stores its data on such as the Internet list

Read More: <https://open.mp/docs/scripting/resources/glossary>

28.2 Escape Codes

When creating a string you may find that some character may be impossible or extremely difficult to express in the source code of your script. This is where escape codes come in handy - these allow you to use the symbols and expressions that come under this category. Below is a list of escape codes for the PAWN language.

Code	Description
<code>\a</code> or <code>\7</code>	Audible beep (on server machine)
<code>\b</code>	Backspace
<code>\e</code>	Escape
<code>\f</code>	Form feed
<code>\n</code>	New line
<code>\r</code>	Carriage return
<code>\t</code>	Horizontal tab
<code>\v</code>	Vertical tab
<code>\h</code>	Backslash
<code>\h'</code>	Single quote
<code>\h"</code>	Double quote
<code>\%</code>	Percent sign
<code>\ddd;</code>	Character code with decimal code
<code>\xhhh;</code>	Character code with hexadecimal code

Note: The semicolon after the `nddd;` and `xhhh;` codes is optional. Its purpose is to give the escape sequence an explicit termination symbol when it is used in a string constant.

Read More: <https://open.mp/docs/scripting/resources/escapecodes>

28.3 Limits

In-game Entities

Type	Limit
Players	1000
Vehicles	2000
Vehicle Models	Unlimited
Objects	1000
Virtual Worlds	2,147,483,647
Interiors	256
Classes	320
Map Icons	100
Race Checkpoints	1
Checkpoints	1
Pickups	4096
Global 3D Labels	1024
Per-player 3D Text Labels	1024
Chat Bubble String	144
SetObjectMaterialText length	2048
SetPlayerObjectMaterialText length	2048
Gangzones	1024
Menus	128
Attached player objects	10
Player Variables	800
Actors (since SA-MP 0.3.7)	1000
Explosions	10

Textdraws

Type	Limit
String Length	1024
Shown In A Single Client's Screen	2048 + 256
Shown In A Single Client's Screen (sprites)	100
Created Serverwise (Global)	2048
Created Serverwise (Per-Player)	256

Dialogs

Type	Limit
Dialog IDs	32768
Info (Main text)	4096
Caption	64
Input Text Box	128
Tab List Columns	4
Tab List Column Characters	128
Tab List Row Characters	256

Read More: <https://open.mp/docs/scripting/resources/limits>

29 SA-MP Macros

SA-MP has in certain cases various macros replacing numeric identifiers by default.

29.1 Animations List

Read More: <https://open.mp/docs/scripting/resources/animations>

29.2 Bullet Hit Types

To be used with function *OnPlayerWeaponShot*.

Value	Name
0	BULLET_HIT_TYPE_NONE
1	BULLET_HIT_TYPE_PLAYER
2	BULLET_HIT_TYPE_VEHICLE
3	BULLET_HIT_TYPE_OBJECT
4	BULLET_HIT_TYPE_PLAYER_OBJECT

Read More: <https://open.mp/docs/scripting/resources/bullethittypes>

29.3 Camera Cut Styles

Camera Cut Styles to be used with functions *SetPlayerCameraLookAt*, *InterpolateCameraPos* and *InterpolateCameraLookAt*.

Value	Name
1	CAMERA_MOVE
2	CAMERA_CUT

Read More: <https://open.mp/docs/scripting/resources/cameracutstyles>

29.4 Component Slots

All available component slots for vehicle customization.

To be used with function *GetVehicleComponentInSlot(vehicleid, slot)*;

Value	Name
0	CARMODTYPE_SPOILER
1	CARMODTYPE_HOOD
2	CARMODTYPE_ROOF
3	CARMODTYPE_SIDESKIRT
4	CARMODTYPE_LAMPS
5	CARMODTYPE_NITRO
6	CARMODTYPE_EXHAUST
7	CARMODTYPE_WHEELS
8	CARMODTYPE_STEREO
9	CARMODTYPE_HYDRAULICS
10	CARMODTYPE_FRONT BUMPER
11	CARMODTYPE_REAR BUMPER
12	CARMODTYPE_VENT_RIGHT
13	CARMODTYPE_VENT_LEFT

Read More: <https://open.mp/docs/scripting/resources/Componentslots>

29.5 Connection Status

Connection status returned as value of function *NetStats_ConnectionStatus(playerid)*;

Value	Name	Description
0	NO_ACTION	N/A
1	DISCONNECT_ASAP	OnPlayerDisconnect called
2	DISCONNECT_ASAP_SILENTLY	N/A
3	DISCONNECT_ON_NO_ACK	N/A
4	REQUESTED_CONNECTION	Connection request cookie sent
5	HANDLING_CONNECTION_REQUEST	N/A
6	UNVERIFIED_SENDER	N/A
7	SET_ENCRYPTION_ON_MULTIPLE_16_BYTE_PACKET	N/A
8	CONNECTED	playerid is connected

Read More: <https://open.mp/docs/scripting/resources/connectionstatus>

29.6 Constants List

Pre-defined constants and identifiers in various SA-MP header files.

Read More: <https://open.mp/docs/scripting/resources/constants>

29.7 Dialog Styles

Styles of dialog windows in SA-MP, to be used with function *ShowPlayerDialog*.

Style	Name
Style 0	DIALOG_STYLE_MSGBOX
Style 1	DIALOG_STYLE_INPUT
Style 2	DIALOG_STYLE_LIST
Style 3	DIALOG_STYLE_PASSWORD
Style 4	DIALOG_STYLE_TABLIST
Style 5	DIALOG_STYLE_TABLIST_HEADERS

Read More: <https://open.mp/docs/scripting/resources/dialogstyles>

29.8 Fighting Styles

Fighting Styles to be used with functions *SetPlayerFightingStyle(playerid, **style**)* and *GetPlayerFightingStyle*.

Value	Name
4	FIGHT_STYLE_NORMAL
5	FIGHT_STYLE_BOXING
6	FIGHT_STYLE_KUNGFU
7	FIGHT_STYLE_KNEEHEAD
15	FIGHT_STYLE_GRABKICK
16	FIGHT_STYLE_ELBOW

Read More: <https://open.mp/docs/scripting/resources/fightingstyles>

29.9 Keys

To be used with functions *GetPlayerKeys* and *textitOnPlayerKeyStateChange*.

Value	Name
1	KEY_ACTION
2	KEY_CROUCH
4	KEY_FIRE
8	KEY_SPRINT
16	KEY_SECONDARY_ATTACK
32	KEY_JUMP
64	KEY_LOOK_RIGHT
128	KEY_HANDBRAKE/KEY_AIM
256	KEY_LOOK_LEFT
512	KEY_LOOK_BEHIND
512	KEY_SUBMISSION
1024	KEY_WALK
2048	KEY_ANALOG_UP
4096	KEY_ANALOG_DOWN
8192	KEY_ANALOG_LEFT
16384	KEY_ANALOG_RIGHT
65536	KEY_YES
131072	KEY_NO
262144(4)	KEY_CTRL_BACK
-	UNDEFINED
-128	KEY_UP
128	KEY_DOWN
-128	KEY_LEFT
128	KEY_RIGHT

Read More: <https://open.mp/docs/scripting/resources/keys>

29.10 Map Icon Styles

To be used with function *SetPlayerMapIcon*.

Value	Name	Marker	Radar Map Range
0	MAPICON_LOCAL	No	Close proximity only
1	MAPICON_GLOBAL	No	Show on radar edge as long as in range
2	MAPICON_LOCAL_CHECKPOINT	Yes	Close proximity only
3	MAPICON_GLOBAL_CHECKPOINT	Yes	Show on radar edge as long as in range

Read More: <https://open.mp/docs/scripting/resources/mapiconstyles>

29.11 Marker Modes

To be used with function *ShowPlayerMarkers(mode)*;

Value	Name
0	PLAYER_MARKERS_MODE_OFF
1	PLAYER_MARKERS_MODE_GLOBAL
2	PLAYER_MARKERS_MODE_STREAMED

Read More: <https://open.mp/docs/scripting/resources/markermodes>

29.12 Material Text Alignments

To be used with function *SetObjectMaterialText*.

Value	Name
0	OBJECT_MATERIAL_TEXT_ALIGN_LEFT
1	OBJECT_MATERIAL_TEXT_ALIGN_CENTER
2	OBJECT_MATERIAL_TEXT_ALIGN_RIGHT

Read More: <https://open.mp/docs/scripting/resources/materialtextalignment>

29.13 Material Text Sizes

To be used with function *SetObjectMaterialText*.

Value	Name
10	OBJECT_MATERIAL_SIZE_32x32
20	OBJECT_MATERIAL_SIZE_64x32
30	OBJECT_MATERIAL_SIZE_64x64
40	OBJECT_MATERIAL_SIZE_128x32
50	OBJECT_MATERIAL_SIZE_128x64
60	OBJECT_MATERIAL_SIZE_128x128
70	OBJECT_MATERIAL_SIZE_256x32
80	OBJECT_MATERIAL_SIZE_256x64
90	OBJECT_MATERIAL_SIZE_256x128
100	OBJECT_MATERIAL_SIZE_256x256
110	OBJECT_MATERIAL_SIZE_512x64
120	OBJECT_MATERIAL_SIZE_512x128
130	OBJECT_MATERIAL_SIZE_512x256
140	OBJECT_MATERIAL_SIZE_512x512

Read More: <https://open.mp/docs/scripting/resources/materialtextsizes>

29.14 Object Edition Response Types

To be used with functions *OnPlayerEditObject* and *OnPlayerEditAttachedObject*.

Value	Name	Description
0	EDIT_RESPONSE_CANCEL	Player cancelled (ESC)
1	EDIT_RESPONSE_FINAL	Player clicked on save
2	EDIT_RESPONSE_UPDATE	Player moved the object (edition did not stop)

Read More: <https://open.mp/docs/scripting/resources/objecteditionresponsetypes>

29.15 Player States

To be used with *GetPlayerState* function or *OnPlayerStateChange* callback.

Value	Name	Description
0	PLAYER_STATE_NONE	Default state, used while initializing
1	PLAYER_STATE_ONFOOT	Player is on foot
2	PLAYER_STATE_DRIVER	Player is driving a vehicle
3	PLAYER_STATE_PASSENGER	Player is in a vehicle as a passenger
4	PLAYER_STATE_EXIT_VEHICLE	Player is exiting vehicle
5	PLAYER_STATE_ENTER_VEHICLE_DRIVER	Entering vehicle (driver)
6	PLAYER_STATE_ENTER_VEHICLE_PASSENGER	Entering vehicle (passenger)
7	PLAYER_STATE_WASTED	Player is either dead or in class selection
8	PLAYER_STATE_SPAWNED	Player just spawned
9	PLAYER_STATE_SPECTATING	Player is in spectator mode

Read More: <https://open.mp/docs/scripting/resources/playerstates>

29.16 Pvar Types

Types of player variables (also called pvar types) used in Per-player variable system.

Value	Name
0	PLAYER_VARTYPE_NONE
1	PLAYER_VARTYPE_INT
2	PLAYER_VARTYPE_STRING
3	PLAYER_VARTYPE_FLOAT

Read More: <https://open.mp/docs/scripting/resources/pvartypes>

29.17 Record Types

To be used with *StartRecordingPlayerData* function.

Value	Name
0	PLAYER_RECORDING_TYPE_NONE
1	PLAYER_RECORDING_TYPE_DRIVER
2	PLAYER_RECORDING_TYPE_ONFOOT

Read More: <https://open.mp/docs/scripting/resources/recordtypes>

29.18 Select Object Types

Select object types used by *OnPlayerSelectObject* function.

Value	Name
1	SELECT_OBJECT_GLOBAL_OBJECT
2	SELECT_OBJECT_PLAYER_OBJECT

Read More: <https://open.mp/docs/scripting/resources/selectobjecttypes>

29.19 Shop Names

To be used with *SetPlayerShopName(playerid, shopname[])* function.

Name	Description	ISAMPP Location ID
FDPIZA	Stock Pizza Stack interior	LOC_PIZZASTACK
FDCHICK	Stock Cluckin' Bell interior	LOC_CBELL
FDBURG	Stock Burger Shot interior	LOC_BURGERSHOT
AMMUN1	First Ammu-Nation interior	LOC_AMMUNATION2
AMMUN2	Second Ammu-Nation interior	LOC_AMMUNATION3
AMMUN3	Third Ammu-Nation interior	LOC_AMMUNATION4
AMMUN4	Fourth Ammu-Nation interior	LOC_AMMUNATION
AMMUN5	Fifth Ammu-Nation interior	LOC_AMMUNATION5

Read More: <https://open.mp/docs/scripting/resources/shopnames>

29.20 Special Actions

Used by *GetPlayerSpecialAction* and *SetPlayerSpecialAction* functions.

Value	Action	Description
0	SPECIAL_ACTION_NONE	Clears player of special actions
1	SPECIAL_ACTION_DUCK	Detect if the player is crouching
2	SPECIAL_ACTION_USEJETPACK	Make the player using jetpack
3	SPECIAL_ACTION_ENTER_VEHICLE	Player is entering a vehicle
4	SPECIAL_ACTION_EXIT_VEHICLE	Player is exiting a vehicle
5	SPECIAL_ACTION_DANCE1	Applies dancing animation for player
6	SPECIAL_ACTION_DANCE2	Applies dancing animation for player
7	SPECIAL_ACTION_DANCE3	Applies dancing animation for player
8	SPECIAL_ACTION_DANCE4	Applies dancing animation for player
10	SPECIAL_ACTION_HANDSUP	Make the player put hands up
11	SPECIAL_ACTION_USECELLPHONE	Make the player speaking on cellphone
12	SPECIAL_ACTION_SITTING	Detects if the player is sitting
13	SPECIAL_ACTION_STOPUSECELLPHONE	Makes players stop using cellphone
20	SPECIAL_ACTION_DRINK_BEER	Increase the player's drunk level
21	SPECIAL_ACTION_SMOKE_CIGGY	Give the player a cigar
22	SPECIAL_ACTION_DRINK_WINE	Give the player a wine bottle
23	SPECIAL_ACTION_DRINK_SPRUNK	Give the player a sprunk bottle
24	SPECIAL_ACTION_CUFFED	Force the player in to cuffs
25	SPECIAL_ACTION_CARRY	Apply a carrying animation to the player
68	SPECIAL_ACTION_PISSING	Player performs the pissing animation

Read More: <https://open.mp/docs/scripting/resources/specialactions>

29.21 Spectate Modes

Used by *PlayerSpectatePlayer* and *PlayerSpectateVehicle* functions.

Name	Description
SPECTATE_MODE_NORMAL	Normal spectate mode - third person camera
SPECTATE_MODE_FIXED	Used with SetPlayerCameraPos function
SPECTATE_MODE_SIDE	Camera will be attached to the side of the player/vehicle

Read More: <https://open.mp/docs/scripting/resources/spectatemodes>