# VCMP 0.4 Server Creation Documentation

## Brief

This is my personal documentation of VCMP Scripting of version 0.4 in Squirrel language. It is to be a Concise Review Text of everything I have ever stumbled upon in my days of scripting and includes some reviews of other works such as Thijn’s Wiki Guide of VCMP (covering standard variables to VCMP servers), Stormeus’s documentation of all the features possible of the 0.4 revision and Sseebbyy’s VCMP 0.4 Blank Server format. This personal documentation also touches on the necessary files for server creation and how to write some of them to get your server running.

## Introduction

### Squirrel Language

Squirrel is a high level imperative, object-oriented programming language, designed to be a light-weight scripting language that fits in the size, memory bandwidth, and real-time requirements of applications like video games. Squirrel offers a wide range of features like:

* Open Source MIT licence
* Dynamic typing
* Delegation
* Classes & inheritance
* Higher order functions
* Lexical scoping
* Generators
* Cooperative threads (co-routines)
* Tail recursion
* Exception handling
* Automatic memory management (CPU bursts free; mixed approach ref counting/GC)
* Both compiler and virtual machine fit together in about 7k lines of C++ code and add only around 100kb-150kb the executable size.
* Optional 16bits characters strings
* Powerful embedding API
  + e.g. function/classes can be defined by scripts or in C
  + e.g. objects can fully exist in the VM or be bound to native code
  + e.g. classes created in C can be extended by scripts or vice-versa and more

Squirrel is inspired by languages like Python, Javascript and especially Lua (The API is very similar and the table code is based on the Lua one)

# Review of Server Construction

If you are completely new to any programming and how it all works there are some things you need to know on how it all works. By knowing these things, life is going to be a lot smoother for you; all those noob questions I’m pretty sure you aren’t seeing being answered by the community I address as I was in the same boat as you.

### The Files & Folders

Generally there are a group of files that should have your attention when making a server. These are:

* server.cfg
* server.conf
* server.exe
* main.nut
* plugin DLLs

These but not limited to these, are the files necessary for your server to work and are used in your life of scripting. The main focus is going to be the main.nut file for scripting but here’s a basic overview of the importance of each file.

* server.cfg – specifies the name of your server’s gamemode, the plugins that are going to be used by the server, the port used for hosting the server, the script containing the gamemode and the max number of players to the server.
* server.conf – one of the more important files that should have your attention, this file can be called “the scaffhold”, “the skeleton” or “the foundation” of your server. This file specifies the name of your server and controls a vast variety of server settings such as the weather, gravity, world boundaries of the game, spawn screen position and gamespeed just to name a few. It is also where ALL your game objects (cars, helis, boats, pickup items) are placed to specify their spawn position in the world of the game and it is the file that specifies your team’s classes and their skins along with their spawn points. This file is almost as important as the main.nut.
* server.exe – this is the executable that picks up the data from all the other files, compiles it into low level language for the computer to understand and creates the server to join.
* Main.nut – this is “the brain” of the server; this is where all the Squirrel code goes to in scripting. This is the most important file as without this, there is no interaction between players and the features of the server. If you want commands to fly across the sky, best believe that it’s not going to happen without coding it into this file. Without anything in this file I doubt the server would even run; you probably would not be able to spawn or even see messages on screen to you, nothing.
* Plugin DLLs – this file is what the server.exe uses to understand all the files you throw at it in low level language for the computer to process. For example, the codes we have using in main.nut means absolute gibberish to the computer currently. They have to be translated somehow, that’s where the DLLs come in. They work in conjunction with server.exe to understand what you wrote, compile it to computer language then runs it all to produce the server. You can make your own DLLs for any accessory file you would like to use for your server or for using a totally different language to code your server (like C++, however this process is incredibly technical and you would have to recreate the entire library of functions and callbacks in C++ coding).

There are 3 default folders that are needed for some of these files and any other file you would wish to implement:

* Plugins
* Scripts
* Store

The plugins folder contains all the server’s plugins needed to function. The scripts folder contains all your scripts, in this case, the main.nut. The store folder contains any modifications that you could have for your server (addition of non-GTA VC objects for example). The main focus is typically the plugins and scripts folders.

### Template of the Files

#### Server.cfg

Server.cfg, server.conf and main.nut all have their certain way of writing these files.

The easiest is the server.cfg. All you need for this one are these headings and the appropriate information:

* gamemode (state gamemode)
* plugins (place name of all plugins used)
* port (state port to use)
* sqgamemode (direct the file to the main.nut file)
* maxplayers (set max players that can join the server)

*Example server.cfg:*

gamemode DeathMatch

plugins squirrel04rel64 xmlconf04rel64 announce04rel64 sockets04rel64 hashing04rel64

port 8192

sqgamemode scripts/main.nut

maxplayers 100

#### Server.conf

The server.conf file uses XML coding tags to set each of its settings. It is more complex than server.cfg however and probably would take some serious practice to get them all by heart but they are all very simple I guarantee you that. The XML coding is abundant in its information to learn from so I won’t bother with telling you how to write it, I will however tell you the different settings and what they do.

<ServerName> - this is where you put in the server’s name

<FriendlyFire> - sets whether friendly fire can be enabled or not. Input true or false

<ShowOnRadar> - sets whether players and in game items are shown on the radar. Input true or false

<WeatherDefault> - sets the default weather of the game. Input integers

<HourDefault> - sets the default hour the game starts at. Input integer based on 24 hour clock.

<MinuteDefault> - sets the default minute of the hour the game starts at. Input integer

<TimeRate> - sets the rate the time changes at. NOTE THAT THE UNIT FOR THIS IS MILLISECONDS.

<PlayerPos> - sets the player position at spawn screen. Input the x, y and z axes coordinates.

<CamPos> - sets the camera position at spawn screen. Input the x, y and z axes coordinates.

<CamLook> - sets the camera’s viewing position at spawn screen. Input the x, y and z axes coordinates.

*CamPos and CamLook can get confusing, so let me give an example through an analogy. You want to record your PC’s monitor so you place your camera on a nearby object (CamPos’s function). The camera may be placed nearby but it doesn’t mean it’s placed looking at the monitor, so we shift it around to look at the monitor (CamLook’s function).*

<WorldBoundaries> - sets the world boundaries of the server. Input a min & max x, y and z axis value.

<SyncFrameLimiter> - TBD. Input true or false

<FrameLimiter> - TBFD. Input true or false

<TaxiBoostJump> - sets whether the zebra cabbie is capable of jumping. Input true or false

<DriveOnWater> - sets whether you can drive on water or not. Input true or false

<FastSwitch> - TBFD. Input true or false

<DisableDriveBy> - sets whether drive by is an available to be used by players or not. Input true or false

<PerfectHandling> - sets the perfect handling for bikers (they stay on the bike even under circumstances where they would surely fall off). Input true or false

<FlyingCars> - TBFD. Input true or false

<JumpSwitch> - sets whether you can switch your weapon while you’re jumping. Input true or false

<DeathMessages> - sets whether death messages are shown in game. Input true or false

<Gravity> - sets the gravity of the server. Input a real number

<GameSpeed> - sets the game speed of the server. Input a real number

<WaterLevel> – sets the water level of the game. Input a real number

<ShootInAir> - sets whether you are able to shoot while falling/suspended in the air. Input true or false

<JoinMessages> - sets whether join messages are shown in game. Input true or false

<ShowNametags> - sets whether nametags of players are shown. Input true or false

<StuntBike> - TBFD. Input true or false

All of those were all the different options for the server’s settings, affecting the server’s world and gameplay. Make note that all these settings are contained under the <Settings></Settings> tags. Without this, none of the settings would be applied. All that’s left now for server.conf are the addition of classes, pickups and the vehicles.

##### Classes

Classes are used to establish the different skins to choose from and the team each skin represents.

<Class team =”integer” skin =”integer” x =”float” y =”float” z =”float” angle =”radians” weapon =”integer” ammo =”integer” r=”integer” g =”integer” b =”integer”/>

Tag parameters’ colours may vary depending on the application used as your choice of editor

* team – this is where you establish the different teams within the game. You have 16 options from the integers 1 – 16.
* Skin – this is where you assign the different skins for your team. You can have as many skins as you please per team. Clashes are not allowed.
* x, y and z – these are your spawn coordinates for these skins. When the skin is selected, they will go to the coordinate you assigned when they spawn. Input values that are within the world’s boundaries.
* Angle – this is the spawning angle of the skin (the direction the player is facing by default when he spawns). **The angles are in radians, NOT degrees.** So if you want them facing exactly east for example, don’t put in 90 degrees, you will have headaches.
* Weapon & ammo – this is where you set the weapon of choice and the ammo amount for the particular skin.
* r, g, b – this is where you assign the team’s colour (what colour their names will be when they select this skin). Your values range from 0 – 255 of reds, greens and blues (r, g and b respectively).

##### Pickups

Pickups are used when you want default items to be picked up by players as they walk into them.

<Pickup model=”integer” x=”float” y=”float” z=”float” world=”integer” />

Tag parameters’ colours may vary depending on the application used as your choice of editor

* model – this is where you assign the model ID of the pickup item you want to use.
* x, y and z – these are the coordinate points which determines where the pickup will be placed in the world of the game. Input values that are within the world’s boundaries.
* world – TBFD

##### Vehicles

This is where you go about placing vehicle spawn points into the world of the game.

<Vehicle model=”integer” x=”float” y=”float” z=”float” angle=”degrees” col1=”integer” col2=”integer” />

Tag parameters’ colours may vary depending on the application used as your choice of editor

* model – this is where you assign the model ID of the pickup item you want to use.
* x, y and z – these are the coordinate points which determines where the vehicles will be placed in the world of the game. Input values that are within the world’s boundaries.
* angle – this is the default angle of the vehicle when it spawns into the world of the game. (investigate whether it is degrees or radians)
* col1 & col2 – these are the default spawning colours of the vehicles. These colours are however not the same as rgb earlier in setting up the skin’s team colour. These are ID colours present to the game so you have to set your value based on that.

After all that is set and done, congratulations; you have pretty much learned how to configure how the world of the game will be when the server loads up, from the weather & gravity of the <settings> tag to the vehicles, pickups and player skins that you make available for use. This is the purpose of server.conf.

#### Main.nut

The main.nut file uses the Squirrel scripting language and is one of the most complex of them all. As stated in the ‘Files & Folders’ section, this is the “the brain” of your server. Everything interactive about the game happens because of this file. The fact that you can spawn, get messages, teleport to players, see killing spree notices, get banned/kicked, have player stats, change the weather, change the gamespeed; this is where it’s all done from.

There are areas you should have in your script as a template for all your code later on. When we get to Scripting Guide 101, we will discuss all the different functions and codes at your disposal for VCMP but for now, this is just addressing the appropriate template needed for the main.nut file.

The main.nut file needs 6 official events to be placed in the code in order for everything to work ideally how you would want it to. These are:

* Server Events
* Player Events
* Vehicle Events
* Pickup Events
* Object Events
* Bind Events

Scripting doesn’t have to be 100% correct for the server itself to run as each event runs independently of the other. You could say that only certain events and/or certain functions have to be absolutely correct for the server to run and not feel like there are not any issues.

##### Server Events

Server Events is the section of the script where you will outline everything that needs to be done as the script is loaded. Think of it as a preparation phase. Custom files and modifications that you would like in your server would need to be coded in this section for example as well as separate nut files that you may write functions in to keep the main.nut file from being excessively long.

###### Official Functions within Server Events

* onScriptLoad – loads all the coding that is to happen as the script is loaded by server.exe.
* onScriptUnload – loads all the coding that is to happen as the scripted is unloaded by server.exe.

##### Player Events

This is the section where you code all the functions and interactions a player gets from the server. This stems from the moment the player joins the server to the messages he gets when he spawns or is choosing a skin to the killing spree notification and death notifications to the commands that are capable to be used at the player’s disposal; this is the section where you code in this information.

###### Official Functions within Player Events

* onPlayerJoin – loads all the coding that is to happen when a player joins the server.
* onPlayerPart - TBFD
* onPlayerRequestClass – loads all the coding that is to happen when a player requests a class.
* onPlayerRequestSpawn – loads all the coding that is to happen when a player requests to spawn.
* onPlayerSpawn – loads all the coding that is to happen when a player spawns.
* onPlayerDeath – loads all the coding that is to happen when a player dies.
* onPlayerKill – loads all the coding that is to happen when a player is killed.
* onPlayerTeamKill – loads all the coding that is to happen when a player is team-killed.
* onPlayerChat – loads all the coding that is to happen when a player chats in-game.
* onPlayerTeamChat – loads all the coding that is to happen when a player team-chats in-game.
* onPlayerCommand – loads all the functions that is at the player’s disposal in the form of commands typed by the player at any given time.
* onPlayerPM – loads all the coding that is to happen when a player gets a private message from another player.
* onPlayerBeginTyping – loads all the coding that is to happen as a player is typing in-game.
* onPlayerEndTyping – loads all the coding that is to happen as a player is finished typing in-game.
* onNameChangeable - TBFD
* onPlayerSpectate – loads all the coding for a player in spectate mode.
* onPlayerCrashDump - TBFD
* onPlayerMove – loads all the coding that is to happen as a player moves.
* onPlayerHealthChange – loads all the coding that is to happen once a player’s health is changed.
* onPlayerArmourChange – loads all the coding that is to happen once a player’s armour is changed.
* onPlayerWeaponChange – loads all the coding that is to happen once a player’s weapon is changed.
* onPlayerAwayChange – loads all the coding that is to happen once a player has changed from his away status.
* onPlayerActionChange – loads all the coding that is to happen once a player’s action has changed.
* onPlayerStateChange - TBFD
* onPlayerOnFireChange - TBFD
* onPlayerCrouchChange - TBFD
* onPlayerGameKeysChange - TBFD

##### Vehicle Events

This is the section where you code particular functions that you would like for vehicles for example if you wanted a car alarm to go off once a player enters a vehicle, you would code it here.

###### Official Functions within Vehicle Events

* onPlayerEnteringVehicle – loads all the coding that is to happen once a player starts entering a vehicle.
* onPlayerEnterVehicle – loads all the coding that is to happen once a player has entered a vehicle.
* onPlayerExitVehicle – loads all the coding that is to happen once a player has exited a vehicle.
* onVehicleExplode – loads all the coding that is to happen once a vehicle has exploded.
* onVehicleRespawn – loads all the coding that is to happen once a vehicle is respawned.
* onVehicleHealthChange – loads all the coding that is to happen once a vehicle’s health is changed.
* onVehicleMove – loads all the coding that is to happen once a vehicle is moved.

##### Pickup Events

This is where you code particular functions that you would like to happen when a player interact with pickups in the game. So if there is a health pickup, it will not automatically heal the person if they walk through it, you have to code it in this section what will happen.

###### Official Functions within Pickup Events

* onPickupClaimedPicked – loads all the coding that is to happen once a player’s body has claimed to pickup a pickup object.
* onPickupPickedUp – loads all the coding that is to happen once a player has picked up a pickup object.
* onPickupRespawn – loads all the coding that is to happen once a pickup object has respawned.

##### Object Events

This is where you would code particular functions that you would like to happen when a player interact with objects in the game. An interesting idea that you would code here is if you wanted a particular wall to reflect all weapon damage to the player that shoots it for example.

###### Official Functions within Object Events

* onObjectShot – loads all the coding that is to happen once an object is shot.
* onObjectBump – loads all the coding that is to happen once an object is bumped into.

##### Bind Events

This is where you would code particular functions that you would like to happen when a player presses a particular key on the keyboard. Bind events are a great way to bind certain commands to particular keys so as to avoid disrupting the game. For example if a player wanted a speedometer to be printed onscreen whilst racing, it would be quite inconvenient for him to type that command. Instead you would use this section to a certain key bring forth that command so that they can toggle on/off the speedometer in the heat of the race as they wish.

###### Official Functions within Bind Events

* onKeyDown – loads all the coding that is to happen once a key is pressed.
* onKeyUp – loads all the coding that is to happen once a key is released.

### Plugin DLLs

As previously stated in The Files & Folders section, plugin DLLs are what the server.exe file uses to understand the files that we throw at it. I also mentioned before that because you can compile your own DLL file, you do not have to code in Squirrel language if you don’t want to. Some servers in the VCMP world today use C++ language to create their servers.

What I didn’t mention however was that suppose you would like to implement an object within the server world. For the object to be recognized by the game, you would have to create a plugin DLL file that would be able to decode the information from this file so that it is useable by the server. Another thing was that if I believe that if I wanted to totally reconstruct the server ecosystem and have particular capabilities like artificial intelligence or an in-game internet browser, in-game voice chat, in-game movie creator along the likes of Rockstar Editor for Grand Theft Auto 5, this is essentially where you would have to end up to implement it.

It is because of this why VCMP 0.4 has great potential but these are not easy feats to accomplish even by the most expertise scripters around. Just take this as a reiteration of the importance of plugin DLLs and the potential it has to make your server truly one of a kind.

# Scripting Guide 101