EvoSC Developers Guide

This section covers everything you need to maintain or extend the Evo Server Controller

- Getting Started
  - Requirements and Installation
  - Workflow
- The Architecture
  - Project Structure
  - Events/Hooks
- Modules

Getting Started

Prerequisites and other userful information

Requirements and Installation

Requirements

- PHP7.4 (with extensions: simplexml, mbstring, gd, dom, mysql)
- Composer
- MySQL/MariaDB Database

Installation

- Clone the project to your desired location
- Go into the directory and call ***composer install***

Workflow

Features and Bugfixes

When starting to work on a new feature or bugfix, create a new branch from develop and make your changes. After you're done commit and push your changes to the repository and create a pull-request from your branch to develop.

If working on a GitHub-Issue put the ticket-number into the branch name, so we can easily follow the development process.

The Architecture

How are files structured and named? General development guidelines.

Project Structure

Directories

cache

As the name suggests this directory contains all the cache-data saved and used by EvoSC. This directory must not be populated with files by anyone other than EvoSC.

There is one exception to that: If you want to tell EvoSC to restart through the filesystem, create a file called "restart_evosc" inside the cache-directory. EvoSC looks for the existence of that file and reboots itself when found. This can be used for update scripts, or if you just dont want to join the server. A graceful shutdown is always better for the players online, than killing the process.

config

This directory contains all the configuration files that EvoSC uses. This is the one and only place a config should be changed by a user. A user must not change the files inside core (more specific the ones in the modules). The *default* sub-directory contains the default configs, which do not belong to any module.

On controller start, all default configs from the modules and the default directory get copied to the config directory, if they are missing. All existing configs are compared to the defaults, so if a new value was added, it gets added to the json in *config.* Also when removing a key in the default, it will be deleted in the json in *config*.

The filename is also the root id to call values. Example: To get the **log file prefix** from **server.config.json** you would call `config('server.log.prefix')`

core

Here's where the code resides, only devs should touch this directory.

Classes	Some classes to help out structure commonly used data and structures.

Commands	This directory contains all callable CLI commands.
Controllers	Controllers are thought to be mandatory parts of EvoSC, that must run to make it work and provide informations for the modules.
Dictionary	A base for the not yet used language support.
Exceptions	EvoSCs Exceptions.
Interfaces	Interfaces.
Models	Eloquent database models more here: https://laravel.com/docs/7.x/eloquent#eloquent-model-conventions
Modules	Modules that ship with EvoSC reside in this directory.
TemplateComponents	Different reusable templates and scripts to use for ManiaLink creation with latte-template-engine.
Tests	Tests.

global-functions.php contains methods that are available throughout the whole project at any time.

logs

Logging files are created here.

Migrations

EvoSC uses incremental database creation through migrations. Migrations created with the CLI command are created in this directory and may be copied to your module from there, if you want to create a thrird-party module.

modules

Put third-party modules (which are not part of EvoSC) into this directory.

vendor

All the php libraries installed by composer.

Events/Hooks

The server communicates with the controller through callbacks/events. Some of the more important events are preprocessed by EvoSC to make them easier to use. All those events called "Hooks" will be listed below.

Registering a Hook

To bind a method to an EvoSC-Hook or TM-Event, you simply call

```
Hook::add('NameOfHookOrEvent', [FullyQualifiedClassName, 'methodName');
```

inside the start method of the module.

You can always create a hook for the methods listed in https://github.com/maniaplanet/script-xmlrpc/blob/master/XmlRpcListing.md#callbacks

Hooks

Name	Passed arguments	Description
PlayerConnect	**Player** $player	Called when a player spawns in the server
PlayerDisconnect	**Player** $player	Called when a player leaves the server
PlayerFinish	**Player** $player, **int** $timeInMilliseconds, **string** $checkpointsCommaSeparated	Called when a player finishes or resets, then the time is 0
PlayerCheckpoint	**Player** $player, **int** $timeInMilliseconds, **int** $checkpointNumber, **bool** $isFinish	Called when a player passes a checkpoint
PlayerStartCountdown	**Player** $player	Called when the 3-2-1-countdown starts for a player
PlayerPb	**Player** $player, **int** $timeInMilliseconds	Called when player drives a new personal best
BeginMatch	-	Called when the countdown/match starts

EndMatch	-	Called when the match ended
BeginMap	**Map** $map	Called when a map starts and when EvoSC boots
EndMap	**Map** $map	Called when a map ends
MatchSettingsLoaded	**string** $matchSettingsFile	Called when a match-settings is loaded
AddedTimeChanged	**int** $addedSeconds	Called when the timelimit was in/decreased
MapPoolUpdated	-	Called when the maplist changed
MatchTrackerUpdated	**Collection** $scores	Called when scores affecting the match changed (player finished with better time, got points, etc...)
MapQueueUpdated	**Collection** $mapsInQueue	Called when a map was added/removed to/from the jukebox
WarmUpStart	-	Called when the warmup phase starts
WarmUpEnd	-	Called when the warmup phase ends
AnnounceWinner	**Player** $winner	Called on match end, when the winner is decided
ShowScores	**Collection** $allPlayersThatParticipatedInMatch	Called when ther end result is displayed after the match ended
GroupChanged	**Player** $player	Called when the group of a player was changed

Modules

Everything you need to know about modules