Shared function

AddCommandHandler

> **Important Note:** Do **NOT** use the same name for your handler function as the command name, as this can lead to confusion if multiple handler functions are used. Use a name that describes your handler's purpose more specifically.

This function will attach a scripting function (handler) to a console command, so that whenever a player or administrator uses the command the function is called.

Multiple command handlers can be attached to a single command, and they will be called in the order that the handlers were attached. Equally, multiple commands can be handled by a single function, and the *commandName* parameter used to decide the course of action.

For users, a command is in the format:

commandName argument1 argument2

This can be triggered from the player's console or directly from the chat box by prefixing the message with a forward slash (*/*). For server side handlers, the server admin is also able to trigger these directly from the server's console in the same way as they are triggered from a player's console.

> **Note:** You can't use "check", "list", "test" and "help" as a command name.

Syntax

Server

```
bool addCommandHandler ( string commandName, function handlerFunction [, bool restricted = false, bool caseSensitive = true ]
 )
```

Required Arguments

- **commandName:** This is the name of the command you wish to attach a handler to. This is what must be typed into the console to trigger the function.
- **handlerFunction:** This is the function that you want the command to trigger, which has to be defined before you add the handler. This function can take two parameters, playerSource and commandName, followed by as many parameters as you expect after your command (see below). These are all optional.

Optional Arguments

NOTE: When using optional arguments, you might need to supply all arguments before the one you wish to use. For more information on optional arguments, see optional arguments.

- **restricted:** Specify whether or not this command should be restricted by default. Use this on commands that should be inaccessible to everyone as default except special users specified in the ACL (Access Control List). This is to make sure admin commands such as ie. 'punish' won't be available to everyone if a server administrator forgets masking it in ACL. Make sure to add the command to your ACL under the proper group for it to be usefull (i.e <right name="command.killEveryone" access="true"></right>). This argument defaults to false if nothing is specified.

- **caseSensitive:** Specifies if the command handler will ignore the case for this command name.

Client

```
bool addCommandHandler ( string commandName, function handlerFunction [, bool caseSensitive = true ] )
```

Required Arguments

- **commandName:** This is the name of the command you wish to attach a handler to. This is what must be typed into the console to trigger the function.
- **handlerFunction:** This is the function that you want the command to trigger, which has to be defined before you add the handler. This function can take commandName parameter, followed by as many parameters as you expect after your command (see below). These are all optional.

Optional Arguments

NOTE: When using optional arguments, you might need to supply all arguments before the one you wish to use. For more information on optional arguments, see optional arguments.

- **caseSensitive:** Specifies if the command handler will ignore the case for this command name.

Handler function parameters

These are the parameters for the handler function that is called when the command is used.

Server

```
player playerSource, string commandName [, string arg1, string arg2, ... ]
```

- **playerSource:** The player who triggered the command or the server console. If not triggered by a player (e.g. by admin) or server console, this will be *false*.
- **commandName:** The name of the command triggered. This is useful if multiple commands go through one function.
- **arg1, arg2, ...:** Each word after command name in the original command is passed here in a seperate variable. If there is no value for an argument, its variable will contain nil. You can deal with a variable number of arguments using the vararg expression, as shown in **Server Example 2** below.

Client

```
string commandName [, string arg1, string arg2, ... ]
```

- **commandName:** The name of the command triggered. This is useful if multiple commands go through one function.
- **arg1, arg2, ...:** Each word after command name in the original command is passed here in a seperate variable. If there is no value for an argument, its variable will contain nil. You can deal with a variable number of arguments using the vararg expression, as shown in **Server Example 2** below.

Returns

Returns *true* if the command handler was added successfully, *false* otherwise.