Shared function

BindKey

Binds a player's key to a handler function or command, which will be called when the key is pressed.

> **Note:** Using escape key will always return false. Use onClientKey event instead.

> **Note:** Handler function won't be triggered while focused in CEGUI editbox. You can use guiSetInputMode or onClientKey in order to fix that.

Syntax

Server - Syntax 1

```
bool bindKey ( player thePlayer, string key, string keyState, function handlerFunction,  [ var arguments, ... ] )
```

Required Arguments

- **thePlayer:** The player you wish to bind the key of.
- **key:** The key or control you wish to bind to the command. See key names for a list of possible keys and control names for a list of possible controls.
- **keyState:** A string that has one of the following values:
  - **"up":** If the bound key should trigger the function when the key is released
  - **"down":** If the bound key should trigger the function when the key is pressed
  - **"both":** If the bound key should trigger the function when the key is pressed or released
- **handlerFunction:** The function that will be triggered when the player's key is pressed. This function should have the form:

  ```
  function functionName ( player keyPresser, string key, string keyState, [ var arguments, ... ] )
  ```

  The values passed to this function are:

  - **keyPresser:** The player who pressed the key
  - **key:** The key that was pressed
  - **keyState:** The state of the key that was pressed, *down* if it was pressed, *up* if it was released.
  - **arguments** The optional arguments you specified when calling bindKey (see below).

Server - Syntax 2

This alternative syntax allows you to bind a key with a command. This will also allow users to customize the control in their Settings menu. Use in conjunction with addCommandHandler to add handler functions to the keybind.

```
bool bindKey ( player thePlayer, string key, string keyState, string commandName, [ string arguments ] )
```

Required Arguments

- **thePlayer:** The player you wish to bind the key of.
- **key:** The key or control you wish to bind to the command. See key names for a list of possible keys.
- **keyState:** A string that has one of the following values:
  - **"up":** If the bound key should trigger the function when the key is released
  - **"down":** If the bound key should trigger the function when the key is pressed
  - **"both":** If the bound key should trigger the function when the key is pressed or released
- **commandName:** The name of the command that the key should be binded to.

Optional Arguments

- **arguments** Space delimited arguments that are entered as if one was typing the command.

Client - Syntax 1

```
bool bindKey ( string key, string keyState, function handlerFunction,  [ var arguments, ... ] )
```

Required Arguments

- **key:** The key or control you wish to bind to the command. See key names for a list of possible keys and control names for a list of possible controls.
- **keyState:** A string that has one of the following values:

    - **"up":** If the bound key should trigger the function when the key is released
    - **"down":** If the bound key should trigger the function when the key is pressed
    - **"both":** If the bound key should trigger the function when the key is pressed or released
- **handlerFunction:** The function that will be triggered when the player's key is pressed. This function should have the form:

  ```
  function functionName ( string key, string keyState, [ var arguments, ... ] )
  ```

  The values passed to this function are:

  - **key:** The key that was pressed
  - **keyState:** The state of the key that was pressed, *down* if it was pressed, *up* if it was released.
  - **arguments** The optional arguments you specified when calling bindKey (see below).

Client - Syntax 2

This alternative syntax allows you to bind a key with a command. This will also allow users to customize the control in their Settings menu. Use in conjunction with addCommandHandler to add handler functions to the keybind.

```
bool bindKey ( string key, string keyState, string commandName, [ string arguments ] )
```

Required Arguments

- **key:** The key or control you wish to bind to the command. See key names for a list of possible keys.
- **keyState:** A string that has one of the following values:
    - **"up":** If the bound key should trigger the function when the key is released
    - **"down":** If the bound key should trigger the function when the key is pressed
    - **"both":** If the bound key should trigger the function when the key is pressed or released
- **commandName:** The name of the command that the key should be binded to.
- **arguments** Space delimited arguments that are entered as if one was typing the command.

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.

- **arguments:** Any arguments you may want to pass to the function when the key is pressed by the user. Any number of arguments of can be specified, each being passed to the designated function. You may not pass functions.

Returns

Returns *true* if the key was bound, *false* otherwise.