



# XML Notes

[Jump to bottom](#)

® omano edited this page Nov 25, 2019 · [19 revisions](#)

## OGP XML Notes / still W.I.P.

*The order of each XML element matters, and this guide presents them in their order of appearance!*

### Linux and Windows:

#### Game Config:

This is the first element. There can only be one `<game_config>` element.

```
<game_config>
the whole XML file content here
</game_config>
```

All the following elements should be contained within `<game_config>` element.

#### Game Key:

Comes after `<game_config>` element (actually within `<game_config>` element as said above). There can only be one `<game_key>` element. Example:

```
<game_key>space_engineers_win64</game_key>
```

This is a unique key used to identify this specific game server in OGP. You should not use spaces, nor any special character in it, only alpha-numeric value and underscores. It should contain a suffix related to the compatible OS. Available suffixes are `_win32` , `_win64` , `_linux32` , `_linux64` , using one of these suffixes in the `game_key` will let OGP know which OS it is available on, making it visible or not when you install a new game server, depending on your OS architecture. `_win` and `_linux` work too, but we highly recommend to now use the previously listed suffixes.

### Query Protocol:

Comes after `<game_key>` element. There can only be one `<protocol>` element. Example:

```
<protocol>lgs1</protocol>
```

It defines the query protocol used by OGP. Available protocols are `lgs1` , `gameq` , `rcon` ( `rcon2` ? `lcon` ? )

### LGSL Query Name:

Comes after `<protocol>` element. There can only be one `<lgs1_query_name>` element. Example:

```
<lgs1_query_name>killingfloor2</lgs1_query_name>
```

This is the unique key referencing this specific game server in the LGSL protocol file, used to query the game server.

### GameQ Query Name:

Comes after `<protocol>` element. There can only be one `<gameq_query_name>` element. Example:

```
<gameq_query_name>redorchestra2</gameq_query_name>
```

This is the unique key referencing this specific game server in the GameQ protocol files, used to query the game server.

### Installer:

Comes after `<protocol>` element (or comes after `<lgs1_query_name>` or `<gameq_query_name>` element when used). There can only be one `<installer>` element. Example:

```
<installer>steamcmd</installer>
```

Defines the use of SteamCMD tool to install the game server.

### Game Name:

Comes after `<installer>` element. There can only be one `<game_name>` element. Example:

```
<game_name>Killing Floor 2</game_name>
```

This is the real game server name appearing in the list when installing a new game server.

### Server Executable Name:

Comes after `<game_name>` element. There can only be one `<server_exec_name>` element. Example:

```
<server_exec_name>SpaceEngineersDedicated.exe</server_exec_name>
```

This is the server executable name used in the start command line.

### Query Port:

Comes after `<server_exec_name>` element. There can only be one `<query_port>` element. Example:

```
<query_port type='add'>13</query_port>
```

Difference between the server port ( `%PORT%` ) and the query port ( `%QUERY_PORT%` ). In this example the variable `%QUERY_PORT%` will be 13 added to the port value.

### CLI Template:

Comes after `<server_exec_name>` element. There can only be one `<cli_template>` element. Example:

```
<cli_template>-console %BASE_PATH% -ignorelastsession</cli_template>
```

```
<cli_template>%MAP%%GAMEMODE%%DIFFICULTY%%GAMELENGTH%%PLAYERS%%MUTATOR% %PORT% %IP%  
%WEB_ADMIN_PORT% %QUERY_PORT%</cli_template>
```

This is the template that will generate the start command line placed after the server executable name. You can use these variables which are known to OGP:

```
GAME_TYPE  
HOSTNAME  
IP  
MAP  
PID_FILE  
PLAYERS  
PORT  
QUERY_PORT  
BASE_PATH  
HOME_PATH  
SAVE_PATH  
OUTPUT_PATH  
CONTROL_PASSWORD
```

These variable should be between % characters, the Panel will then replace them with the proper value when generating the start command line.

You can also use custom variables that you will define later in the XML.

### CLI Parameters:

Comes after <cli\_template> element. There can only be one <cli\_params> element. Example:

```
<cli_params>  
  <cli_param id="MAP" cli_string="" />  
  <cli_param id="IP" cli_string="-MultiHome=" options="q"/>  
  <cli_param id="PORT" cli_string="-Port=" options="sq"/>  
  <cli_param id="PID_FILE" cli_string="" />  
  <cli_param id="GAME_TYPE" cli_string="" />  
  <cli_param id="HOME_PATH" cli_string="" />  
</cli_params>
```

It defines the known variables used in `<cli_template>` . In this example we can imagine that for `%MAP%` it will generate the map name without space or quotes around it, because there is no options , for `%IP%` it will generate `-MultiHome="123.123.123.123"` using the `cli_string` and adding only quotes around the game server IP value because of `options="q"` , and `%PORT%` will generate `-Port= "27015"` using the `cli_string` and adding space and quotes around the game server port because of `options="sq"` .

### Reserve Ports:

Comes after `<cli_template>` or `<cli_params>` element. There can only be one `<reserve_ports>` element. Example:

```
<reserve_ports>
  <port type="subtract" id="WEB_ADMIN_PORT" cli_string="-WebAdminPort=">5</port>
  <port type="add" id="STEAM_PORT" cli_string="-SteamPort=">19238</port>
  <port type="add" id="MY_CUSTOM_PORT">666</port>
</reserve_ports>
```

You can add reserved ports here to use in the generated start command line. These ports will also be managed by OGP if OGP is used to manage the Agent machine firewall. Type can be `add` or `subtract` which is self explanatory. In this example when using the `%WEB_ADMIN_PORT%` variable in the `<cli_template>` it will generate `-WebAdminPort=XXX` , XXX being 5 subtracted to the Port set for this game server, when using the `%STEAM_PORT%` variable in the `<cli_template>` it will generate `-SteamPort=XXX` where XXX will be 19238 added to the Port set for this game server. As you can see, the variable `%MY_CUSTOM_PORT%` have no `cli_string` , this can be used this way to simply open this port (which here would be 666 added to the game server port) in the Agent machine firewall, when OGP is set to control the machine firewall (note: the firewall management may actually not open anything else than the game server port, to be verified).

### CLI Allowed Characters:

Comes after `<reserve_ports>` element. There can only be one `<cli_allow_chars>` element. Example:

```
<cli_allow_chars>;</cli_allow_chars>
```

Used to allow some special characters in the command line. Escaped by default: `\ " ' | & ; > < ` $ ( ) [ ]`

## Maps Location:

Comes after `<cli_template>` element. There can only be one `<maps_location>` element. Example:

```
<maps_location>folder/maps</maps_location>
```

It sets the path of the map folder for this game server, which will be used to generate a selectable map list available before starting the game server. If folder contains map files it will use their name without the extension, if it contains sub folders with each map inside each own sub folder, it will generate the map list based on the folders names contained in the defined path. The selected map in the list will be used to replace the `%MAP%` variable in the `<cli_template>` .

## Map List:

Comes after `<cli_template>` element. There can only be one `<map_list>` element. Example:

```
<map_list>maplist.txt</map_list>
```

The map list file path used to generate the selectable map list available before starting the game server. In this example it will look for a file called `maplist.txt` inside the root of the game server. Map list should have one map per line. The selected map in the list will be used to replace the `%MAP%` variable in the `<cli_template>` .

## Console Log:

Comes after `<cli_template>` element. There can only be one `<console_log>` element. Example:

```
<console_log>KFGame/Logs/Launch.log</console_log>
```

It defines the path of the log file that will be shown in the LOG page of the game server. Most game servers, especially on Linux, will not need that, when in general, Windows game server will need it to properly show the output log.

## Executable Location:

Comes after `<console_log>` element. There can only be one `<exe_location>` element. Example:

```
<exe_location>Binaries/Win64</exe_location>
```

It defines the path of the game server executable when not in the root folder.

### Max User Amount:

Comes after `<exe_location>` element. There can only be one `<max_user_amount>` element. Example:

```
<max_user_amount>64</max_user_amount>
```

It defines the maximum player number you will be able to set when creating the game server.

### Control Protocol:

Comes after `<max_user_amount>` element. There can only be one `<control_protocol>` element. Example:

```
<control_protocol>rcon2</control_protocol>
```

Can be `rcon`, `rcon2`, or `lcon` (legacy). Note that `rcon` can also have type option to define, which can be `old` or `new`. Example:

```
<control_protocol>rcon</control_protocol>  
<control_protocol_type>old</control_protocol_type>
```

### Mods:

Comes after `<control_protocol>` element. There can only be one `<mods>` element. Example:

```
<mods>  
  <mod key="insurgency">  
    <name>none</name>  
    <installer_name>237410</installer_name>  
  </mod>  
</mods>
```

Used to define different mods for the game server, in this example there is only one mod available which will be the default installed one (actually the game server itself here). The `<installer_name>` here is the Steam appID that will be used to install and update the game server. (note: case RSync to explain? Case multi mods to explain?)

## Replace Texts

`<replace_texts>` Comes after `<mods>` .

Contains multiple `<text>` entries like so:

```
<replace_texts>
  <text key="control_password">
    <default>ServerAdminPassword=.*</default>
    <var>ServerAdminPassword=</var>
    <filepath>ShooterGame/Saved/Config/LinuxServer/GameUserSettings.ini</filepath>
    <options>sq</options>
  </text>
  <text key="home_name">
    <default>SessionName=.*</default>
    <var>SessionName=</var>
    <filepath>ShooterGame/Saved/Config/LinuxServer/GameUserSettings.ini</filepath>
    <options>sq</options>
  </text>
</replace_texts>
```

`<default>` within `<text>` is what the line to replace starts with.

`<var>` within `<text>` is the key for what should be kept when the line is replaced with the value entered by the user when replacing text occurs.

`<filepath>` within `<text>` specifies the text file to make the replacement in.

`<options>` within `<text>` specifies how to enter the user's value after the `<var>` key. Possible options are:

```
nothing / no value = placed as is
s = space / separated
q = quoted
sq = space and quotes
sc = space and ends with a comma
sqc = space, quoted, and ends with a comma
```



These replace text will be applied on server start and modify the specified config files with the values generated by the Panel.

## Server Params:

`<server_params>` Comes after `<replace_texts>` .

Contains multiple `<param>` entries like so:

```
<server_params>
  <param id="SP" key="?ServerPassword=" type="text">
    <option>ns</option>
    <caption>Server Password</caption>
    <desc>Players must know this password to connect.</desc>
  </param>
  <param id="DIFFICULTY" key="?Difficulty=" type="select">
    <option value=""></option>
    <option value="0">Normal</option>
    <option value="1">Hard</option>
    <options>ns</options>
    <caption>Difficulty</caption>
    <desc>This sets the server difficulty. Leave empty to configure this parameter in
the config files or webadmin</desc>
  </param>
  <param key="-EnableCheats" type="checkbox_key_value">
    <caption>Cheats</caption>
    <desc>Enable the cheats to be used from the ingame Admin menu</desc>
  </param>
</server_params>
```

`id` attribute on the `<param>` specifies which variable to replace in the `<cli_template>` `key` attribute specifies what will replace the variable defined by `id` attribute `type` attribute will define what kind of parameter it is, possible values are `text` `select` `checkbox_key_value` :

- `text` will allow to write text value to be added during the replacement of the variables in startup command line. For example, `%SP%` in `<cli_template>` would be replaced with `?ServerPassword=XXX` where `XXX` would be the value entered by the user in the text box, if nothing is entered the variable is not replaced but removed from `<cli_template>` . The value entered can be modified to fit your needs by using the `<option>` element within the `<param>` element.
- `select` will create a selectable list to choose the parameter value. The `%DIFFICULTY%` variable in `<cli_template>` would be replaced with `?Difficulty=1` if `Hard` would have been selected before starting the server.

- checkbox\_key\_value will add a simple checkbox that, if ticked before starting server, would add the parameter -EnableCheats at the end of the startup command line.

Valid options are for the <options> element within the <param> element within the <server\_params> element are:

```
ns = no space between key and value
q = quotes wrapped around value after key (no space added)
s = space added after key before value (no quotes added)
anything else = space after key and quotes around the value
```

## Custom Fields:

Comes after <server\_params> element. There can only be one <custom\_fields> element. Example:

```
<custom_fields>
  <field key="sv_maxrate" type="text">
    <default>sv_maxrate.*</default>
    <default_value>0</default_value>
    <var>sv_maxrate</var>
    <filepath>orangebox/cspromod/cfg/server.cfg</filepath>
    <options>s</options>
    <desc>Max bandwidth rate allowed on server ( bytes per second ), 0 == unlimited.
  </desc>
  </field>
  <field key="sv_minrate" type="text">
    <default>sv_minrate.*</default>
    <default_value>5000</default_value>
    <var>sv_minrate</var>
    <filepath>orangebox/cspromod/cfg/server.cfg</filepath>
    <options>s</options>
    <desc>Min bandwidth rate allowed on server ( bytes per second ), 0 == unlimited.
  </desc>
  </field>
  <field key="sv_maxcmdrate" type="text">
    <default>sv_maxcmdrate.*</default>
    <default_value>66</default_value>
    <var>sv_maxcmdrate</var>
    <filepath>orangebox/cspromod/cfg/server.cfg</filepath>
    <options>s</options>
    <desc>If sv_mincmdrate is > 0, this sets the maximum value for cl_cmdrate.</desc>
  </field>
  <field key="sv_mincmdrate" type="text">
    <default>sv_mincmdrate.*</default>
```

```

    <default_value>67</default_value>
    <var>sv_mincmdrate</var>
    <filepath>orangebox/cspromod/cfg/server.cfg</filepath>
    <options>s</options>
    <desc>This sets the minimum value for cl_cmdrate. 0 == unlimited.</desc>
</field>
<field key="sv_maxupdaterate" type="text">
    <default>sv_maxupdaterate.*</default>
    <default_value>66</default_value>
    <var>sv_maxupdaterate</var>
    <filepath>orangebox/cspromod/cfg/server.cfg</filepath>
    <options>s</options>
    <desc>Maximum updates per second that the server will allow.</desc>
</field>
<field key="sv_minupdaterate" type="text">
    <default>sv_minupdaterate.*</default>
    <default_value>67</default_value>
    <var>sv_minupdaterate</var>
    <filepath>orangebox/cspromod/cfg/server.cfg</filepath>
    <options>s</options>
    <desc>Minimum updates per second that the server will allow.</desc>
</field>
</custom_fields>

```

Custom Fields available when you go to Custom Fields button from the game server page. These custom fields will be applied on every server start and will replace the specific values in the specified config files.

### List Players Command:

Comes after `<custom_fields>` element. There can only be one `<list_players_command>` element. Example:

```
<list_players_command>status</list_players_command>
```

This is the command to list the players on the server when using `control_protocol`.

### Player Info Regex:

Comes after `<list_players_command>` element. There can only be one `<player_info_regex>` element. Example:

```
<player_info_regex>#\#\s*(\d*)\s*"(.*)\".*#</player_info_regex>
```

Regex used for player info when using `control_protocol` .

### Player Info:

Comes after `<player_info_regex>` element. There can only be one `<player_info>` element.

Example:

```
<player_info>
  <index key="1">userid</index>
  <index key="2">Name</index>
</player_info>
```

Defines the different keys for player infos when using `control_protocol` .

### Player Commands:

Comes after `<player_info>` element. There can only be one `<player_commands>` element. Example:

```
<player_commands>
  <command key="Kick" type="hidden">
    <string>kick "%Name%"</string>
  </command>
  <command key="Ban" type="select">
    <option value="0">Permanent</option>
    <option value="15">15m</option>
    <option value="30">30m</option>
    <option value="60">1h</option>
    <option value="1440">1D</option>
    <option value="10080">1W</option>
    <option value="43200">30D</option>
    <string>banid %input% %userid% kick</string>
  </command>
  <command key="Change Nick" type="text">
    <default>new nick</default>
    <string>sm_rename #%userid% "%input%"</string>
  </command>
</player_commands>
```

Defines the commands available when using `control_protocol` .

## Pre Install:

Comes after `<player_commands>` element. There can only be one `<pre_install>` element. It can contain multiple entries one per line. Example:

```
<pre_install>  
</pre_install>
```

Script executed before installing the game server.

## Post Install:

Comes after `<pre_install>` element. There can only be one `<post_install>` element. It can contain multiple entries one per line. Example:

```
<post_install>  
cp linux64/steamclient.so ./steamclient.so  
echo &apos;#!/bin/bash  
./bin/AvorionServer $@&apos; &gt; avorion_ogpstarter.sh  
chmod +x avorion_ogpstarter.sh  
</post_install>
```

Script executed after installing or updating the game server, similar to `pre_install` (which `pre_install` is executed before installation/update obviously) . In this example, after installation or update of the game server, it will copy the file `linux64/steamclient.so` to `./steamclient.so` and generate a bash script in the file `avorion_ogpstarter.sh` and then make this bash script executable. Note that the bash script must be written to be properly interpreted by the XML (see the `"` character replaced by `&apos;` , `>` replaced by `&gt;` , and so on.. if you write plain bash script it will break the XML in Panel, so keep that in mind and replace special characters with their XML readable counterpart.

## Windows:

### Pre-Start Commands:

Comes after the `<server_params>` element. There can only be one `<pre_start>` element. It can run multiple lines of script that will be executed by the cmd batch environment.

This can be filled with lines of script that will be run in a batch script just before the game server is started. The script will always change directory into the home directory before your commands will run, so you can reference things locally.

```
<pre_start></pre_start>
```

### Environment Variables:

Comes after `<pre_start>` element. There can only be one `<environment_variables>` element. It can contain multiple entries one per line.

```
<environment_variables>  
</environment_variables>
```

Used for setting environment variables which may be needed by some servers. This is run in the batch environment, so please make sure you're using Windows commands for your environment SETS.

Special entries:

{OGP\_HOME\_DIR} will be replaced by the home directory for the game server.

## Linux:

### Pre-Start Commands:

Comes after the `<server_params>` element. There can only be one `<pre_start>` element. It can run multiple lines of script that will be executed by the bash shell.

This can be filled with lines of script that will be run in a bash script just before the game server is started. You do NOT need to provide the shebang `"#!/bin/bash"` in your commands. The script will always change directory into the home directory before your commands will run, so you can reference things locally.

```
<pre_start></pre_start>
```

Example (writes hiya to a file named testingPreStart in the home directory of the server):

```
<pre_start>
    echo "hiya" >> testingPreStart
</pre_start>
```

## Environment Variables:

Comes after `<pre_start>` element. There can only be one `<environment_variables>` element. It can contain multiple entries one per line.

```
<environment_variables>
</environment_variables>
```

Used for setting environment variables which may be needed by some servers such as Rust.

Example:

```
<environment_variables>
export LD_LIBRARY_PATH="{OGP_HOME_DIR}/RustDedicated_Data/Plugins/x86_64"
</environment_variables>
```

Special entries:

`{OGP_HOME_DIR}` will be replaced by the home directory for the game server.

## Linux and Windows:

### Locking / Protecting Additional Files:

Comes after `<environment_variables>` element. There can only be one `<lock_files>` element. It can contain multiple entries one per line.

```
<lock_files>
</lock_files>
```

Used for protecting additional game server files by using the system program `chattr`. This is a Linux only element. You can use relative (paths are relative to the home directory for the game server) or absolute paths.

Example:

```
<lock_files>
bin/AvorionServer
</lock_files>
```

The above example adds chattr +i to the bin/AvorionServer executable. This prevents the user from changing it. When SteamCMD or Rsync is run to update the files, everything is unlocked, but the files specified here will be locked post update / install. These files are also locked again (just to make sure) before the server is restarted and started. Entries here will not BREAK or AFFECT the steamcmd or Rsync update process.

Special entries:

{OGP\_HOME\_DIR} will be replaced by the home directory for the game server. However, if you're using this variable, you should just use a local path.

## Configuration Files:

Comes after <lock\_files> element. There can only be one <configuration\_files> element. There can be multiple <file> entries within <configuration\_files> element, the description in the <file> element is not mandatory. Example:

```
<configuration_files>
  <file description="The Main config file">Path/To/Config/File.ini</file>
  <file description="Another Config file">FolderX/GameEngine.ini</file>
  <file>FolderZ/GameConfig.ini</file>
</configuration_files>
```

It defines the files directly editable from Panel even without access to the server files directly. This requires the EditConfigFiles extra module.

► Pages 3

## Navigation

### Home

- [Easy Installation \(recommended\)](#)
- [Manual Agent Linux Installation](#)
- [Manual Web Linux Installation](#)



## Running Windows Game Servers on Linux using WINE

### XML Notes

- [Game Config](#)
- [Game Key](#)
- [Query Protocol](#)
- [LGSL Query Name](#)
- [GameQ Query Name](#)
- [Installer](#)
- [Game Name](#)
- [Server Executable Name](#)
- [Query Port](#)
- [CLI Template](#)
- [CLI Parameters](#)
- [Reserve Ports](#)
- [CLI Allowed Characters](#)
- [Maps Location](#)
- [Map List](#)
- [Console Log](#)
- [Executable Location](#)
- [Max User Amount](#)
- [Control Protocol](#)
- [Mods](#)
- [Replace Texts](#)
- [Server Params](#)
- [Custom Fields](#)
- [List Players Command](#)
- [Player Info Regex](#)
- [Player Info](#)
- [Player Commands](#)
- [Pre Install](#)
- [Post Install](#)
- [Pre Start Commands \(Windows\)](#)
- [Environment Variables \(Windows\)](#)
- [Pre Start Commands \(Linux\)](#)
- [Environment Variables \(Linux\)](#)
- [Locking / Protecting Additional Files](#)
- [Configuration Files](#)

Clone this wiki locally

<https://github.com/OpenGamePanel/OGP-Website.wiki.git>

