Overview

In this section you can find the commands you need to execute to build all your assets for your preferred platform.

The following table contains the description of the placeholder used inside the example commands to describe the different directories needed to complete the process.

Chapters:

Placeholder name	Description
<engineRoot>	The folder where the Tools/rc/rc.exe you need to use is located. Usually it is the root folder of the build you want to compile the assets for.
<assetPath>	The older where the assets used inside the game are stored.
<gameDir>	The folder name for the game directory (usually "Game").
<workPath>	The root where the compilation process will be executed.

RC Jobs

An "**RC Job**" is a set of operations the RC needs to execute to perform some tasks (in our specific case it describe the asset compilation process).

Every "**RC Job**" is defined as an xml file.

RC Job file - Main structure

There is an example jobfile attached to this page ("rcjob_all.xml"), which is usually sufficient to start building your assets. The description of the jobs need to included in-between the main tag:

```
<RCJobs>

[...]

</RCJobs>
```

Inside the RCJobs you first can specify some default properties (none of the property is mandatory, they just make easier to maintain the rc job file):

```
  <DefaultProperties
    game="Game"
    engine="Engine"
        languages="Localization"
    src="."
    trg="TempRC\${p}"
    pak_root="OutRC\${p}"
    enable_cuda="true"
        />
```

Property name	Property meaning
game	This is the name of the game folder where the rc.exe can find your game assets.
engine	This is the name of the folder where the shaders and the configuration files (cfg) for the engine are.
language	This is the name of the localization folder.
src	This is the name of the root folder where the previous directories are.
trg	This is the name of the folder where the compiled assets are temporary outputted. the ${p} symbols is the platform name passed as a parameter to the rc.exe
pak_root	This is the name of the folder where the RC will output the pak files.

You can also define some properties:

```
  <Properties
    xml_types="*.animevents;*.cdf;*.chrparams;*.dlg;*.ent;*.fsq;*.fxl;*.ik;*.lmg;*.mtl;*.setup;*.xml;*.node;
*.veg"
    non_xml_types="*.ag;*.gfx;*.png;*.usm;*.fev;*.fsb;*.fdp;*.sfk;*.ogg;*.txt;*.anm;*.cal;*.grd;*.grp;*.cfg;
*.csv;*.lua;*.dat;*.ini;*.xls;*.as;*.lut;*.mp2;*.mp3;*.xma"

    src_game="${src}\${game}"
    src_engine="${src}\${engine}"
        src_languages="${src}\${languages}"
    trg_game="${trg}\${game}"
    trg_engine="${trg}\${engine}"
        trg_languages="${trg}\${languages}"
    pak_game="${pak_root}\${game}"
    pak_engine="${pak_root}\${engine}"
        />
```

These are describing, for example, which file types are or are not xml files. Note that you can set new properties or override the default properties from the command line with the **myVarName=myValue** syntax.

From now on you can define your own jobs the rc will perform. In our case we use 4 jobs:

1. The conversion job, defined in-between the tag **<ConvertJob> [...] </ConvertJob>**
2. The job to perform some extra copy operations defined in-between the tag **<CopyJob> [...] </CopyJob>**
3. The packing job defined in-between the tag **<PakJob> [...] </PakJob>**
4. The cleaning job that cleans the target folder defined in-between the tag **<CleanJob> [...] </CleanJob>**

The job name is not fixed, so you can use the tag name you prefer. You can also define more jobs that what you are actually using (for test purpose for example). To define which job needs to be performed by RC, you need to use the <Run> tag at the end of the file as follows:

```
  <Run Job="ConvertJob"/>
  <Run Job="CopyJob"/>
  <Run Job="PakJob"/>
  <Run Job="CleanJob"/>
```

You can also, for example, create a job that is actually running other job, as follows:

```
  <CompileAssetsJob>
    <Run Job="ConvertJob"/>
    <Run Job="CopyJob"/>
    <Run Job="CleanJob"/>
  </CompileAssetsJob>
```

How to define a specific operation in one job

Inside each job (ConvertJob, CopyJob, ...) you need to describe which operations needs to be executed. To obtain this goal you can use the tag **<Job>** . This tag has different parameters that gives RC the exact information about the task he needs to perform. You can read some examples in the following paragraphs but a full list of possible operations is contained in the RCJob attached to this page.

Converting a tif

```
<if p="PC">
    <Job sourceroot="${src_game}"    targetroot="${trg_game}"   input="*.tif"
imagecompressor="${imagecompressor}" />
</if>
<ifnot p="PC">
    <Job sourceroot="${src_game}"    targetroot="${trg_game}"   input="*.tif"
imagecompressor="${imagecompressor}" streaming="1" />
</ifnot>
```

The first command is used for the compilation of .tif files for the PC platform while the second one is used for all other platforms.

Sourceroot is the folder of the source files, targetroot is the folder where the rc must output the results, input is the kind of file that needs to be processed, imagecompressor tells the rc that the operation needs the specific image compressor defined into the ${imagecompressor} variable, and streaming define if the output files needs to be outputted into the streaming folder. As you can see from this example, to select specific operation for a specific platform the <if> or <ifnot> tag can be used. The p variable contains the string that defines the platform the rc.exe is run for.

Copying a specific file from one folder to another

```
<Job sourceroot="${src_game}"   targetroot="${trg_game}"   input="*.chr" copyonly="1"/>
```

The previous command copies from sourceroot into targetroot all the chr files.

Packing the Scripts.pak

```
<Job sourceroot="${trg_game}" input="Scripts\*.*"     zip="${pak_game}\Scripts.pak" />
```

The previous command paks all the files contained into the *{sourceroot}\Scripts* folder into the Scripts.pak. The pak files is outputted into the ${pak_game} folder.

Cleaning a specific folder

```
 <Job input="" targetroot="${trg}" clean_targetroot="1" />
```

The previous command clean the targetroot specified.

Description of the steps needed to get the assets compiled for a specific target console

The compilation process for a specific target console could follow different steps. This is the flow we suggest to approach:

1. Copy all the assets you need for your game into a target folder <workPath>\<gameDir>\ .
2. Execute the RC Job you created for your project for a specific platform.

Platform	Example of the command needed to be executed to run an rc job
PC	Tools\rc\rc.exe /job=Tools\rc\RcJob_Build_SDK.xml /p=pc /threads=8 > pclog.log
Xbox One	Tools\rc\rc.exe /job=Tools\rc\RcJob_Build_SDK.xml /p=xb1 /threads=8 > xb1log.log
PlayStation 4	Tools\rc\rc.exe /job=Tools\rc\RcJob_Build_SDK.xml /p=ps4 /threads=8 > ps4log.log