Dead Cells Mods documentation

Document purpose

This document explains in detail how the game data workflow works, how to manipulate its data, create scripts for it and upload mods to the Steam workshop. It's targeted to people who want to create mods and it explains everything from scratch, step by step. We'll make the difference between data modifications and scripting. If you already played with the pak and cdb there is probably a lot in here that you already know.

General presentation

There is a major difference between the flow for altering data and creating some scripts. The process of creating scripts for a mod is more straightforward but requires more technical skills.

Brief presentation of tech and production flows to modify data

The game uses a big file to store its data: the PAK. If you browse your game files, you can see it as "res.pak" at the root of the installation directory of Dead Cells. It embeds all files used by the game.

To modify any data in the game you need to be able to open and extract every file from this pak. It's one of the purposes of PAKTool.

With mods we introduced the possibility, for what we call a "mod pak", to override data present in the original pak. This mod pak will be loaded during runtime and each file it contains will override the original ones. There are few exceptions with the CDB file and room edition with Tiled, for more detailed information report to the CDBTool & Rooms.

So, to create a mod, you have to extract files from the res.pak, modify the extracted files and then rebuild a mod pak. This mod pak will be created with the PAKTool and contain only files you modified from the original res.pak (to be as light as possible).

To run the tools, you need to install (or already have installed the .NET Framework version 4.5.2 or above. You can find it on Microsoft site here : https://www.microsoft.com/fr-fr/download/details.aspx?id=42642

Brief presentation of scripting flow

The game now supports scripting for creating level structures, defining level info and mob roster in a level. The flow is very simple, when a mod is activated, it checks for the presence of a directory `./Script/Struct/` in the mod's directory and tries to load scripts present in it. It's compatible with some res.pak modifications (you can change the res.pak and have scripts in the same mod).

The types of files

res.pak

As stated before, the res.pak is a container for all game files. It's in the installation root directory. It can be expanded using the PAKTool.

data.cdb

Data.cdb is a file containing a lot of different things concerning both gameplay and cosmetics. It's a big JSON file and can be edited with CastleDB. It contains rooms layouts, weapon parameters, fogs color etc...

.atlas & .png

In Dead Cells, all textures are PNGs and most of them are atlases. An atlas is multiple PNGs merged in one big PNG. In Dead Cells there are (few) standalone PNGs and .png/.atlas pairs. .atlas files contain information on where and how big a given image is stored in the .png. The AtlasTool is here to expand atlases in several "original files" to allow you to modify them and then rebuild the atlas from expanded files.

.hx

They are script code files. They are loaded and interpreted by the game during different steps like: creating the structure of a level, setting level parameters, generating mob roster, spawning mobs etc.

.tmx

Those files are located in tiled/tmx in the expanded pak folder. They contain the data of the rooms in the Tiled tool format. Be careful, the tmx stored in the pak and used by the game is actually a binary file, that can't be opened with Tiled. In order to edit them, you will have to export them with TmxTool.

How mods will behave in the game

The game supports multiple mods activated at the same time. In some cases, it will consider that two mods are incompatible with each other and prevent from activating the second one. In that case a message will warn the user on why the mod cannot be activated.

Two mods will be considered incompatible if one (or more) of this proposition is true:

- Both mods contain scripts
- Both mods modify the same file (except for the CDB)
- Both mods modify the same line of the CDB

A mod can only be activated when creating a new save or restarting a run from prison start.

For now, with each update (even minor ones) mods will be deactivated and current saves using mods will be reset (only the run, not meta items). This is to prevent the game from crashing and/or corrupt the saves.

What you can do with mods (current state)

This list is non-exhaustive

- Change some atlases (or part of atlases) (see list of unalterable atlases)
- Add floor junks
- Modify parameters in the CDB (except for table "Truelle")
- Create script to define a customized level structure, world graph, mob roster

What you cannot do with mods (current state)

All this may change with future releases and the list is non-exhaustive.

For now you cannot modify some atlases:

- atlas/ui.atlas
- atlas/fxCommon.atlas
- atlas/ui.atlas
- atlas/fxCommon.atlas
- atlas/fxEnemy.atlas
- atlas/fxWeapon.atlas
- atlas/fxDisplace.atlas

- atlas/gameElements.atlas
- atlas/lore.atlas

For caching reasons, we cannot reload those during runtime and modifying them will result in a no-op

You cannot modify the table "Truelle" in data.cdb. It will be a no-op too.

Some parameters in the CDB may have no effect when changed, but there is no exhaustive list at this point.

Removing lines of separators in the CDB may cause the game to crash

Removing entries from atlases may cause the game to crash or graphic bugs (depending on which atlas is concerned)

Generally removing files or part of files will cause the game to crash at one point or another.

Language should be added or altered via the existing process:
https://steamcommunity.com/games/588650/announcements/detail/1253537422578152950

Tools list

Except for castleDB, the tools presented are command line tools.

They accept - and / at the start of arguments. Argument names are NOT case sensitive.

Order of arguments is not important (as long as you respect the order -argument <parameter>):

Example:

```
-EXPAND -OUTDIR "C:\Tests\ExpandedAtlas\kingsGuardian\" -ATLAS
"C:\Tests\Expanded\atlas\kingsGuardian.atlas"
```

is equivalent to

```
-ATLAS "C:\Tests\Expanded\atlas\kingsGuardian.atlas" -OUTDIR "C:\Tests\ExpandedAtlas\kingsGuardian\" -
EXPAND
```

To run the tools you need to install (or already have installed) the .NET Framework version 4.5.2 or above. You can find it on Microsoft site here: https://www.microsoft.com/fr-fr/download/details.aspx?id=42642.

PAKTool

What is it for?
The pak tool is used to manipulate the .pak files. With it you can:

- Unpack all file data it contains into a directory, so you can visualize them and alter them
- Repack a full directory tree into a pak
- Create a diff pak against a reference pak containing only the files you changed and the lines of the CDB you altered. (creates a mod pak)

How do I use it?
```
-?: Display this help

-Expand -outdir <output directory> -refpak <input pak path> [-s]: Expands a given PAK to a file tree

-Collapse -indir <input directory> -outpak <output pak path> [-s]: Collapse a given file tree to a pak
```

`-CreateDiffPak -refpak <input pak path> -indir <input directory> -outPak <output pak path> [-s]: Create a pak from a directory with only what has changed or been added from the ref pak (typically for mods)`

arguments:

`-s/-silent: Do not display message error (deactivated by default)`

AtlasTool

Be aware that Atlas tool is a 64bits executable and will not run on 32bits systems. Also, commands ExpandAll and CollapseAll may consume a few GB of RAM (up to 3.5GB)

What is it for?

This tool is used to:

- create an atlas from a directory tree containing PNGs
- create a directory tree with all PNGs from an atlas

How do I use it?

`-?: Display this help`

`-Expand -outdir <output directory> -Atlas <input atlas path> [-s]: Expands a given Atlas to a file tree`

`-ExpandAll -indir <input atlases directory> -outdir <output directory> [-s]: Expands every atlas found in indir into outdir`

`-Collapse -indir <input directory> -Atlas <output atlas path> [-s][-ascii]: Collapse a given file tree to an atlas`

`-CollapseAll -indir <input directories> -outdir <output atlases path> [-s][-ascii]: Collapse every directory in the input directory into atlases`

arguments:

`-s/-silent: Do not display message error (deactivated by default)`

`-ascii: Export atlases as ascii (binary by default, not supported by Dead Cells)`

Example:

`AtlasTool -EXPAND -OUTDIR "C:\Tests\ExpandedAtlas\kingsGuardian\" -ATLAS "C:\Tests\Expanded\atlas\kingsGuardian.atlas"`

CDBTool

What is it for?

It's used to manipulate the CDB. You can:

- Create a directory tree where every table is a directory and every line in this table is a json file in the directory. It will also create files for the structure and property of the table.
- Create a CDB from a directory tree where every table is a directory and every line in this table is a json file in the tree.
- Create a directory tree with only altered and added lines/tables from a reference CDB and a "expanded" CDB directory tree.

Unless you want to edit the CDB with notepad line by line (which could be a valid option) or just want to browse it more easily than in CastleDB. You will probably won't need it.

How do I use it?

`-EXPAND -OUTDIR C:\Tests\ExpandedCDB -REFCDB C:\Tests\PAKTool\Expanded\data.cdb"`

```
-CREATEDIFFCDB "-INDIR" "C:\Tests\ExpandedCDB" -OUTDIR "C:\Tests\Diff" -REFCDB
"<SteamInstallPath>\steamapps\common\Dead Cells\res.pak"

-COLLAPSE -INDIR " C:\Tests\ExpandedCDB" -OUTCDB "C:\Tests\Expanded\data.cdb"
```

ScriptTool

What is it for?

It's used to create a bootstrap script file to start using script in a mod. It also includes a lot of comments and direction on how to use different structures and functions.

If you launch the tool without arguments, it can also be used to visualize and help create script code you can copy and paste with an UI but it's very basic at this moment.

How do I use it?

```
-CommandLine -NewFile -OutScript <output file name> [-s] : create a script file with default function
and basic doc inside.
```

arguments:

```
-s/-silent: Do not display message error (deactivated by default)
```

Example:

```
ScriptTool.exe -commandline -newfile -outscript "C:\Program Files
(x86)\Steam\steamapps\workshop\content\588650\1405810340\Scripts\Struct\test.hx"
```

Be careful that it will overwrite any existing file without warning.

CastleDB

What is it for?

Edit the cdb file and edit rooms

Where can I find it?

http://castledb.org/

How do I use it?

1) Download the .zip at http://castledb.org/
2) Replace the castle.js in the castled you unzipped by the one found in `<Steam installation Dir>\steamapps\common\Dead Cells\ModTools\CastleDB (C:\Program Files (x86)\Steam\steamapps\common\Dead Cells\ModTools\CastleDB by default)`

You can add rooms :

From version 2.4.0 you can now use Tiled.
<mark>Warning :</mark> Tiled is an external tool that operates only on tmx files, while the former RoomEditor used to directly modify the CDB. Thus, some modifications that were previously made through RoomEditor now have to be done manually with CastleDB, including most notably the rooms type and flags.
Read the Full Process in order to fully understand the new method.

TmxTool

What is it for?

The tmx files you find in tiled/tmx in your expanded pack are actually binary files generated from Tiled tmx format, which contains xml data. TmxTool is used to make the conversion between those two formats.

How do I use it?

The tool carries some dlls with it, so copying it elsewhere more convenient is not the best way of using it. You should either create a shortcut to the tool at the place you want to call it from, or you can just type the full path to it in cmd/powershell.

The command you need to do the conversion is :

```
TmxTool <-Expand|-Collapse> [-TmxBin <path to tiled folder containing binaries>] [-TmxXml <path to folder containing xml>]
```
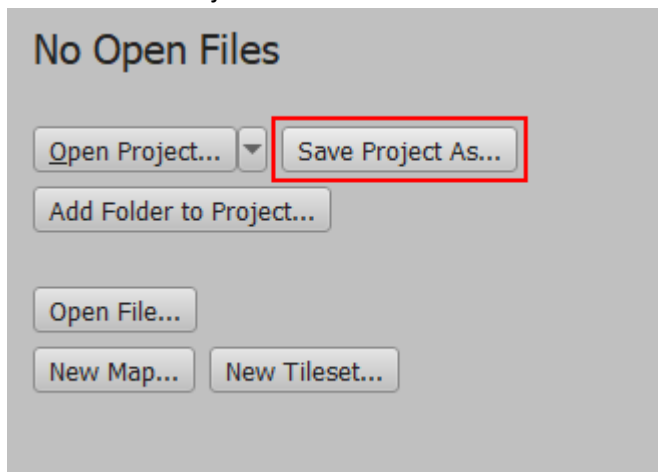
You can also print the help using

```
TmxTool -help
```

Tiled

What is it for?
Edit rooms data

Where can I find it?
The version of Tiled currently supported by TmxTool is the 1.4 version. Newer versions might crash.

https://www.mapeditor.org/

How do I use it?
1) Launch Tiled
2) Create a new Project

3) Add the tiled folder (exported with TmxTool) to the project
4) Go to Project > Project Properties
   a) Set directory to tiled folder
   b) Select "objecttypes.json" for Object types
5) Create or modify any room

Full Process

Unpakking data
First, we have to locate the original res.pak used by the game. You can find it in the installation directory root: <Steam installation directory>\steamapps\common\Dead Cells (C:\Program Files (x86)\Steam\steamapps\common\Dead Cells by default).

Open a new command line window. Move current directory to where your PALTool.exe is located. Extract the res.pak you found previously in an empty directory (you must have writing rights in this directory) by typing the following:

```
PAKTool -Expand -OutDir C:\Tests\Expanded -RefPak "C:\Program Files (x86)\Steam\steamapps\common\Dead Cells\res.pak"
```

Once executed, we have our directory filled with the data files contained in the res.pak :

Modifying data

The CDB

You can now see the CDB file, called data.cdb at the root of the expanded directory. You can modify it either by breaking it down with the CDBTool or with CastleDB.

Breaking down the CDB

You Can break down the CDB into a directory tree containing json files for each line with the CDBTool. Here, we break down the one we extracted previously from the res.pak.

```
CDBTool -Expand -OutDir C:\Tests\ExpandedCDB\ -RefCDB " C:\Tests\Expanded\data.cdb"
```


After that a directory tree is created where each table in the data.cdb is a directory and each line in each table is a json file. Also special files __PROPS__.json and __STRUCTURE__.json are created for each table. It's strongly advised NOT to modify them as you may break the game or render your mod incompatible with the game/other mods very easily by doing so.

OS (C:) › Tests › ExpandedCDB › biome

Nom	Modifié le	Type	Taille
__PROPS__.json	25/06/2018 12:22	Fichier JSON	1 Ko
__STRUCTURE__.json	25/06/2018 12:22	Fichier JSON	4 Ko
AncientTemple.json	25/06/2018 12:22	Fichier JSON	2 Ko
BeholderPit.json	25/06/2018 12:22	Fichier JSON	3 Ko
Bridge.json	25/06/2018 12:22	Fichier JSON	2 Ko
BridgeBoatDock.json	25/06/2018 12:22	Fichier JSON	2 Ko
Castle.json	25/06/2018 12:22	Fichier JSON	3 Ko
CastleAlchemy.json	25/06/2018 12:22	Fichier JSON	3 Ko
CastleTorture.json	25/06/2018 12:22	Fichier JSON	3 Ko
CastleVegan.json	25/06/2018 12:22	Fichier JSON	3 Ko
Cavern.json	25/06/2018 12:22	Fichier JSON	2 Ko
Cemetery.json	25/06/2018 12:22	Fichier JSON	3 Ko
CemeteryInt.json	25/06/2018 12:22	Fichier JSON	3 Ko
ClockTower.json	25/06/2018 12:22	Fichier JSON	3 Ko
Crypt.json	25/06/2018 12:22	Fichier JSON	2 Ko
Ossuary.json	25/06/2018 12:22	Fichier JSON	2 Ko
PrisonCorrupt.json	25/06/2018 12:22	Fichier JSON	2 Ko
PrisonCourtyard.json	25/06/2018 12:22	Fichier JSON	3 Ko
PrisonCourtyard2.json	25/06/2018 12:22	Fichier JSON	3 Ko
PrisonDepths.json	25/06/2018 12:22	Fichier JSON	2 Ko
PrisonHub.json	25/06/2018 12:22	Fichier JSON	2 Ko
PrisonRoof.json	25/06/2018 12:22	Fichier JSON	3 Ko
PrisonRoofCorrupt.json	25/06/2018 12:22	Fichier JSON	2 Ko
PrisonStart.json	25/06/2018 12:22	Fichier JSON	2 Ko
SecretRooms.json	25/06/2018 12:22	Fichier JSON	2 Ko
Sewer.json	25/06/2018 12:22	Fichier JSON	2 Ko
SewerOld.json	25/06/2018 12:22	Fichier JSON	2 Ko
StiltVillage.json	25/06/2018 12:22	Fichier JSON	3 Ko
StiltVillageInt.json	25/06/2018 12:22	Fichier JSON	3 Ko
Throne.json	25/06/2018 12:22	Fichier JSON	2 Ko
TopClockTower.json	25/06/2018 12:22	Fichier JSON	2 Ko

Rebuilding the CDB

After altering or adding files to this directory tree, you'll have to rebuild the CDB and put it back with the other files you are editing before creating the final mod.

To do so you have to call CDBTool with some other arguments:

```
CDBTool -Collapse -InDir C:\Tests\ExpandedCDB\ -OutCDB "C:\Tests\Expanded\data.cdb"
```

Invite de commandes — □ ×

```
C:\DeadCells\ModTools>CDBTool -Collapse -InDir C:\Tests\ExpandedCDB\ -OutCDB "C:\Tests\Expanded\data.cdb"
```

Editing the CDB with CastleDB

You can edit in place your data.cdb using CastleDB.

id	atlasName	ambient	ambientScale	cellShadow	smoke	water	cloud	waterLight	fog	fogScale	vegetation	floorStamps	floorJunkDensity	wallJunkDensity	layers	lightColors	smokeShader	camEffects	scatterConf	vegetationRandScale	oneWayOpaque	reverbKind	reverbMix	uiProps
0	PrisonHub	prison								0.7	0.4	0.7	0	0	[BackgroundF [Hero,, ,], [			camFogTopAlp camFogBotAlph lensDustBigAlp lensDustSmallA		Y	N	CASTLE_MED	0.15	alphaBlack 0 alphaBlue 0 alphaGradientE
1	PrisonStart	prison								0.75	0.15	0.7	0.6	0.4	[BackgroundF [Hero,, ,], [I		mode P alpha 0 contrib 1 power 0 scale 0 speed 0	camFogTopAlp lensDustBigAlp lensDustSmallA		Y	N	CASTLE_SMA	0.15	alphaBlack 0 alphaBlue 0 alphaGradientE
2	PrisonDepths	prison								0.8	0.1	0.8	0.6	0.4	[BackgroundF [Hero,, ,], [		mode P alpha 0 contrib 1 power 0 scale 0 speed 0	camFogTopAlp lensDustBigAlp lensDustSmallA		Y	N	CASTLE_LAR	0.15	alphaBlack 0 alphaBlue 0 alphaGradientE
3	PrisonRoof	prisonRoof								0.6	0.2	0.7	0.6	0.4	[BackgroundF [Hero,, ,], [			camFogTopAlp camFogBotAlph lensDustBigAlp lensDustBigCol lensDustSmallA camFogTopCol camFogBotCol	Sun	Y	N	OUTDOORS_(	0.15	alphaBlack 0 alphaGradientE
4	Sewer	sewer								0.7	0	0.7	0.6	0.4	[BackgroundF [Hero,, ,], [		mode P alpha 0 contrib 1 power 0 scale 0 speed 0	camFogBotAlp camFogTopAlp lensDustBigAlp lensDustSmallA		Y	Y	PIPE_SMALL	0.1	alphaBlack 0 alphaBlue 0 alphaGradientE
5	SewerOld	sewerOld								0.7	0.25	0.7	0.6	0.4	[BackgroundF [Hero,, ,], [		mode A alpha 0 contrib 1 power 0 scale 0 speed 0	camFogTopAlp camFogBotAlph lensDustBigAlp lensDustSmallA		N	Y	PIPE_LARGE	0.1	alphaBlack 0 alphaBlue 0 alphaGradientE
6	Bridge	bridge								0	0.3	0.7	0	0	[BackgroundF [Hero,, ,]			camFogTopAlp camFogBotAlp lensDustBigAlp lensDustSmallA	Moon	Y	N	CASTLE_COL	0.15	alphaBlack 0 alphaBlue 0 alphaGradientE
7	BridgeBoatDo	bridge								0	0.3	0.7	0	0	[BackgroundF [Hero,, ,]					Y	N	CASTLE_COL	0.15	alphaBlack 0 alphaBlue 0 alphaGradientE
8	Ossuary	ossuary								0.3	0	0.7	0.6	0.25	[BackgroundF [Hero,, ,], [			camFogTopAlp camFogBotAlp lensDustBigAlp lensDustSmallA		Y	N	MOOD_HELL	0.15	alphaBlack 0 alphaBlue 0 alphaGradientE
9	Cavern	cemetery								0.3	0	0.7	0.6	0.5	[BackgroundF [Hero,, ,], [		mode C alpha 0 contrib 3 power 0 scale 0 speed 0	camFogBotAlp lensDustSmallA		Y	N	MOOD_HELL	0.15	alphaBlack 0 alphaBlue 0 alphaGradientE
10	SecretRooms	secretRooms								0.65	0	1	0.75	0.5	[BackgroundF [Hero,, ,], [			camFogTopAlp camFogBotAlp lensDustBigAlp lensDustSmallA		Y	N	WOODEN_SM	0.15	alphaBlack 0 alphaGradientE
11	StiltVillage	stiltVillage								0.4	0	1	0.8	0.5	[BackgroundF [Hero,, ,], [I		mode P alpha 0 contrib 1 power 0 scale 0 speed 0	camFogTopAlp camFogBotAlph lensDustBigAlp lensDustSmallA lensDustBigCol camFogTopCol	Sky	Y	N	WOODEN_CO	0.15	alphaBlack 0 alphaBlue 0 alphaGradientE

Atlases

As we saw before, most of Dead Cells graphic data are packed in atlases. To modify them you can either modify directly PNGs or use the AtlasTool to break down atlases and build them back. The first solution surely demands less manipulation but can be tricky if you want, for example, change only one animation on a mob as everything is shuffled.

When breaking down an atlas into separate files, there is the advantage that every file created has a meaningful name, allowing you to identify easily what you are looking for, and also grouping animation frames as they have the same name with a sequential number.

Let's work with an example: the base zombie, let's say you want to change its walk anim. All you have to do is:

Break down the atlas:

```
AtlasTool -Expand -Atlas "C:\Tests\Expanded\atlas\zombie.atlas" -OutDir "C:\Tests\ExpandedAtlas"
```


You can now easily identify frames of the walk animation:

For each frame there are actually 2 files: one representing the "diffuse" including "purple" zones which are interpreted as "glowing" in the game shader, one "normal map" used to give some "bumps" to models.

Be careful, you can add or alter files, it will impact the looks and timing of animations, but some animations (like attacks) rely on a given time. Also, you can see all files are ending with -=-xx-=- where xx is a number. Please do not remove or change them, they are additions to allow the AtlasTool to breakdown animation when every frame is named the same way.

Rooms

Generate tmx xml files

First you will need to generate the tmx xml files from the tmx binary files located in the expanded pak. You will need to use the TmxTool, that is located in a subfolder of your ModTools folder, with its dlls. You can call it either by typing the full path leading to it from the folder you use, or by creating a shortcut.

```
TmxTool -Expand -TmxBin <folder tiled in your expanded pak> -TmxXml <destination folder>
```

This should get you the tiled files (two pngs, two tsx, a json, and the tiled folder with converted tmx) in the destination folder.

Creating a new room

In order to create a new room, you have to add it in the CDB, and to create its data with Tiled (make sure to have it set up following the How do I use it ? section, especially for object types).

Create a room in CDB

Open the CDB with CastleDB and go to the rooms tab. Scroll down to where you want to add your room, and insert it.

Prison		
578	PrisonFlaskRoom	
579	PrisonArm	Move Up
580	PrisonTube	Move Down
581	PrisonTube	
582	PrisonTube	Insert
583	PrisonStar	Delete
584	PrisonMon	✓ Separator
585	GiantEscap	
586	GiantEscap	Show References
587	PrisonDepthsEntrance	

Notice that CastleDB will automatically update all the rooms id. Not using it would require you to update all those ids manually (in the file name and data in case of an expanded CDB).

Enter your room name, select its type, activate it, and set the flags to your liking.

	id	type	active	flags
Prison				
578	PrisonFlaskRoom	FlaskRoom	Y	NoFlip
579	MyAmazingRoom	Combat	Y	Ø
580	PrisonArmory	Special	Y	NoFlip
581	PrisonTube1	TubeEntrance	Y	NoFlip
582	PrisonTube2	TubeEntrance	Y	NoFlip
583	PrisonTube2NoSkel	TubeEntrance	Y	NoFlip
584	PrisonStartExit	Exit	Y	Ø
585	PrisonMonsterDoor	GenericZDoor	N	NoFlip

Create a new room in Tiled

Open Tiled, and create a new room. You can duplicate a room directly in your file explorer, or use File > New > New Map (in that case you will have to set the correct tile size and recreate the layers, so the duplicate method is recommended).

Make sure you use the exact same name for the tmx and for the CDB entrance.

Edit a room in Tiled

There are three layers and two tilesets. Make sure that you use the correct tileset with a layer.

The collision (col) and link (lnk) layers are edited in the same way.
Select a layer by clicking it in the layer list.
Select a tileset by clicking on it in the tileset list. You select a tile or a group of tiles by clicking on it / dragging them.
The toolbar at the top offers various editing tools, including selection tools and the eraser.
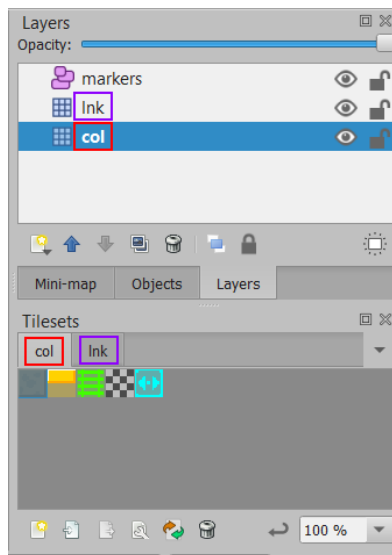
The markers layer is a bit different and has specific tools. It is recommended to enable the grid snapping in View > Snapping, in order to facilitate the placement of the markers.

Placing markers is done using the "Insert Rectangle" tool. You can also select an existing marker using the "Select Objects" tool.
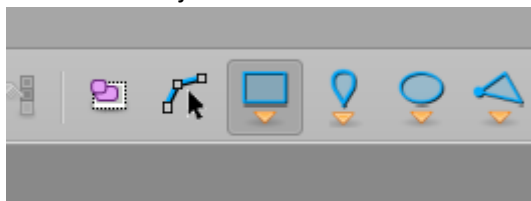
Click and drag to draw the area you want. You then have access to the property tab of the marker.
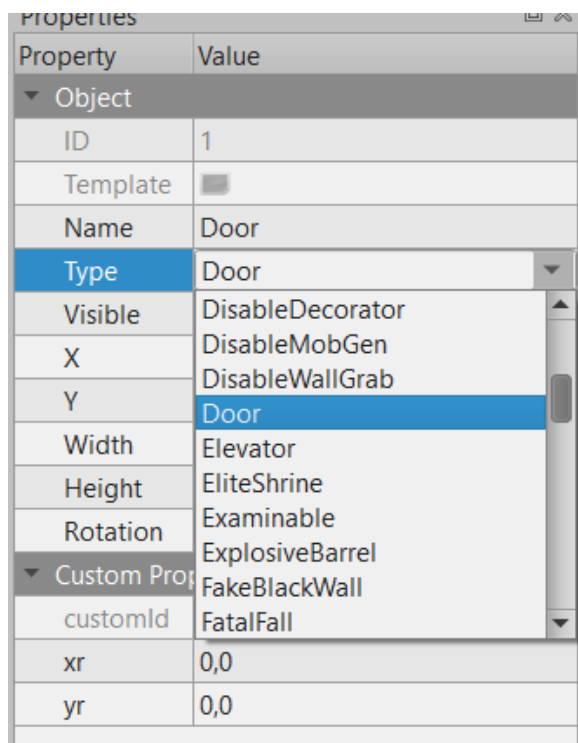
Set a convenient name, and choose a type from the drop down list. If the drop down list does not appear, please ensure you have set the Object Type file in the Project Properties, and then close and reopen the room file.

Generate the tmx binary files

Once you're done creating and editing rooms, you'll have to generate the tmx binary files back from the tmx xml files, again with TmxTool.

```
TmxTool -Collapse -TmxBin <destination folder : tiled/tmx in your expanded pak> -TmxXml <the folder you made your modifications into>
```

Repakking modified data

Once you've done every modification you want to, it's time to build a mod res.pak so you can test it inside the game and eventually upload it to steam.

To do so you have to use the PAKTool again and create a diff pak from the original pak :

```
PAKTool -CreateDiffPak -RefPak "C:\Program Files (x86)\Steam\steamapps\common\Dead Cells\res.pak" -InDir "C:\Tests\Expanded" -OutPak "C:\Program Files (x86)\Steam\steamapps\workshop\content\588650\1405810340\res.pak"
```


Before you do that, don't forget to rebuild your data.cdb from your working directory if you work with a broken down cdb, and to generate the cached room data if you modified any room with Tiled.

Now create a json file next to the res.pak created by the last step called settings.json and containing the following fields :

You can also add a preview.jpg or .png (max 1024x1024) to use as preview/thumbnail of your mod (like it's done for languages)

Be careful, the name cannot be changed once the mod has been uploaded. The visibility is optional (visible to all by default) and can take the values Private, Public, Friends (case is not important) which correspond to "visible only to you", "visible to all" and "visible to you and your friends".

The available categories for now are "Gameplay", "Cosmetic", "Test".

At this moment there is no restriction to the category name other than it should not be "Language" as it is reserved for localization mods.

Creating some scripts

We'll use the dummy mod to create our first script. It will only create a very simple level with an entrance, a combat room and an exit, with a mob roster taken from an existing level and parameters from the same existing level.

You will then be able to download some more advanced scripting tutorials via mods.

1) Install the dummy mod (points 1 and 2).
2) Execute the script tool with the following arguments in a command line window :
   `ScriptTool.exe -commandline -newfile -outscript "C:\Program Files (x86)\Steam\steamapps\workshop\content\588650\1405810340\Scripts\Struct\test.hx"`
3) Rename the test.hx in main.hx in
   `C:\Program Files (x86)\Steam\steamapps\workshop\content\588650\1405810340\Scripts\Struct\`
   This step can be avoided if you export a new file directly as main.hx, but beware that when doing so, if a modified main.hx already exists, it will be replaced without warning.
4) (optional) Launch the game and create a new run, you can see it uses a default struct only composed of one random entrance, a random combat room and a random exit that leads back to the start of the level.
5) You can now open the main.hx with your favorite text editor and see for yourself the code (and comments) used for this.

You will find some technical documentation on the script API in the `Scripts/TechnicalDocumentation` directory inside your ModTools directory (present in the install directory of Dead Cells). Or on https://dead-cells.com/mods

You can also find some samples on https://github.com/motion-twin/deadcells_mods_samples

Testing, creating and uploading the mod on Steam
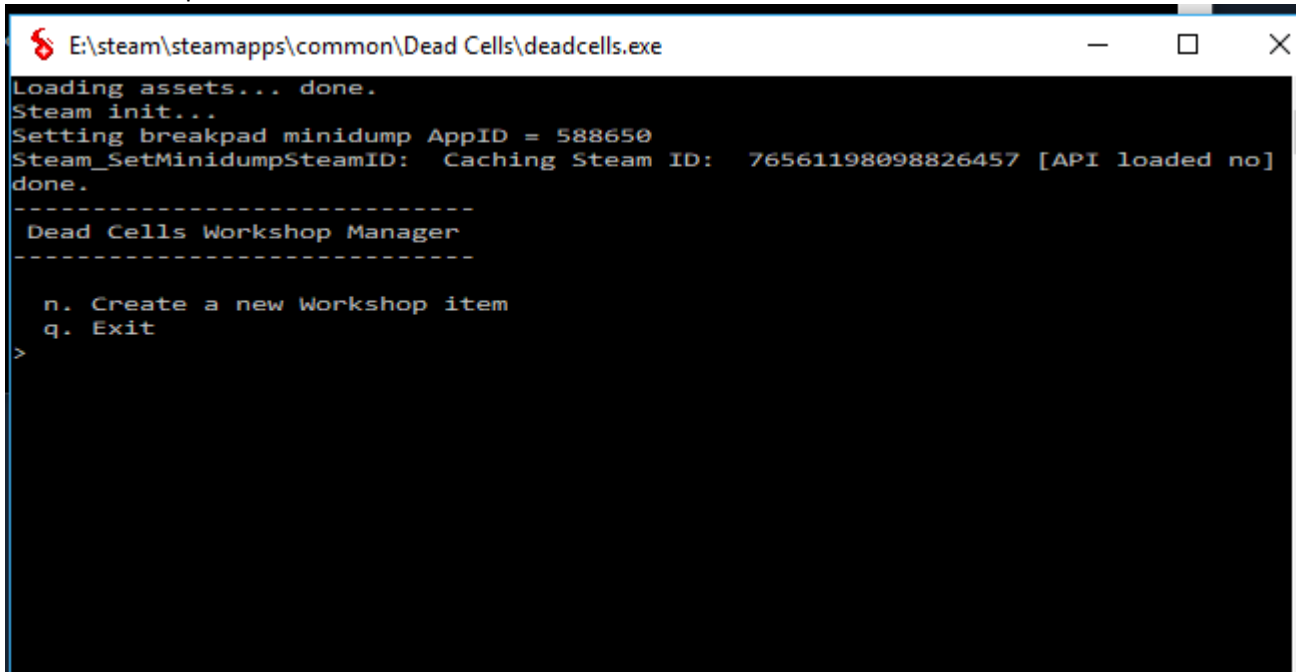
Testing

You can test your mod locally on your machine without having to upload it to the internet each time. To do this:

1. Go to https://steamcommunity.com/sharedfiles/filedetails/?id=2661514293 and download the Dummy Mod V2

2. Navigate to <Steam installation Directory> \steamapps\workshop\content\588650\1405810340 (by default C:\Program Files (x86)\Steam\steamapps\workshop\content\588650\1405810340)

3. Replace the res.pak in the directory by your own mod pak you created

4. Launch the game and start a new run, activate "Dummy Mod V2" and you can test your own mod
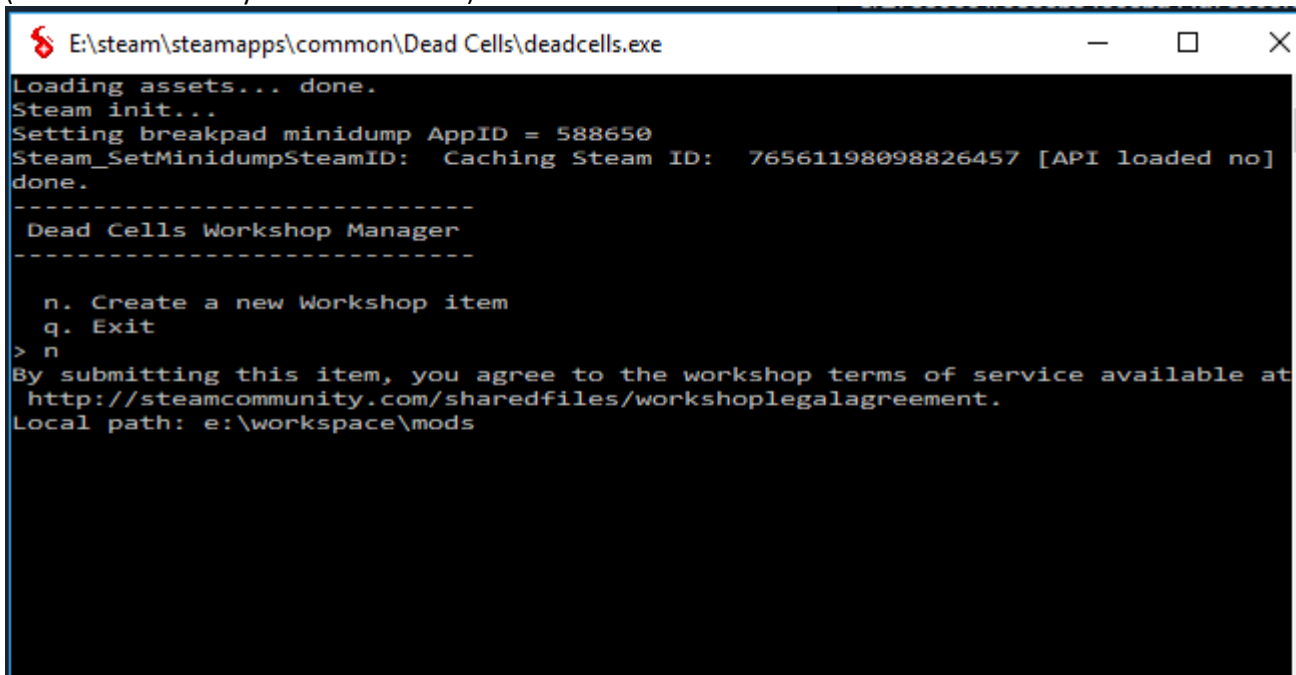
Create and Upload:

To upload your new mod to steam workshop and put it online you need to use the command line tool again.

1. Open the command line.
2. Navigate to the dead cells folder here: <DRIVE>:\steam\steamapps\common\Dead Cells
3. Type: deadcells.exe --workshop
4. Click on ok on the steam popup (and remember it might be hiding behind another window).
5. This will then open another command line window like this:

6. Choose "create a new workshop" item by typing 'n' and inputting the path to your mod (which is the folder you created earlier):

7. Then confirm the creation and upload by typing yes.

```
You are about to create a new mod with name: Awesome Mod This name cannot be changed later! Are you sure ? (yes/no):_
```

8. This will upload the file to your steam workshop and allow you to edit the workshop item through the steam interface.