

Section 1: X2 Modding

All content of the X2 modder pack is copyright 2004 by egosoft. You are granted the permission to use these tools only for the purpose of creating free non commercial X2 modifications. Tools and documentation © egosoft 2004.

1.1 INTRODUCTION

Modding in X2 consists of these main operations: Extracting files from the big catalogs, modifying or replacing single files in the data file tree, and assembling a "mod pack" from the modified files. The modified game can then be played by selecting a modified game from the start dialog and choosing the option Modified Game when starting a new game from the main menu.

1.2 INSTALLING AND RUNNING A MODIFIED GAME

A mod package will have been prepared by someone modifying X2. This mod package consists of two main files, a catalog file (.dat) and a catalog index file (.cat). Lets assume a mod called "x2cool", this would then be distributed as x2cool.cat and x2cool.dat. To play the mod it has to be copied into the "mods" folder under the x2 installation folder. When starting X2, the latest patch will display a new button on the setup dialog called "Select Mod Package". The button is only clickable if a mod package has been installed in the mods folder. After selecting the button a new dialog with a list of available mod packages is shown. Selecting a package and clicking OK will prepare the package to be used when starting a new game. The package can also be removed in this dialog by clicking on "Original".

The mod package author might also have provided new scripts or a new galaxy map. The scripts have to be copied to the "scripts" folder, a galaxy map has to be copied into the "maps" folder. If a new map has been provided, a new game using this new galaxy map can be started by choosing "New Game in Custom Galaxy". A game started in this way does not contain the main plot, since the map might look completely different and the plot needs certain key sectors to work.[^]

To make installation of mods as easy as possible we suggest to distribute your mods in the form of a ZIP file which contains the two files (cat and dat) inside a directory called "mods". This means the user just has to extract your archive to the X2 installation folder. Version 1.3 of X2 will automatically recognize the mods folder and activate the "Select Mod Package" button.

Understanding .cat .dat files

Catalogues are basically nothing more than archives of files. All the files that make up X2 are stored in such catalogues. Each archive / catalogue consists of two files. One called .cat and one called .dat. Currently (at version 1.3) the game consist of four catalogues named 01.dat/cat to 04.dat/cat

When playing, all catalogues are opened and files can be read from all. The higher numbered catalogues have priority over lower numbered catalogues. X2 version 1.0 used catalogues 01.dat only and later versions added further catalogues which partially introduced new files and partially replaced old ones.

A mod catalogue is a special case. It is read from a separate folder and has the highest priority. This means that all files which you put into your catalogues will definitely be used and can replace files of the same name in 01.dat to 04.dat

Running X2 in RAW mode during development

During development of your X2 modification, you can extract all catalogues and test X2 with extracted files (see below under tools for this procedure).

When you do this you want to make sure that you extract all cat/dat files in the right order: Extract 01.cat/dat first then extract 02.cat/dat and overwrite the extracted files, and so on. You would then move all cat/dat files away to make sure that you are indeed loading the extracted files.

For development it makes sense to not just extract files from the archive but also uncompress them. This is also done with the tools explained below.

The game will now have the choice between packed (.pck) files and raw (uncompressed) files (e.g *.txt files inside the types folder, *.xml files in the t folder etc.)

You can use the **+r** commandline parameter to force X2 to load only uncompressed files.

1.3 Modding - Which files can be modified

Suitable files for modding are ship and station parameters, meshes and textures, scripts, and of course the Galaxy map. ship and station parameters are stored in text files, which can be extracted from the catalog files with a tool (see below). The text file can then be edited and an updated version packed into a new catalog (the "mod pack"). Mesh data is stored in packed "body" files. Most of these can also be extracted by the tool into a human readable form. However, modifying the extracted text files is not suggested. We provide our 3dsmax exporter plugin to export mesh and animation data from 3dsmax, version 5. Scripts can just be added into the scripts folder in the X2 installation folder. The galaxy map, which is an XML file, can be edited from within X2 after mod mode has been turned on by typing "There shall be wings" in the main start menu of X2. As a general rule, files are always searched first in the mod package catalog, then in normal X2 catalogs and if they are not found in any catalog, the game tries loading them directly from the X2 installation folder and its subfolders. To ensure

the correct mod files are loaded, all modded text files have to be converted to binary files with "x2tool" and then packed together in a x2 mod pack catalog with "x2build".

The Scripteditor

The most interesting files for modding however are scripts. You can activate an editor inside the game with the same magic term that also activates the galaxy editor: Type "Thereshallbewings" while inside the game (type this slowly and note the capital "T").

Be aware that activating the script editor with this cheat will turn your universe into a "modified universe" (see below for details).

There is a separate documentation for the scripteditor called "***MSCI Documentation***".

Modified universe versus original game

Starting from version 1.3 X2 has two modes of running. It can be started either in modified or unmodified mode. An unmodified game is one which uses only the files distributed by egosoft without any self written scripts, modified types or universe data.

A modified universe game can be started "modified" from the start (e.g by using game mods) or a normal "real" game can be turned into a modified one (e.g if you activate the script editor and import unsigned scripts).

You may ask why this procedure is necessary?

The reason for the difference between unmodified games and modified games is that many people want to play X2 for an incredible long time. Part of the motivation for playing a game such a long time would be lost once they know that it is very easy to cheat. But in a modified universe, it is unfortunately very easy to cheat yourself everything. So with this approach we want to give you the best of both worlds. On the one hand you can create your own games and manipulate every aspect of X2, but on the other hand people playing an unmodified X2 can still tell the difference and can enjoy the cozy feeling that they are spending their time earning real X2 credits ;)

Signing your scripts for integration in non modified games

Not all scripts are cheats (far from it). So if you write scripts which you think should be integrated in the "normal" X2 universe too, please send them to us for signing. We want the X2 community to evaluate your scripts and if they agree that your script will be a good addition to the X universe and not spoil the game, we will sign your script, so that it can be used in the normal universe too.

Signed scripts can also be added in the form of upgrade products. We have reserved a set of products for this reason and will put player written scripts on sale in future versions of X2.

Section 2: THE TOOLS

2.1 x2tool

This tool is mainly used to decode and extract files from the catalogs shipped with X2. The tool is a commandline tool with various options (starting with '-'). X2tools has the following options to work on X2 datafiles:

`x2tool -extract-dat catalogname`

Unpack the catalog file *catalogname.dat* using the catalog index *catalogname.cat* to the current directory. Necessary subfolders are created as needed.

Example: `x2tool -extract-dat 01`

`x2tool -decrypt-cat catalogindexname textfile`

Decrypts the (binary) catalog index (.cat file) *catalogindexname* into *textfile* which is a human-readable version of the .cat file containing the names of all files which are stored in the catalog (.dat file). While this gives you the information which files are in the catalog, this step is not necessary to unpack the catalog. -extract-dat uses the original encrypted catalog index.

Example: `x2tool -decrypt-cat 01.cat 01index.txt`

`x2tool -unpack infile outfile`

Decrypt and uncompresses *infile* which can be packed body and scene files (.pbd), or packed text or xml files (.pck).

Examples:

`x2tool -unpack 00138.pbd 00138.bod`

```
x2tool -unpack maps\x2_universe.pck maps\x2_universe.xml
x2tool -unpack types\Ships.pck types\Ships.txt
```

`x2tool -crypt infile outfile`

Encrypts file *infile* which has been compressed with gzip into *outfile*. This is used to generate both .pbd out of .bod files and .pck out of .txt and .xml files.

Examples:

```
gzip <00138.bod >00138.bod.gz
x2tool -crypt 00138.bod.gz 00138.pbd
x2tool -crypt types\Ships.txt.gz types\Ships.pck
```

It is important that gzip is used exactly as shown here, that is using the input and output redirection brackets syntax. Just using *gzip filename* creates a file *filename.gz* which is **NOT** compatible with the unpack code used in X2 and x2tool.exe.

2.2 x2build

This tool is used to build catalog files from a folder tree using a resource file to select files to add or gathering all files from a folder and all of its subfolders. This does the reverse of x2tool -extract-dat. Depending on the number of arguments the mode is selected:

`x2build catalogname`

This builds the catalog files .cat and .dat by reading file names from a resource file. The .rct file is a text file containing one filename including path on each line. x2build parses this file, reads each file and adds it to *catalogname.dat*, the path and filename being added to the (encrypted) catalog index *catalogname.cat* .

Example: x2build 04

`x2build catalogname sourcerootfolder`

This builds the catalog files .cat and .dat by reading all files contained in the folder *sourcerootfolder* and its subfolders, adding them to the catalog *catalogname.dat* and adding the filename to the catalog index *catalogname.cat*. The path to the file relative to *sourcerootfolder* is stored together with the filename.

Example:

```
cd \games\x2\mods
x2build cool c:\x2modding\coolmod
```

where the folder *coolmod* contains all changed files

2.3 gzip

This GNU utility is used to pack single files before they are "encrypted" and bundled together in a catalog. It is used with body files (.bod), text files (.txt) and XML files (.xml). The resulting compressed file can then be encrypted with `x2tool -crypt` before adding it to the mod pack catalog with `x2build`. The best way to use `gzip` without destroying the original uncompressed file is

```
gzip < uncompressedfile > compressedfile
```

`gzip` is distributed under GPL (GNU public license). The (unmodified) distribution zipfile `gzip124xN.zip` in this modpack distribution contains instructions on how to obtain the `gzip` source code.

2.4 EGOxport.dle

This is a plugin for 3dsmax(5) to export meshes ("bodies") and "scenes" from 3dsmax into a special data format used by X2. There are now versions for both 3dsmax versions 5 and 6. The plugin for 3dsmax 6 now also supports the GNormal plugin

which allows you to use the same normalmaps both for X2 and test rendering inside 3dsmax 6.

(See the additional document [X2_GFX_Modding.pdf](#) for details)

Section 3: Galaxy Editing

The Galaxy editor is invoked by typing "**Thereshallbewings**" (note the capital T and please type slowly) while being in the X2 main start menu or on the START subpage. A new entry "Galaxy Editor" appears in the menu shown after "Start" is selected in the main menu.

Maps are loaded from the maps folder (either from inside the cat/dat files or from a real folder of that name).

Fileformat

The maps are stored in XML format. You can edit these files with every text editor and look at them with every XML interpreter (e.g Internet Explorer).

Many operations which are not possible with the ingame galaxy editor are very easy with a text editor once you understand the fileformat.

The XML file basically represents a hierarchy of instances of the objects defined in the types folder (see below). For example a factory is stored inside a sector and can hold various containers one of which is the ship container which holds defense ships and transporters working for that factory.

By editing the XML file with a text editor you can quickly copy a whole factory including all the things inside it (products, resource definitions, ships, shields etc.) or even copy a whole sector.

Default object behaviour

It is important to understand that not all objects need to be stored in the file. Some things are implicitly created on creation of an object if they are not defined in the map otherwise. For example if a ship does not contain any shields it will automatically receive default shields on creation. If however you add shields to a ship, it will have exactly those shields and no default configuration.

3.1 Guide to using the galaxy editor

The galaxy editor is controlled inside a normal game. You will find yourself inside a ship and can fly around a completely inactive universe.

The main functions of the galaxy editor are controlled from three menus:

Galaxymap:

You can jump to all sectors in your current map with the "g" key.

To create new sectors or copy whole sectors it is best to simply use a text editor and edit the XML file inside the maps folder.

Main menu -> DEBUG tab

From this tab you can create new objects, export the map and check the validation of gate connections (all gates must be connected in both ways)

When creating a new object you usually have to first select the subtype (e.g the ship type if the category of object selected was a ship). Second you select the owner (usually a race or the player. Note that just because a ship is typically used by one race, you can still assign it freely to everybody). Next you select what subcontainer to put the newly created object in (usually the current sector for a ship that you want to appear in the sector immediately, or inside a station if it should be assigned to a specific station). At the end you have to enter three numbers for the initial position of objects created inside the sector directly (x,y,z).

Sectormap:

Most editor functionality is controlled from the sectormap.

From here you can modify all objects placed inside a sector and change the whole look of the system.

Many of the below keys affect the object you last selected on the right side of the sectormap. Many functions are only available on specific objecttypes.

Sectormap keys:

Keys affecting the selected object:

<Return>	Change subtype of selected object
<numpad 4,6,7,8>	Move object slow on x/z plane
<numpad 7,9,1,3>	Move object fast
<numpad 5>	Reset rotation
<numpad 0, DEL>	Rotate object alpha
<numlock, div>	Rotate beta
<numpad \053,->	Rotate gamma
<d>	Destroy object
<n>	Set special object name

Keys affecting only suns:

<r/R,g/G,f/F>	Change color of sun
---------------	---------------------

Global keys for sector:

<s>	Change sector owner
<v>	Change galaxy background
<h/H,j/J,k/K>	Change sector fog color
<o/O>	Change sector fog near dist.
<l/L>	Change sector fog far dist.
<m>	Override music track ID

Object Symbols:

@t@	Warning of 3D polygon collision test
@c@	Warning of object is colliding
@d@	Warning of SQUASH mine exploding

4.1 Structure of types files

Units

Distances, sizes and speed are given using "length units". This unit is historical and 1 Meter equals 222 of these units.

types\Ships.txt

The ship types file can appear in two formats. The version used in the X2 release version has been automatically converted from the original version exported from our object database. There are two ways you can change this file: Either directly modify the version contained on the X2 CD or in patches, or by modifying the original version supplied together with this mod package. Using the original version has a slight disadvantage in speed as the game has to rebuild information about the number of turrets from the "ship scenes". Using the original file is also the only option if you changed mesh or scene files for ships.

File format (original version):

first line: *file version;number of lines;*

The *file version* is currently 13. If the ship types are extended with new parameters, 2 will be added to the version number. If the ship type file is converted to extended version 14, additional laser, cockpit and turret info is added to the types file. *number of lines* is the number of ship records (lines) following this first line.

Remaining lines:

3dbody;	Body number for this object (if the object (ship) has a scene then this is the reference body).
3dcompbody;	Body number used for detail display inside menus.
Rotation Alpha;	The maximum turning rate (in rotations per second) for Yaw.
Rotation Beta;	The maximum turning rate (in rotations per second) for Pitch.
Rotation Gamma;	The maximum turning rate (in rotations per second) for Roll.
GSubType;	Galaxy SubType. For ships this is the ship class (TL, TS,

	M0, M1, M2, M3, M4, M5, TL_P, TS_P, GO, M6, TP)
TextID;	textid of ship class name in textpage 17. textid+1 is the long description of the ship.
SpeedMax;	Maximum speed in units/second.
SpeedDeltaPerSec;	Acceleration in units/s ²
SoundIDOnFlyBy;	3d soundeffect of ship engine.
AverageReactDelay;	
EngineEffect;	Visual effect number used at ship engines
GlowEffect;	
PowerGeneration;	Generated energy per second.
3DSoundVolMin;	Minimum sound volume for ship with speed
3DSoundVolMax;	Maximum sound volume for maximum speed
CutID;	CutID used for ship scene.
CockpitBody;	BodyID or CutID used for Cockpit.
LaserMask;	Bitmask with usable laser subtypes for this ship.
MaxNumLasers;	Max. number of lasers - valid only for converted type file
MaxShieldType;	Max. usable shield subtype.
MaxNumShields;	Max. number of installable shields.
MaxRocketType;	Max. usable rocket subtype.
MaxNumRockets;	unused.
MaxExtraSpeed;	Max. extra speed in 10% steps (10 = +100% Upgrade)
MaxExtraRotationSpeed;	Max. extra rotation speed in 10% steps (10 = +100% Upgrade)
TradeContainerSize;	Default number of ware containers to carry.
MaxTradeContainerSize;	Maximum of installable containers (with container upgrades).
CockpitType1;	Index into static cockpit types for extra cockpit #1
CockpitTextID1;	TextID of extra cockpit #1 (0=Main,1=Front,2=Back,3=Left,4=Right,5=Up,6=Down).
CockpitType2;	
CockpitTextID2;	
CockpitType3;	
CockpitTextID3;	
CockpitType4;	
CockpitTextID4;	
CockpitType5;	
CockpitTextID5;	

CockpitType6;
 CockpitTextID6;
 DockBaySize; Number of ships which can dock at this (carrier) ship.
 MaxTKClass; Maximum Transport Class ship can carry.
 DefaultRace; Default race for this ship type.
 HullStrength;
 ExplosionEffect; Effect for main body of exploding ship.
 PartExplosionEffect; Effect for smaller explosions of ship parts (subbodies)
 EngineTrail; Type of trail - index into types\particles3.txt.

only for converted version 14:

 NumberOfCockpits;

 for each cockpit:

 Class; unique ID defined in ship scene.

 TurretID;

 BodyID;

 PathID;

 NumberOfTurrets;

 for each turret:

 LaserPos; Laser position of first laser in this turret

 NumLasers;

 Class; unique control ID, which cockpit can control which turrets.

 NumGuns;

 for each gun:

 LaserPos;

 NumLasers;

 BodyID; body id of gun body in ship main scene

 PathID; path id of gun body in ship main scene

 BodyID2; body id of gun barrel in gun scene (or -1)

 PathID2; path id of gun barrel in gun scene (or -1)

common entries for all object types:

Volume; Volume of one cargo unit.

RelValue; RelValue * valuefactor for a ship = ship price.

PriMaxP; price variation as primary resource

SecMaxP; price variation as secondary resource

TK: Transport Class (as ware)

4.2 File hierarchy used in X2

The following directories contain:

.../types/	Object type definitions and special effects (textfiles)
.../T/	Text of X2 in XML format. Usually named by a language ID (44 for English) and a serial number. 440001.xml is the main text for X2, 440002.xml is the text used only for startup dialogues.
.../tex/true/	Textures stored either in JPEG or TGA format
.../v/	3d bodyfiles either in textform (bod) or more common in binary form (see 3dsmax exporter)
.../cut/	Cutscenes and object scenes. Similar to the v folder. Scenes are oftentimes combinations of bodies stored in the v folder.
.../scripts/	AI scripts stored in XML format. See script editor documentation above.
.../s/	Sound effects usually stored in WAV format
.../maps/	Universe maps stored in XML format. See galaxy editor documentation for details

Less interesting for modding:

.../L/	Story files. This is the core of the game. Storyfiles can be compiled online on the X2 devnet for level 5 developers
.../L/true/	Additional 2D artwork (loading screens etc.)
.../shader/	Shaders used by engine
.../f/	Font images and size information
.../mov/	Streams containing either speech or movies (the face animations)
.../soundtrack/	Music